Proceedings of the 24th International Workshop on Description Logics

July 13–16, 2011
Barcelona, Spain

Edited by
Riccardo Rosati, Sebastian Rudolph and Michael Zakharyaschev

Preface

Welcome to the 24th International Workshop on Description Logics, DL 2011, in Barcelona, Spain. The workshop continues the long-standing tradition of international workshops devoted to discussing developments and applications of knowledge representation formalisms and systems based on Description Logics. The list of the International Workshops on Description Logics can be found at `http://dl.kr.org`.

There were 59 papers submitted, each of which was reviewed by at least three members of the program committee or additional reviewers recruited by the PC members. Apart from the presentation of the accepted papers, posters, and demos, the program was further enhanced by the following keynotes:

- Marcelo Arenas (Pontificia Universidad Católica de Chile), *Exchanging More than Complete Data.*
- Gert Smolka (Saarland University), *Incremental Decision Procedures for Modal Logic with Nominals and Eventualities.*
- Heiner Stuckenschmidt (Universität Mannheim), *A Little Logic Goes a Long Way – Logical Reasoning in Web Data Integration and Ontology Learning.*

The Best Student Paper Prize (€500) has been awarded to Szymon Klarman (Vrije Universiteit Amsterdam) and Víctor Gutiérrez-Basulto (Universität Bremen) for their paper *Two-Dimensional Description Logics of Context.*

The organizers of the DL 2011 workshop gratefully acknowledge the logistical and financial support of Yahoo, Inc. and Yahoo! Research Barcelona, and the financial support of the Artificial Intelligence Journal. The organization of the workshop also greatly benefited from the help of Barcelona Media, in particular Sònia Campdepadrós Pérez.

Our thanks go to all the authors for submitting to DL, and to the invited speakers, PC members, and all additional reviewers who made the technical program possible. Finally, we would like to acknowledge that the work of the PC was greatly simplifed by using the EasyChair conference management system (`www.easychair.org`) developed by Andrei Voronkov.

Riccardo Rosati, Sebastian Rudolph and Michael Zakharyaschev
DL 2011 Conference and PC chairs

Peter Mika, Estefania Ricart and Natalia Pou
Local organizers

Organizing Committee

General Chair

Riccardo Rosati	Sapienza Universitá di Roma, Italy

Programm Chairs

Sebastian Rudolph	Karlsruhe Institute of Technology, Germany
Michael Zakharyaschev	Birkbeck College London, UK

Local Chair

Peter Mika	Yahoo! Research Barcelona, Spain

Program Committee

Carlos Areces	LORIA
Alessandro Artale	Free University of Bolzano-Bozen
Meghyn Bienvenu	Universität Bremen
Alex Borgida	Rutgers University
Andrea Cali	Birkbeck College London
Diego Calvanese	Free University of Bolzano-Bozen
Bernardo Cuenca Grau	University of Oxford
Giuseppe De Giacomo	Sapienza Universitá di Roma
Achille Fokoue	IBM Research
Enrico Franconi	Free University of Bozen-Bolzano
Birte Glimm	University of Oxford
Rajeev Gore	Australian National University
Stijn Heymans	TU Vienna
Pascal Hitzler	Wright State University
Ian Horrocks	University of Oxford
Ullrich Hustadt	University of Liverpool
Yevgeny Kazakov	University of Oxford
Boris Konev	University of Liverpool
Roman Kontchakov	Birkbeck College London
Markus Krötzsch	University of Oxford
Thomas Lukasiewicz	University of Oxford
Carsten Lutz	Universität Bremen
Thomas Meyer	Meraka Institute
Maja Milicic	University of Manchester
Boris Motik	University of Oxford
Ralf Möller	Hamburg University of Technology

Table of Contents

Exchanging More than Complete Data

Marcelo Arenas

Pontificia Universidad Católica de Chile
marenas@ing.puc.cl

In the traditional data exchange setting source instances are restricted to be complete, in the sense that every fact is either true or false in these instances. Although natural for a typical database translation scenario, this restriction is gradually becoming an impediment to the development of a wide range of applications that need to exchange objects that admit several interpretations. In particular, we are motivated by two specific applications that go beyond the usual data exchange scenario: exchanging incomplete information and exchanging knowledge bases.

In this talk, we propose a general framework for data exchange that can deal with these two applications. More specifically, we address the problem of exchanging information given by representation systems, which are essentially finite descriptions of (possibly infinite) sets of complete instances, and then we show the robustness of our proposal by applying it to the problems of exchanging incomplete information and exchanging knowledge bases, which are both instantiations of the exchanging problem for representation systems.

This is joint work with Jorge Perez and Juan Reutter.

Incremental Decision Procedures for Modal Logic with Nominals and Eventualities

Gert Smolka

Saarland University
smolka@ps.uni-saarland.de

The talk will discuss different decision procedures for modal logic with nominals and eventualities. This logic has an ExpTime-complete decision problem and is not compact. There is a simple and worst-case optimal decision procedure, which is not practical since it is not incremental. I will discuss two incremental procedures, one worst-case optimal procedure for the fragment without nominals, and one not worst-case optimal procedure for the full logic. A main concern will be the correctness arguments for the procedures.

The talk is based on joint work with Mark Kaminski.

A Little Logic Goes a Long Way – Logical Reasoning in Web Data Integration and Ontology Learning

Heiner Stuckenschmidt

Universität Mannheim
heiner@informatik.uni-mannheim.de

There is an ongoing dispute in the Semantic Web Community about the usefulness of (Description) Logic as a basis for describing data on the web. While researchers in logic argue with the benefits of logic in terms of a clean semantics and richness of the language, criticism against the use of logic normally focusses on two points: its computational complexity and its inability to represent soft constraints. In this talk, we will address these criticisms and argue that if used in the right way description logics are a valuable tool for typical tasks on the semantic web. We use problem of semantic matchmaking as an example to show that the use of rather inexpressive logics with good computational properties already provide significant benefits by eliminating incoherent matches. In the second part of the talk we address the problem of dealing with soft constraints and show two solutions to this problem that have proven useful for matchmaking: Approximate subsumption as a purely logical framework for partial matchmaking and Log-Linear Description Logics as a new combination of Description Logics with (log-linear) probabilistic models. We show that purely logical matchmaking achieves results comparable with state of the art matchmaking systems that rely on similarity functions and present results that show that log-linear description logics outperform existing matching systems. We conclude that in the context of semantic web applications expressive power of the logics used is less important than the integration with other formalisms and technologies for improving efficiency and the ability to deal with imperfect knowledge.

Knowledge Base Exchange

Marcelo Arenas[1], Elena Botoeva[2], and Diego Calvanese[2]

[1] Dept. of Computer Science, PUC Chile
marenas@ing.puc.cl
[2] KRDB Research Centre, Free Univ. of Bozen-Bolzano, Italy
lastname@inf.unibz.it

Abstract. In this paper, we study the problem of exchanging knowledge between two knowledge bases (KBs) connected through mappings, with a special interest in exchanging implicit knowledge, not only data like in the traditional database exchange setting. As representation formalism we use Description Logics (DL) that exhibit a reasonable tradeoff between expressive power and complexity of reasoning. Thus, we assume that the source and target KBs are given as a DL TBox+ABox, while the mappings have the form of DL TBox assertions. We study the problem of translating the knowledge in the source KB according to these mappings. We define a general framework of KB exchange, and specify the problems of computing and representing different kinds of solutions, i.e., target KBs with specified properties, given a source KB and a mapping. We then develop first results and techniques and study the complexity of KB exchange for the case of $\textit{DL-Lite}_{\textit{RDFS}}$, a DL that corresponds to the FOL fragment of RDFS.

1 Introduction

In data exchange, data structured under one schema (called source schema) must be restructured and translated into an instance of a different schema (called target schema), and the way in which this restructuring should occur is specified by means of a mapping from the source schema to the target schema [5]. Such a problem has been studied extensively in recent years, under various choices for the languages used to specify the source and target schema, and the mappings [2]. While incomplete information in this setting is introduced by the mapping layer (see also [6]), one fundamental assumption in the works on data exchange is that the source is a (completely specified) database.

In this paper, we go beyond this setting, and consider data exchange in the case where implicit knowledge is present in the source, by which new data may be inferred. We follow the line of the work in [1], where a general framework for data exchange is proposed, in which the source data may be incompletely specified, and thus (possibly infinitely) many source instances are implicitly represented. The framework in [1] in based on the general notion of *representation system*, as a mechanism to represent multiple instances of a data schema, and considers the problem of incomplete data exchanges under mappings constituted by a set of tuple generating dependencies (tgds).

We refine that framework to the case where as a representation system we use Description Logics (DL) knowledge bases (KBs) constituted by a TBox and an ABox, and where mappings are sets of DL inclusions. While in the traditional data exchange setting, given a source instance and a mapping specification, *(universal) solutions* are target instances derived from the source instance and the mapping, in our case solutions are target DL KBs, derived from the source KB and the mapping. Besides a notion of (universal) solution based on the correspondence between models of source and target KBs, we introduce also the weaker notion of *(universal) CQ-solution*, based on the correspondence between answers to conjunctive queries over source and target KBs.

In our setting where we exchange DL KBs, in order to minimize the exchange (and hence transfer and materialization) of explicit (i.e., ABox) information, we are interested in computing universal (CQ-)solutions that contain as much implicit knowledge as possible. This leads us to define a new problem, called *representability*, whose goal is to compute from a source TBox and a mapping, a target TBox that leads to a universal (CQ-)solution when it is combined with a suitable ABox computed from the source ABox, independently of the actual source ABox.

We then develop first results and techniques for KB exchange and for the representability problem in the case of KBs expressed in *DL-Lite*$_{RDFS}$, a DL that corresponds to the FOL fragment of RDFS. *DL-Lite*$_{RDFS}$ is a fragment of *DL-Lite*$_{\mathcal{R}}$ [4] that does not allow for existential quantification (i.e., concepts of the form $\exists R$) in the right-hand side of concept inclusions, nor for disjointness assertions.

The paper is organized as follows. In Section 2 we give preliminary notions on DLs and conjunctive queries (CQs). In Section 3 we define our framework of KB exchange. In Section 4 we present the techniques for deciding the defined reasoning tasks. Finally, in Section 5 we draw some conclusions and outline issues for future work.

2 Preliminaries

We introduce here the necessary notions about the description logic (DL) that we use in this article, and about conjunctive queries, which we adopt as our query formalism.

2.1 *DL-Lite*$_{\mathcal{R}}$ Knowledge Bases

The DLs of the *DL-Lite* family [4] of light-weight description logics are characterized by the fact that reasoning can be done in polynomial time, and that data complexity of reasoning and conjunctive query answering is in AC^0. We adopt here *DL-Lite*$_{\mathcal{R}}$, and present now its syntax and semantics.

Let N_C, N_R, N_a be sets of concept, role, and individual names, respectively, and assume that $A \in N_C$ and $P \in N_R$. In *DL-Lite*$_{\mathcal{R}}$, B and C are used to denote basic and complex concept descriptions, respectively, and R and Q are used to denote basic and complex roles, respectively. These concept and roles constructs are defined by the following grammar:

$$\begin{array}{lcl@{\qquad\qquad}lcl} R & ::= & P \mid P^- & B & ::= & A \mid \exists R \\ Q & ::= & R \mid \neg R & C & ::= & B \mid \neg B \end{array}$$

In the following, for a basic role R, we use R^- to denote P^- when $R = P$, and P when $R = P^-$.

A *DL-Lite*$_\mathcal{R}$ TBox is a finite set of concept inclusions $B \sqsubseteq C$ and role inclusions $R \sqsubseteq Q$. A *DL-Lite*$_\mathcal{R}$ ABox is a finite set of membership assertions of the form $A(a)$ and $P(a, b)$, where $a, b \in N_a$. A *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a *DL-Lite*$_\mathcal{R}$ TBox and $\mathcal{A}$ is a *DL-Lite*$_\mathcal{R}$ ABox. As usual, TBoxes represent implicit knowledge, while ABoxes represent the data.

In the following, we will also make use of a restricted form of *DL-Lite*$_\mathcal{R}$ TBoxes. A *DL-Lite*$_\mathcal{R}$ TBox is said to be *definite* if there are only atomic concepts and atomic roles on the right-hand side of its inclusions. In other words, a definite TBox may not mention in its right-hand side a concept of the form $\exists R$, and may not contain concept or role disjointness assertions. We call *DL-Lite*$_{\mathit{RDFS}}$ the fragment of *DL-Lite*$_\mathcal{R}$ obtained by considering only definite *DL-Lite*$_\mathcal{R}$ TBoxes. Intuitively, *DL-Lite*$_{\mathit{RDFS}}$ corresponds to the fragment of RDFS [3] that is embeddable in FOL (and hence in DLs).

The semantics of *DL-Lite*$_\mathcal{R}$ is defined in the standard way. We just remark that we use $\textsc{Mod}(\mathcal{K})$ to denote the set of all models of KB $\mathcal{K}$. From now on, we assume that interpretations satisfy the standard name assumption, that is, we assume given a fixed infinite set $\mathbf{U}$ of individual names, and we assume that for every interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, it holds that $\Delta^\mathcal{I} \subseteq \mathbf{U}$ and $a^\mathcal{I} = a$ for every $a \in \Delta^\mathcal{I}$. Notice that this implies the Unique Name Assumption (UNA), i.e., different individuals are interpreted as different domain elements.

A *signature* Σ is a set of concept and role names. An interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ is said to be an interpretation of Σ if it is defined exactly on the concept and role names in Σ. Given a KB $\mathcal{K}$, the *signature* $\Sigma(\mathcal{K})$ *of* $\mathcal{K}$ is the alphabet of concept and role names occurring in $\mathcal{K}$, and $\mathcal{K}$ is said to be *defined over* (or simply, *over*) a signature Σ if $\Sigma(\mathcal{K}) \subseteq \Sigma$ (and likewise for a TBox $\mathcal{T}$, an ABox $\mathcal{A}$, a concept inclusions $B \sqsubseteq C$, a role inclusions $R \sqsubseteq Q$, and membership assertions $A(a)$ and $R(a, b)$).

2.2 Conjunctive Queries and Certain Answers

A *conjunctive query (CQ) over a signature* Σ is a first-order formula of the form $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\mathit{conj}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$, $\boldsymbol{y}$ are tuples of variables and $\mathit{conj}(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form: *(1)* $A(t)$, with A a concept name in Σ and t either a constant from $\mathbf{U}$ or a variable from $\boldsymbol{x}$ or $\boldsymbol{y}$, or *(2)* $P(t_1, t_2)$, with P a role name in Σ and t_i $(i = 1, 2)$ either a constant from $\mathbf{U}$ or a variable from $\boldsymbol{x}$ or $\boldsymbol{y}$. In a conjunctive query $q(\boldsymbol{x}) = \exists \boldsymbol{y}.\mathit{conj}(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x}$ is the tuple of free variables of $q(\boldsymbol{x})$. A *union of conjunctive queries (UCQ)* is a formula of the form: $q(\boldsymbol{x}) = \bigvee_{i=1}^{n} \exists \boldsymbol{y}_i.\mathit{conj}_i(\boldsymbol{x}, \boldsymbol{y}_i)$, where each $\mathit{conj}_i(\boldsymbol{x}, \boldsymbol{y}_i)$ is as before. A query q (either a CQ or a UCQ) is said to be a query over a KB $\mathcal{K}$ if q is a query over a signature Σ and $\Sigma \subseteq \Sigma(\mathcal{K})$.

Let q be a CQ $\exists y_1 \cdots \exists y_\ell.\mathit{conj}(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$ over a signature Σ and $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ an interpretation of Σ. Then the answer of q over $\mathcal{I}$, denoted by $q^\mathcal{I}$, is defined as the set of tuples $(a_1, \ldots, a_k)$ of elements from $\Delta^\mathcal{I}$ for which there exist a tuple $(b_1, \ldots, b_\ell)$ of elements from $\Delta^\mathcal{I}$ such that $\mathcal{I}$ satisfies every conjunct in $\mathit{conj}(a_1, \ldots, a_k, b_1, \ldots, b_\ell)$. Moreover, given a UCQ $q = \bigvee_{i=1}^{n} q_i$, the answer of q over an interpretation $\mathcal{I}$, denoted by $q^\mathcal{I}$, is defined as $\bigcup_{i=1}^{n} q_i^\mathcal{I}$. Finally, given a query q (either a CQ or a UCQ) over a KB $\mathcal{K}$, the answer to q over $\mathcal{K}$, denoted by $\mathit{cert}(q, \mathcal{K})$, is defined as $\mathit{cert}(q, \mathcal{K}) = \bigcap_{\mathcal{I} \in \textsc{Mod}(\mathcal{K})} q^\mathcal{I}$. Each tuple in $\mathit{cert}(q, \mathcal{K})$ is called a *certain answer* for q over $\mathcal{K}$.

Certain answers in *DL-Lite*$_\mathcal{R}$ can be characterized through the notion of chase. We call a *chase* a (possibly infinite) set of assertions of the form $A(t)$, $P(t,t')$, where t, t' are either individuals, or *labeled nulls* interpreted as not necessarily distinct domain elements. For *DL-Lite*$_\mathcal{R}$ KBs, we employ the notion of oblivious chase as defined in [4]. For such a KB $\langle\mathcal{T},\mathcal{A}\rangle$, the chase of $\mathcal{A}$ w.r.t. $\mathcal{T}$, denoted *chase*$_\mathcal{T}(\mathcal{A})$, is a chase obtained from $\mathcal{A}$ by adding facts implied by inclusions in $\mathcal{T}$, and introducing labeled nulls whenever required by an inclusion with $\exists R$ in the right-hand side (see [4] for details).

3 Exchanging Knowledge Bases

In this section, we introduce the knowledge exchange framework used in the paper. The starting point to define this framework is the notion of mapping, which has been shown to be of fundamental importance in the context of data exchange [5]. Formally, a *DL-Lite*$_\mathcal{R}$*-mapping* (or just *mapping*) is a tuple $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where Σ_1, Σ_2 are disjoint signatures and $\mathcal{T}_{12}$ is a *DL-Lite*$_\mathcal{R}$ TBox whose inclusions are of the form: (1) $C_1 \sqsubseteq C_2$, where C_1, C_2 are complex concepts over Σ_1 and Σ_2, respectively, and (2) $Q_1 \sqsubseteq Q_2$, where Q_1 and Q_2 are complex roles over Σ_1 and Σ_2, respectively.

Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping. Intuitively, mapping $\mathcal{M}$ specifies how a knowledge base over the vocabulary Σ_1 should be translated into a knowledge base over the vocabulary Σ_2. This intuition is formalized in terms of the notion of *solution*, which is defined as follows. Given an interpretation $\mathcal{I}_1$ of Σ_1 and an interpretation $\mathcal{I}_2$ of Σ_2, pair $(\mathcal{I}_1,\mathcal{I}_2)$ *satisfies* TBox $\mathcal{T}_{12}$, denoted by $(\mathcal{I}_1,\mathcal{I}_2) \models \mathcal{T}_{12}$, if for each concept inclusion $C_1 \sqsubseteq C_2 \in \mathcal{T}_{12}$, it holds that $C_1^{\mathcal{I}_1} \subseteq C_2^{\mathcal{I}_2}$, and for each role inclusion $Q_1 \sqsubseteq Q_2 \in \mathcal{T}_{12}$, it holds that $Q_1^{\mathcal{I}_1} \subseteq Q_2^{\mathcal{I}_2}$. Moreover, given an interpretation $\mathcal{I}$ of Σ_1, $\mathrm{SAT}_\mathcal{M}(\mathcal{I})$ is defined as the set of interpretations $\mathcal{J}$ of Σ_2 such that $(\mathcal{I},\mathcal{J}) \models \mathcal{T}_{12}$, and given a set $\mathcal{X}$ of interpretations of Σ_1, $\mathrm{SAT}_\mathcal{M}(\mathcal{X})$ is defined as:

$$\mathrm{SAT}_\mathcal{M}(\mathcal{X}) = \textstyle\bigcup_{\mathcal{I}\in\mathcal{X}} \mathrm{SAT}_\mathcal{M}(\mathcal{I}).$$

Then the notion of solution under a mapping is defined as follows, by considering this notion of satisfaction and the knowledge exchange framework proposed in [1].

Definition 1. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping, $\mathcal{K}_1$ a KB over Σ_1, and $\mathcal{K}_2$ a KB over Σ_2. Then $\mathcal{K}_2$ is said to be a solution for $\mathcal{K}_1$ under $\mathcal{M}$ if:*

$$\mathrm{MOD}(\mathcal{K}_2) \subseteq \mathrm{SAT}_\mathcal{M}(\mathrm{MOD}(\mathcal{K}_1)).$$

That is, $\mathcal{K}_2$ is a solution for $\mathcal{K}_1$ under $\mathcal{M}$ if for every model $\mathcal{I}_2$ of $\mathcal{K}_2$, there exists a model $\mathcal{I}_1$ of $\mathcal{K}_1$ such that $(\mathcal{I}_1,\mathcal{I}_2) \models \mathcal{T}_{12}$.

Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$. A KB $\mathcal{K}_1$ over Σ_1 can have an infinite number of solutions under $\mathcal{M}$. Thus, it is natural to ask what is a *good* solution for this knowledge base. Next we introduce the notion of universal solution, which is a simple extension of the concept of solution introduced in Definition 1, and is based on the notion of universal solution introduced in [1].

Definition 2. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping, $\mathcal{K}_1$ a KB over Σ_1, and $\mathcal{K}_2$ a KB over Σ_2. Then $\mathcal{K}_2$ is said to be a universal solution for $\mathcal{K}_1$ under $\mathcal{M}$ if:*

$$\mathrm{MOD}(\mathcal{K}_2) = \mathrm{SAT}_\mathcal{M}(\mathrm{MOD}(\mathcal{K}_1)).$$

In the preceding definition, KB $\mathcal{K}_2$ is considered to be a good solution for KB $\mathcal{K}_1$ under mapping $\mathcal{M}$ as the models of $\mathcal{K}_2$ exactly correspond to the valid translations of the models of $\mathcal{K}_1$ according to $\mathcal{M}$. We illustrate Definitions 1 and 2 in an example.

Example 1. Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, where $\Sigma_1 = \{A_1, B_1\}$, $\Sigma_2 = \{A_2, B_2\}$, and $\mathcal{T}_{12} = \{A_1 \sqsubseteq A_2,\ B_1 \sqsubseteq B_2\}$. Furthermore, assume that $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$, where $\mathcal{T}_1 = \{B_1 \sqsubseteq A_1\}$ and $\mathcal{A}_1 = \{B_1(a)\}$. Then the following knowledge bases over Σ_2 are solutions for $\mathcal{K}_1$ under $\mathcal{M}$:

$$\begin{array}{llll} \mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_2 \rangle, & \text{where} & \mathcal{T}_2 = \emptyset, & \mathcal{A}_2 = \{B_2(a), A_2(a)\} \\ \mathcal{K}_2' = \langle \mathcal{T}_2', \mathcal{A}_2' \rangle, & \text{where} & \mathcal{T}_2' = \{B_2 \sqsubseteq A_2\}, & \mathcal{A}_2' = \{B_2(a)\} \end{array}$$

Moreover, $\mathcal{K}_2$ is a universal solution for $\mathcal{K}_1$ under $\mathcal{M}$, while $\mathcal{K}_2'$ is not. In fact, if $\mathcal{I}_2$ is an interpretation of Σ_2 such that: $\Delta^{\mathcal{I}_2} = \{a, b\}$, $A_2^{\mathcal{I}_2} = \{a\}$, and $B_2^{\mathcal{I}_2} = \{a, b\}$, then we have that $\mathcal{I}_2 \notin \textsc{Mod}(\mathcal{K}_2')$ since $\mathcal{I}_2$ does not satisfy inclusion $B_2 \sqsubseteq A_2$, but $\mathcal{I}_2 \in \textsc{Sat}_{\mathcal{M}}(\textsc{Mod}(\mathcal{K}_1))$ since $(\mathcal{I}_1, \mathcal{I}_2) \models \mathcal{T}_{12}$ for $\mathcal{I}_1 \in \textsc{Mod}(\mathcal{K}_1)$ defined as $\Delta^{\mathcal{I}_1} = \{a\}$, $A_1^{\mathcal{I}_1} = \{a\}$ and $B_1^{\mathcal{I}_1} = \{a\}$. ■

In the previous example, $\mathcal{K}_2'$ is not a universal solution for $\mathcal{K}_1$ under $\mathcal{M}$ since inclusion $B_2 \sqsubseteq A_2$ cannot be deduced from the information in $\mathcal{K}_1$ and $\mathcal{M}$. Or, more formally, $\mathcal{K}_2'$ is not a universal solution as $B_2 \sqsubseteq A_2$ is not implied by $\langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle$. However, $\mathcal{K}_2'$ can also be considered as a solution of $\mathcal{K}_1$ that is desirable to materialize, as the implicit knowledge in $\mathcal{K}_2$ (i.e., TBox $\mathcal{T}_2$) represents the implicit knowledge in $\mathcal{K}_1$ (i.e., TBox $\mathcal{T}_1$), given the way that concepts A_1 and B_1 have to be translated according to mapping $\mathcal{M}$. In fact, solution $\mathcal{K}_2'$ is as good as solution $\mathcal{K}_2$ in terms of the information that can be extracted from them by using some specific queries, but with the advantage that $\mathcal{K}_2'$ represents knowledge in a more compact way. In what follows, we introduce a new class of good solutions that captures this intuition with respect to the widely used fragment of CQs.

Definition 3. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping, $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ a KB over Σ_1, and $\mathcal{K}_2$ a KB over Σ_2. Then $\mathcal{K}_2$ is said to be a CQ-solution for $\mathcal{K}_1$ under $\mathcal{M}$ if for every CQ q over Σ_2, we have that $cert(q, \langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle) \subseteq cert(q, \mathcal{K}_2)$.*

Moreover, $\mathcal{K}_2$ is said to be a universal CQ-solution for $\mathcal{K}_1$ under $\mathcal{M}$ if for every CQ q over Σ_2, we have that $cert(q, \langle \mathcal{T}_1 \cup \mathcal{T}_{12}, \mathcal{A}_1 \rangle) = cert(q, \mathcal{K}_2)$.

Notably, we have in Example 1 that both KB $\mathcal{K}_2$ and KB $\mathcal{K}_2'$ are universal CQ-solutions for KB $\mathcal{K}_1$ under mapping $\mathcal{M}$.

A natural question at this point is what is the relationship between the notions of solution presented in this section. The following proposition, which is straightforward to prove, shows that the notion of (universal) CQ-solution is weaker than the notion of (universal) solution.

Proposition 1. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping, $\mathcal{K}_1$ a KB over Σ_1, and $\mathcal{K}_2$ a KB over Σ_2. If $\mathcal{K}_2$ is a (universal) solution for $\mathcal{K}_1$ under $\mathcal{M}$, then $\mathcal{K}_2$ is a (universal) CQ-solution for $\mathcal{K}_1$ under $\mathcal{M}$.*

In this paper, we study several fundamental problems related to the notions of solution presented here. These problems are formally introduced below.

3.1 Knowledge Base Exchange: Reasoning Tasks

In the data exchange scenario [5], as well as in the knowledge exchange scenario [1], the problem of materializing solutions has been identified as the fundamental problem to solve. Given a class $\mathcal{U}$ of mappings (for example, the class of *DL-Lite*$_{\mathcal{R}}$-mappings) and a DL $\mathcal{L}$ (for example, *DL-Lite*$_{\mathcal{R}}$), the problem of computing solutions is defined as follows for $\mathcal{U}$ and $\mathcal{L}$:

PROBLEM	: CompSol($\mathcal{U}, \mathcal{L}$)
INPUT	: A mapping $\mathcal{M} \in \mathcal{U}$ and an $\mathcal{L}$-knowledge base $\mathcal{K}_1$ over Σ_1
TO DO	: Compute an $\mathcal{L}$-knowledge base $\mathcal{K}_2$ over Σ_2 such that $\mathcal{K}_2$ is a solution for $\mathcal{K}_1$ under $\mathcal{M}$

Given a class $\mathcal{U}$ of mappings and a DL $\mathcal{L}$, the computation problems CompUnivSol($\mathcal{U}, \mathcal{L}$), CompCQSol($\mathcal{U}, \mathcal{L}$), and CompUnivCQSol($\mathcal{U}, \mathcal{L}$) are defined exactly as above, but considering universal solutions, CQ-solutions, and universal CQ-solutions instead of solutions, respectively.

In the rest of this paper, we study these problems, and some other fundamental problems associated to them, for a restriction of the class of mappings introduced in this section. More precisely, a mapping $\mathcal{M} = \langle \Sigma_1, \Sigma_2, \mathcal{T}_{12} \rangle$ is said to be *definite* if $\mathcal{T}_{12}$ is a *DL-Lite*$_{RDFS}$ TBox (recall the definition of definite TBoxes in Section 2.1). We use $\mathcal{U}_{\text{def}}$ to denote the class of definite mappings. Then, as our first result, we obtained that the chase can be used to compute universal solutions for definite mappings and *DL-Lite*$_{RDFS}$ TBoxes. In what follows, let $chase_{\mathcal{T},\Sigma}(\mathcal{A})$ denote the projection of $chase_{\mathcal{T}}(\mathcal{A})$ on the signature Σ.

Proposition 2. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a definite mapping and $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ a DL-Lite$_{RDFS}$ KB over Σ_1. Then $\langle \emptyset, chase_{\mathcal{T}_{12},\Sigma_2}(chase_{\mathcal{T}_1}(\mathcal{A}_1)) \rangle$ is a universal solution for $\mathcal{K}_1$ under $\mathcal{M}$.*

Thus, given that for definite mappings and *DL-Lite*$_{RDFS}$ TBoxes, the sets $chase_{\mathcal{T}_1}(\mathcal{A}_1)$ and $chase_{\mathcal{T}_{12},\Sigma_2}(chase_{\mathcal{T}_1}(\mathcal{A}_1))$ are always finite and can be computed in polynomial time [4], we obtain as a corollary of Propositions 1 and 2 that the problems of computing solutions defined above can be solved in polynomial time for definite mappings and *DL-Lite*$_{RDFS}$ TBoxes.

Corollary 1. CompSol($\mathcal{U}_{\text{def}}$, *DL-Lite*$_{RDFS}$), CompUnivSol($\mathcal{U}_{\text{def}}$, *DL-Lite*$_{RDFS}$), CompCQSol($\mathcal{U}_{\text{def}}$, *DL-Lite*$_{RDFS}$), *and* CompUnivCQSol($\mathcal{U}_{\text{def}}$, *DL-Lite*$_{RDFS}$) *can all be solved in polynomial time.*

Unfortunately, the solutions obtained by using Proposition 2 are of little interest to us because the generated target ABoxes can be very large. Hence, we turn our attention to the case of non-empty target TBoxes and, more specifically, to the problem of computing universal CQ-solutions that include as much implicit knowledge as possible. This gives rise to the following *separation properties*.

Definition 4. *Let $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ be a mapping and $\mathcal{T}_1$ a TBox over Σ_1.*
- *$\mathcal{T}_1$ is* representable *in $\mathcal{M}$ if there exists a TBox $\mathcal{T}_2$ over Σ_2 such that for every ABox $\mathcal{A}_1$ over Σ_1, it holds that $\langle \mathcal{T}_2, chase_{\mathcal{T}_{12},\Sigma_2}(\mathcal{A}_1) \rangle$ is a universal CQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ under $\mathcal{M}$. $\mathcal{T}_2$ is called a* representation *of $\mathcal{T}_1$ in $\mathcal{M}$.*

– $\mathcal{T}_1$ *is* weakly representable *in* $\mathcal{M}$ *if there exists a mapping* $\mathcal{M}^\star = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^\star)$ *such that* $\mathcal{T}_{12} \subseteq \mathcal{T}_{12}^\star$*,* $\mathcal{T}_1 \cup \mathcal{T}_{12} \models \mathcal{T}_{12}^\star$ *and* $\mathcal{T}_1$ *is representable in* $\mathcal{M}^\star$*.*

The separation problems are interesting to us because a positive answer would mean that we can construct the TBox of a solution by considering only the input TBox and the mapping, independently of the input ABox. On the other hand, the ABox of the solution can be found simply by translating the input ABox according to the rules in the mapping (and the input TBox). We illustrate the notions just defined in the following example.

Example 2. Let $\Sigma_1 = \{A_1, B_1\}$, $\Sigma_2 = \{A_2, B_2\}$, and $\mathcal{T}_1 = \{B_1 \sqsubseteq A_1\}$. If $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$ with $\mathcal{T}_{12} = \{A_1 \sqsubseteq A_2,\ B_1 \sqsubseteq B_2\}$, then we have that $\mathcal{T}_2 = \{B_2 \sqsubseteq A_2\}$ is a representation of $\mathcal{T}_1$ in $\mathcal{M}$.

On the other hand, if $\mathcal{M}' = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}')$ with $\mathcal{T}_{12}' = \{A_1 \sqsubseteq A_2\}$, then we have that $\mathcal{T}_1$ is not representable in $\mathcal{M}'$: let $\mathcal{A}_1' = \{B_1(a)\}$, then $\mathit{chase}_{\mathcal{T}_{12}', \Sigma_2}(\mathcal{A}_1') = \emptyset$ and for no TBox $\mathcal{T}_2'$, $\langle \mathcal{T}_2', \mathit{chase}_{\mathcal{T}_{12}', \Sigma_2}(\mathcal{A}_1')\rangle$ is a universal CQ-solution to $\langle \mathcal{T}_1, \mathcal{A}_1'\rangle$ under $\mathcal{M}'$. However, if we consider $\mathcal{T}_{12}^\star = \mathcal{T}_{12}' \cup \{B_1 \sqsubseteq A_2\}$, we conclude that $\mathcal{T}_1$ is weakly representable in $\mathcal{M}'$ since $\mathcal{T}_{12}' \subseteq \mathcal{T}_{12}^\star$, $\mathcal{T}_1 \cup \mathcal{T}_{12}' \models \mathcal{T}_{12}^\star$ and $\mathcal{T}_1$ is representable in $\mathcal{M}^\star = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}^\star)$ (in fact, $\emptyset$ is a representation of $\mathcal{T}_1$ in $\mathcal{M}^\star$). Note, that $\mathcal{T}_{12}' \subset \mathcal{T}_{12}$.

Now, if $\mathcal{M}'' = (\Sigma_1, \Sigma_2, \mathcal{T}_{12}'')$ with $\mathcal{T}_{12}'' = \{A_1 \sqsubseteq A_2,\ B_1 \sqsubseteq B_2,\ C_1 \sqsubseteq B_2\}$, then again we have that $\mathcal{T}_1$ is not representable in $\mathcal{M}''$: let $\mathcal{A}_1'' = \{B_1(a),\ C_1(c)\}$, then $\mathit{chase}_{\mathcal{T}_{12}'', \Sigma_2}(\mathcal{A}_1'') = \{B_2(a),\ B_2(c)\}$. The query $q() \leftarrow A_2(a)$ evaluates to true in $\langle \mathcal{T}_1 \cup \mathcal{T}_{12}'', \mathcal{A}_1''\rangle$, hence in order for a TBox $\mathcal{T}_2''$ to be a representation of $\mathcal{T}_1$ in $\mathcal{M}''$, it must contain the inclusion $B_2 \sqsubseteq A_2$. On the other hand, it implies that the query $q'() \leftarrow A_2(c)$ evaluates to true in $\langle \mathcal{T}_2'', \mathit{chase}_{\mathcal{T}_{12}'', \Sigma_2}(\mathcal{A}_1'')\rangle$, whereas it evaluates to false in $\langle \mathcal{T}_1 \cup \mathcal{T}_{12}'', \mathcal{A}_1''\rangle$. However, again if we consider $\mathcal{T}_{12}^{\star\star} = \mathcal{T}_{12}'' \cup \{B_1 \sqsubseteq A_2\}$, we conclude that $\mathcal{T}_1$ is weakly representable in $\mathcal{M}''$. ■

4 Techniques for Deciding Representability

We address now the problem of deciding (weak) representability of a TBox in a mapping, for the case where TBoxes are expressed in *DL-Lite*$_{\mathit{RDFS}}$ and mappings are definite. We start by showing some properties that characterize solutions in terms of chases.

In the following for two chases $\mathcal{C}_1$ and $\mathcal{C}_2$, a *homomorphism* from $\mathcal{C}_1$ to $\mathcal{C}_2$ is a mapping h from the individuals and labeled nulls in $\mathcal{C}_1$ to those in $\mathcal{C}_2$ such that *(1)* h is the identity on the individuals, *(2)* if $A(t) \in \mathcal{C}_1$ then $A(h(t)) \in \mathcal{C}_2$, and *(3)* if $P(t, t') \in \mathcal{C}_1$ then $P(h(t), h(t')) \in \mathcal{C}_2$. We write $\mathcal{C}_1 \to \mathcal{C}_2$ if there is a homomorphism from $\mathcal{C}_1$ to $\mathcal{C}_2$, and $\mathcal{C}_1 \leftrightarrows \mathcal{C}_2$ if $\mathcal{C}_1 \to \mathcal{C}_2$ and $\mathcal{C}_2 \to \mathcal{C}_1$.

From now on, we assume Σ_1 and Σ_2 as given, and for a mapping $\mathcal{M} = (\Sigma_1, \Sigma_2, \mathcal{T}_{12})$, we use simply $\mathcal{M}$ to denote also $\mathcal{T}_{12}$. Then we have the following characterizations of solutions in terms of chases, which are similar to the characterizations in [1].

Proposition 3. *Let* $\mathcal{M}$ *be a definite mapping,* $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1\rangle$ *a DL-Lite*$_{\mathit{RDFS}}$ *KB over* Σ_1*, and* $\mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_2\rangle$ *a DL-Lite*$_{\mathit{RDFS}}$ *KB over* Σ_2*. If* $\mathit{chase}_{\mathcal{M}, \Sigma_2}(\mathit{chase}_{\mathcal{T}_1}(\mathcal{A}_1)) \to \mathit{chase}_{\mathcal{T}_2}(\mathcal{A}_2)$*, then* $\mathcal{K}_2$ *is a CQ-solution for* $\mathcal{K}_1$ *under* $\mathcal{M}$*.*

Corollary 2. *Let $\mathcal{M}$ be a definite mapping, $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ a DL-Lite$_{RDFS}$ KB over Σ_1, and $\mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_2 \rangle$ a DL-Lite$_{RDFS}$ KB over Σ_2. Then $\mathcal{K}_2$ is a universal CQ-solution for $\mathcal{K}_1$ under $\mathcal{M}$ if $chase_{\mathcal{M},\Sigma_2}(chase_{\mathcal{T}_1}(\mathcal{A}_1)) \leftrightarrows chase_{\mathcal{T}_2}(\mathcal{A}_2)$.*

4.1 Checking Representability of a TBox in a Mapping

We address now the *representability* problem, which is to decide, given a TBox $\mathcal{T}_1$ over Σ_1 and a mapping $\mathcal{M}$, whether $\mathcal{T}_1$ is representable in $\mathcal{M}$ (cf. Definition 4).

We start by considering the decision problem associated with representability:

> Given a mapping $\mathcal{M}$, a TBox $\mathcal{T}_1$ over Σ_1, and a TBox $\mathcal{T}_2$ over Σ_2, check whether $\mathcal{T}_2$ is a representation of $\mathcal{T}_1$ in $\mathcal{M}$, i.e., for each ABox $\mathcal{A}_1$ over Σ_1, $\langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle$ is a universal CQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$.

For definite mappings and *DL-Lite$_{RDFS}$* TBoxes, the decision problem associated with representability can be solved in two steps:

1. Check whether for each ABox $\mathcal{A}_1$ over Σ_1, $\langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle$ is a CQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$.
2. Check whether for each ABox $\mathcal{A}_1$ over Σ_1 and for each CQ q over Σ_2, we have that $cert(q, \langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle) \subseteq cert(q, \langle \mathcal{T}_1 \cup \mathcal{M}, \mathcal{A}_1\rangle)$.

If both checks succeed, then $\mathcal{T}_2$ is a representation of $\mathcal{T}_1$ in $\mathcal{M}$, otherwise it is not. We develop now techniques to perform these two tests.

Step 1: *Checking whether for each ABox $\mathcal{A}_1$ over Σ_1, $\langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle$ is a CQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$.*

We introduce now some notions that help us to characterize when a mapping is able to "translate" the inclusions in $\mathcal{T}_1$ to the target TBox.

We use $pn(X)$ to denote A if X is A, and P if X is $\exists P$, $\exists P^-$, P, or P^-. For X, Y *DL-Lite$_{RDFS}$* expressions, we say that X is *compatible with* Y if *(i)* $pn(X) = pn(Y)$, and *(ii)* if X is $\exists R$, then Y is $\exists R$, R, or R^-. For a *DL-Lite$_{RDFS}$* inclusion $\alpha = X_1 \sqsubseteq X_2$, we use $lhs(\alpha)$ to denote X_1, and $rhs(\alpha)$ to denote X_2. We say that α is *left-compatible* (resp., *right-compatible*) with X, if X is compatible with $lhs(\alpha)$ (resp., $rhs(\alpha)$).

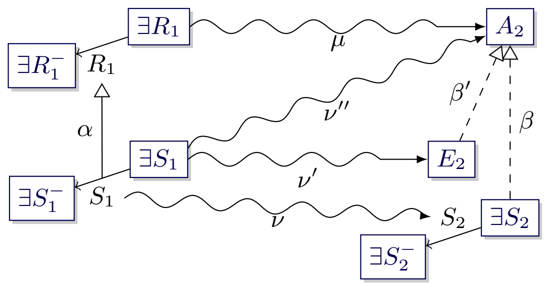
Let $\mathcal{M}$ be a definite mapping, $\alpha = N_1 \sqsubseteq M_1$ a *DL-Lite$_{RDFS}$* inclusion over Σ_1, and $\mu \in \mathcal{M}$ left-compatible with M_1. Then the *translation set* of α and μ in $\mathcal{M}$, denoted $\mathsf{M}(\alpha, \mu, \mathcal{M})$, is the set of *DL-Lite$_{RDFS}$* inclusions over Σ_2 such that, if there

Table 1. Definition of $\mathsf{M}(\alpha, \mu, \mathcal{M})$. A_i, E_i are atomic concepts, R_i, S_i are basic roles.

α	μ	ν	β
$A_1 \sqsubseteq E_1$	$E_1 \sqsubseteq E_2$	$A_1 \sqsubseteq A_2$	$A_2 \sqsubseteq E_2$
$\exists R_1 \sqsubseteq E_1$	$E_1 \sqsubseteq E_2$	$\exists R_1 \sqsubseteq A_2$	$A_2 \sqsubseteq E_2$
		$R_1 \sqsubseteq R_2$	$\exists R_2 \sqsubseteq E_2$
$R_1 \sqsubseteq S_1$	$S_1 \sqsubseteq S_2$	$R_1 \sqsubseteq R_2$	$R_2 \sqsubseteq S_2$
	$\exists S_1 \sqsubseteq E_2$	$\exists R_1 \sqsubseteq A_2$	$A_2 \sqsubseteq E_2$
		$R_1 \sqsubseteq R_2$	$\exists R_2 \sqsubseteq E_2$
	$\exists {S_1}^- \sqsubseteq E_2$	$\exists {R_1}^- \sqsubseteq A_2$	$A_2 \sqsubseteq E_2$
		$R_1 \sqsubseteq R_2$	$\exists {R_2}^- \sqsubseteq E_2$

exists an inclusion ν in $\mathcal{M}$ left-compatible with N_1, then $\beta \in \mathsf{M}(\alpha, \mu, \mathcal{M})$, where β is uniquely defined by α, μ, ν as shown in Table 1. When the mapping $\mathcal{M}$ is clear from the context, we write $\mathsf{M}(\alpha, \mu)$. We say that α is *redundant* w.r.t. $\mu = M_1' \sqsubseteq M_2$ (in $\mathcal{M}$) if $M_2 \sqsubseteq M_2 \in \mathsf{M}(\alpha, \mu)$. We explain these definitions in an example.

Example 3. Let $\alpha = S_1 \sqsubseteq R_1$ and $\mu = \exists R_1 \sqsubseteq A_2$.



Let $\nu = S_1 \sqsubseteq S_2$ and $\nu' = \exists S_1 \sqsubseteq E_2$ be in $\mathcal{M}$. Then $\{\exists S_2 \sqsubseteq A_2, E_2 \sqsubseteq A_2\} \subseteq \mathsf{M}(\alpha, \mu, \mathcal{M})$. If $\nu'' = \exists S_1 \sqsubseteq A_2 \in \mathcal{M}$, then α is redundant w.r.t. μ.

If none of ν, ν' and ν'' is in $\mathcal{M}$, then $\mathsf{M}(\alpha, \mu, \mathcal{M})$ is empty. ■

With these notions in place, we can characterize CQ-solutions in the context of the representability problem.

Proposition 4. *Let $\mathcal{M}$ be a definite mapping, $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1, and $\mathcal{T}_2$ a DL-Lite$_{RDFS}$ TBox over Σ_2. Then for each ABox $\mathcal{A}_1$, the KB $\langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle$ is a CQ-solution for $\langle \mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$ if and only if for each inclusion α, s.t. $\mathcal{T}_1 \models \alpha$, and for each inclusion $\mu \in \mathcal{M}$ left-compatible with $rhs(\alpha)$, there exists $\beta \in \mathsf{M}(\alpha, \mu)$ such that $\mathcal{T}_2 \models \beta$.*

Step 2: *Checking whether for each ABox $\mathcal{A}_1$ over Σ_1, and for each CQ q over Σ_2, we have that $cert(q, \langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle) \subseteq cert(q, \langle \mathcal{T}_1 \cup \mathcal{M}, \mathcal{A}_1\rangle)$.*

Recall the definition of the translation set. Now, let $\beta = N_2 \sqsubseteq M_2$ be a *DL-Lite$_{RDFS}$* inclusion over Σ_2, and $\nu \in \mathcal{M}$ right-compatible with N_2. Then the *reverse-translation set* of β and ν in $\mathcal{M}$, denoted $\mathsf{M}^-(\beta, \nu, \mathcal{M})$, is the set of *DL-Lite$_{RDFS}$* inclusions over Σ_1 such that, if there exists an inclusion μ in $\mathcal{M}$ right-compatible with M_2, then $\alpha \in \mathsf{M}^-(\beta, \nu, \mathcal{M})$, where α is uniquely defined by β, ν, and μ as shown in Table 1.

Proposition 5. *Let $\mathcal{M}$ be a definite mapping, $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1, and $\mathcal{T}_2$ a DL-Lite$_{RDFS}$ TBox over Σ_2. Then for each ABox $\mathcal{A}_1$ over Σ_1 and for each CQ q over Σ_2, $cert(q, \langle \mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle) \subseteq cert(q, \langle \mathcal{T}_1 \cup \mathcal{M}, \mathcal{A}_1\rangle)$ if and only if for each inclusion β, s.t. $\mathcal{T}_2 \models \beta$, and for each inclusion $\nu \in \mathcal{M}$ right-compatible with $lhs(\beta)$, there exists $\alpha \in \mathsf{M}^-(\beta,\nu)$, s.t. $\mathcal{T}_1 \models \alpha$.*

With the techniques for Steps 1 and 2 at hand, we are ready to characterize representations.

Corollary 3. *Let $\mathcal{M}$ be a definite mapping, $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1, and $\mathcal{T}_2$ a DL-Lite$_{RDFS}$ TBox over Σ_2. Then for each ABox $\mathcal{A}_1$ over Σ_1, $\langle\mathcal{T}_2, chase_{\mathcal{M},\Sigma_2}(\mathcal{A}_1)\rangle$ is a universal CQ-solution for $\langle\mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$ iff*

- *for each inclusion α, s.t. $\mathcal{T}_1 \models \alpha$, and for each inclusion $\mu \in \mathcal{M}$ left-compatible with $rhs(\alpha)$, there exists $\beta \in \mathsf{M}(\alpha, \mu)$ s.t. $\mathcal{T}_2 \models \beta$, and*
- *for each inclusion β, s.t. $\mathcal{T}_2 \models \beta$, and for each inclusion $\nu \in \mathcal{M}$ right-compatible with $lhs(\beta)$, there exists $\alpha \in \mathsf{M}^-(\beta,\nu)$, s.t. $\mathcal{T}_1 \models \alpha$.*

Now, we can check whether a given $\mathcal{T}_1$ is representable in a given $\mathcal{M}$.

Theorem 1. *Let $\mathcal{M}$ be a definite mapping and $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1. Then we can check whether $\mathcal{T}_1$ is representable in $\mathcal{M}$ in polynomial time.*

4.2 Checking Weak Representability

We can easily solve the weak representability problem for *DL-Lite$_{RDFS}$* KBs even if the mappings are arbitrary tgds, i.e., assertions of the form $q_1 \to q_2$, mapping a CQ q_1 over Σ_1 to a CQ q_2 over Σ_2 of the same arity as q_1. We call such a mapping a tgd-mapping. Let $\mathcal{T}_1$ be a *DL-Lite$_{RDFS}$* TBox and $\mathcal{M}$ a tgd-mapping. We can enrich $\mathcal{M}$ by compiling knowledge from $\mathcal{T}_1$ into it:

- Let $q_1 \to q_2$ be a tgd in $\mathcal{M}$, with q_1 a CQ over Σ_1 and q_2 a CQ over Σ_2. Let the UCQ $Q_1 = \{q_1^1, \ldots, q_1^k\}$ be the perfect reformulation of q_1 w.r.t. $\mathcal{T}_1$ (see, e.g., [4]). Then we extend $\mathcal{M}$ with a tgd $q_1^i \to q_2$ for each $q_1^i \in Q_1$.
- We perform this for each tgd in $\mathcal{M}$. The resulting mapping is denoted with $\mathcal{M}^*$.

Proposition 6. *Let $\mathcal{M}$ be a tgd-mapping, $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1, and $\mathcal{M}^*$ the mapping constructed as described above. Then for each ABox $\mathcal{A}_1$ over Σ_1, the KB $\langle\emptyset, chase_{\mathcal{M}^*,\Sigma_2}(\mathcal{A}_1)\rangle$ is a universal CQ-solution for $\langle\mathcal{T}_1, \mathcal{A}_1\rangle$ under $\mathcal{M}$.*

From this we immediately get:

Theorem 2. *Let $\mathcal{M}$ be a definite mapping and $\mathcal{T}_1$ a DL-Lite$_{RDFS}$ TBox over Σ_1. Then $\mathcal{T}_1$ is weakly representable in $\mathcal{M}$.*

Thus, when the source KB is expressed in *DL-Lite$_{RDFS}$* it is always possible to separate data by enriching mappings, even if mappings are tgds. When $\mathcal{M}$ is a set of tgds, the size of $\mathcal{M}^*$ might be exponential in the size of $\mathcal{M}$. If $\mathcal{M}$ is a *DL-Lite$_{RDFS}$* mapping, the size of $\mathcal{M}^*$ is always polynomial in the size of $\mathcal{M}$.

5 Conclusions

We have specialized the framework for KB exchange proposed in [1] to the case of DLs, i.e., the source and target KBs are given as DL KBs and the mappings have the form of DL TBox assertions. Moreover, we have defined a new reasoning problem: representability of a TBox in a mapping, whose goal is to compute from a source TBox and a mapping, a target TBox that leads to a universal (CQ-)solution when it is combined with a suitable ABox computed from the source ABox, independently of the actual source ABox. A variation of this problem, called weak representability, allows for modification of the mapping, so that the source TBox becomes representable.

Then, we have developed first results and techniques for KB exchange and for the representability problem in the case of mappings and KBs expressed in *DL-Lite*$_{RDFS}$ (such mappings are called definite). *DL-Lite*$_{RDFS}$ is a fragment of *DL-Lite*$_{\mathcal{R}}$ that does not allow for existential quantification (i.e., concepts of the form $\exists R$) in the right-hand side of concept inclusions, nor for disjointness assertions. It implies, first, that the chase is always finite in *DL-Lite*$_{RDFS}$, and, second, that KBs are always consistent. We have shown the following results for definite mappings and *DL-Lite*$_{RDFS}$ KBs: *(i)* the problems of computing (universal) (CQ-)solutions can be solved in polynomial time; *(ii)* the problem of representability of a TBox in a mapping is decidable in polynomial time; *(iii)* every *DL-Lite*$_{RDFS}$ TBox is weakly representable in a definite mapping.

The main direction for future work is to extend the results to the case of full *DL-Lite*$_{\mathcal{R}}$. This brings up the problem of labelled nulls in the chase, which becomes infinite in general. Moreover, due to the possible presence of disjointness assertions in TBoxes, consistency of the source and target KBs has to be checked.

Acknowledgments

The authors are grateful to Vladislav Ryzhikov and Evgeny Sherkhonov for helpful discussions, and to the anonymous referees for many helpful comments. The authors were supported by Marie Curie action "International Research Staff Exchange Scheme (IRSES)", FP7-PEOPLE-2009-IRSES, under grant number 24761. M. Arenas was also supported by Fondecyt grant 1090565. E. Botoeva and D. Calvanese were also supported by the EU under the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), grant agreement number FP7-257593, and under the large-scale integrating project (IP) OntoRule (ONTOlogies meet Business RULEs), grant agreement number FP7-231875.

References

1. M. Arenas, J. Pérez, and J. L. Reutter. Data exchange beyond complete data. In *Proc. of the 30th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2011)*, pages 83–94, 2011.
2. B. Barcelò. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.
3. D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/rdf-schema/`.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.
6. L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. *J. of Computer and System Sciences*, 77(3):542–571, 2011.

DL-Lite with Attributes and Sub-Roles (Extended Abstract)

A. Artale,[1] A. Ibáñez García,[1] R. Kontchakov,[2] and V. Ryzhikov[1]

[1] KRDB Research Centre
Free University of Bozen-Bolzano, Italy
{*lastname*}@inf.unibz.it

[2] Dept. of Comp. Science and Inf. Sys.
Birkbeck College, London, UK
roman@dcs.bbk.ac.uk

1 Introduction

The *DL-Lite family* of description logics has recently been proposed and investigated in [5–7] and later extended in [1, 8, 3]. The relevance of the *DL-Lite* family is witnessed by the fact that it forms the basis of OWL 2 QL, one of the three profiles of OWL 2 (http://www.w3.org/TR/owl2-profiles/). According to the official W3C profiles document, the purpose of OWL 2 QL is to be the language of choice for applications that use very large amounts of data.

This paper extends the *DL-Lite* languages of [3] by relaxing the restriction on the interaction between cardinality constraints ($\mathcal{N}$) and role inclusions (or hierarchies, $\mathcal{H}$). We also introduce a new family of languages, $\textit{DL-Lite}_{\alpha}^{\mathcal{HNA}}$, $\alpha \in \{\textit{core}, \textit{krom}, \textit{horn}, \textit{bool}\}$, extending *DL-Lite* with *attributes* ($\mathcal{A}$).

The notion of attributes, borrowed from conceptual modeling formalisms, introduces a distinction between (abstract) objects and application domain values, and consequently, between concepts (sets of objects) and datatypes (sets of values), and between roles (relating objects to objects) and attributes (relating objects to values). The advantage of the presented languages over $\textit{DL-Lite}_{\mathcal{A}}$ [8] is that the range restrictions for attributes can be *local* (and not only *global* as in $\textit{DL-Lite}_{\mathcal{A}}$). Indeed, $\textit{DL-Lite}_{\alpha}^{\mathcal{HNA}}$ has a possibility to express concept inclusion axioms of the form $C \sqsubseteq \forall U.T$, for an attribute U and its datatype T. In this way, we allow re-use of the very same attribute associated to different concepts with different range restrictions. For example, we can say that employees' salary is of type *Integer*, researchers' salary is in the range 35,000–70,000 (enumeration type) and professors' salary in the range 55,000–100,000—while both researchers and professors are employees. Note that local attributes are strictly more expressive than global ones. For example, concept disjointness (or unsatisfiability) can be inferred just from datatype disjointness for the same (existentially qualified) attribute. Since *DL-Lite* languages have been proved useful in capturing conceptual data models [8, 4, 2], the extension with attributes, as presented here, will generalize their use even further.

We aim at establishing computational complexity of knowledge base satisfiability in these new DLs. In particular we prove the following results:

1. We can relax the restrictions presented in [3] limiting the interaction between sub-roles and number restrictions without increasing the complexity of reasoning as far as the problem is limited to TBox satisfiability checking. As

for KB satisfiability, the presence of the ABox should be taken into account if we want to preserve the complexity results.
2. The introduction of *local* range restrictions for attributes is for free for the languages *DL-Lite*$^{\mathcal{NA}}_{bool}$, *DL-Lite*$^{\mathcal{NA}}_{horn}$ and *DL-Lite*$^{\mathcal{NA}}_{core}$.

2 The Description Logic *DL-Lite*$^{\mathcal{HNA}}_{bool}$

The language of *DL-Lite*$^{\mathcal{HNA}}_{bool}$ contains *object names* $a_0, a_1, \ldots$, *value names* $v_1, v_2, \ldots$, *concept names* $A_0, A_1, \ldots$, *role names* $P_0, P_1, \ldots$, *attribute names* $U_0, U_1, \ldots$, and *datatype names* $T_0, T_1, \ldots$. Complex *roles* R and *concepts* C are defined below:

$$\begin{array}{rcl} R & ::= & P_i \mid P_i^-, \\ B & ::= & \top \mid \bot \mid A_i \mid \geq q\, R \mid \geq q\, U_i \\ C & ::= & B \mid \neg C \mid C_1 \sqcap C_2, \end{array}$$

where q is a positive integer. The concepts of the form B are called *basic concepts*.

A *DL-Lite*$^{\mathcal{HNA}}_{bool}$ *TBox*, $\mathcal{T}$, is a finite set of concept, role, attribute and datatype *inclusion axioms* of the form:

$$C_1 \sqsubseteq C_2, \quad C \sqsubseteq \forall U.T, \quad R_1 \sqsubseteq R_2, \quad U \sqsubseteq U', \quad T \sqsubseteq T', \quad T \sqcap T' \sqsubseteq \bot,$$

and an *ABox*, $\mathcal{A}$, is a finite set of assertions of the form:

$$A_k(a_i), \quad \neg A_k(a_i), \quad P_k(a_i, a_j), \quad \neg P_k(a_i, a_j), \quad U_k(a_i, v_j) \quad \text{and} \quad T_k(v_j).$$

We standardly abbreviate $\geq 1\, R$ and $\geq 1\, U$ by $\exists R$ and $\exists U$, respectively. Absence of an attribute (i.e., $C \sqsubseteq \forall U.\bot$) can be expressed as $C \sqcap \exists U \sqsubseteq \bot$.

Together, a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ constitute the *DL-Lite*$^{\mathcal{HNA}}_{bool}$ *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. In the following, we denote by *role*$(\mathcal{K})$ and *att*$(\mathcal{K})$ the sets of role and attribute names occurring in $\mathcal{K}$, respectively; *role*$^{\pm}(\mathcal{K})$ denotes the set $\{P_k, P_k^- \mid P_k \in role(\mathcal{K})\}$.

Semantics. As usual in description logic, an *interpretation*, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, consists of a nonempty *domain* $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation domain $\Delta^{\mathcal{I}}$ is the union of two non-empty *disjoint* sets: the *domain of objects* $\Delta^{\mathcal{I}}_O$ and the *domain of values* $\Delta^{\mathcal{I}}_V$. We assume that all interpretations agree on the semantics assigned to each datatype T_i, as well as on each constant v_i. In particular, $T_i^{\mathcal{I}} = val(T_i) \subseteq \Delta^{\mathcal{I}}_V$ is the set of values of datatype T_i, and each v_i is interpreted as one specific value, denoted $val(v_i)$, i.e., $v_i^{\mathcal{I}} = val(v_i) \in val(T_i)$. Furthermore, $\cdot^{\mathcal{I}}$ assigns to each object name a_i an element $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}_O$, to each concept name A_k a subset $A_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_O$ of the domain of objects, to each role name P_k a binary relation $P_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_O$ over the domain of objects, and to each attribute name U_k a binary relation $U_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_V$. We adopt here the *unique name assumption* (UNA): $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$, for all $i \neq j$. The role and concept constructs are interpreted in $\mathcal{I}$ in the standard way:

$$\begin{array}{rcll}
(P_k^-)^{\mathcal{I}} & = & \{(y,x) \in \varDelta_O^{\mathcal{I}} \times \varDelta_O^{\mathcal{I}} \mid (x,y) \in P_k^{\mathcal{I}}\}, & \text{(inverse role)}\\
\top^{\mathcal{I}} & = & \varDelta_O^{\mathcal{I}}, & \text{(domain of objects)}\\
\bot^{\mathcal{I}} & = & \emptyset, & \text{(the empty set)}\\
(\geq q\, R)^{\mathcal{I}} & = & \big\{x \in \varDelta_O^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}}\} \geq q\big\}, & \text{(at least } q\ R\text{-successors)}\\
(\geq q\, U)^{\mathcal{I}} & = & \big\{x \in \varDelta_O^{\mathcal{I}} \mid \sharp\{v \mid (x,v) \in U^{\mathcal{I}}\} \geq q\big\}, & \text{(at least } q\ U\text{-attributes)}\\
(\forall U.\, T)^{\mathcal{I}} & = & \big\{x \in \varDelta_O^{\mathcal{I}} \mid \forall v.\, (x,v) \in U^{\mathcal{I}} \to v \in T^{\mathcal{I}}\big\}, & \text{(attribute value restriction)}\\
(\neg C)^{\mathcal{I}} & = & \varDelta_O^{\mathcal{I}} \setminus C^{\mathcal{I}}, & \text{(not in } C)\\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}, & \text{(both in } C_1 \text{ and in } C_2)
\end{array}$$

where $\sharp X$ is the cardinality of X. The *satisfaction relation* $\models$ is also standard:

$$\begin{array}{rll@{\qquad}rll}
\mathcal{I} \models C_1 \sqsubseteq C_2 & \text{iff} & C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}, & \mathcal{I} \models R_1 \sqsubseteq R_2 & \text{iff} & R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}},\\
\mathcal{I} \models T_1 \sqsubseteq T_2 & \text{iff} & T_1^{\mathcal{I}} \subseteq T_2^{\mathcal{I}}, & \mathcal{I} \models U_1 \sqsubseteq U_2 & \text{iff} & U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}},\\
\mathcal{I} \models T_1 \sqcap T_2 \sqsubseteq \bot & \text{iff} & T_1^{\mathcal{I}} \cap T_2^{\mathcal{I}} = \emptyset, & \mathcal{I} \models P_k(a_i,a_j) & \text{iff} & (a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \in P_k^{\mathcal{I}},\\
\mathcal{I} \models A_k(a_i) & \text{iff} & a_i^{\mathcal{I}} \in A_k^{\mathcal{I}}, & \mathcal{I} \models \neg P_k(a_i,a_j) & \text{iff} & (a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \notin P_k^{\mathcal{I}}\\
\mathcal{I} \models \neg A_k(a_i) & \text{iff} & a_i^{\mathcal{I}} \notin A_k^{\mathcal{I}}, & \mathcal{I} \models U_k(a_i,v_j) & \text{iff} & (a_i^{\mathcal{I}}, v_j^{\mathcal{I}}) \in U_k^{\mathcal{I}},\\
 & & & \mathcal{I} \models T_k(v_j) & \text{iff} & v_j^{\mathcal{I}} \in T_k^{\mathcal{I}}.
\end{array}$$

A KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is said to be *satisfiable* (or *consistent*) if there is an interpretation, $\mathcal{I}$, satisfying all the members of $\mathcal{T}$ and $\mathcal{A}$. In this case we write $\mathcal{I} \models \mathcal{K}$ (as well as $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$) and say that $\mathcal{I}$ is a *model of* $\mathcal{K}$ (of $\mathcal{T}$ and $\mathcal{A}$).

2.1 Fragments of *DL-Lite*$_{bool}^{\mathcal{HNA}}$

We consider various syntactical restrictions on the language of *DL-Lite*$_{bool}^{\mathcal{HNA}}$ along two axes: (i) the Boolean operators ($_{bool}$) on concepts, (ii) the role and attribute inclusions ($\mathcal{H}$). Similarly to classical logic, we adopt the following definitions. A TBox $\mathcal{T}$ will be called a *Krom TBox*[1] if its concept inclusions are restricted to:

$$B_1 \ \sqsubseteq\ B_2, \qquad B_1 \ \sqsubseteq\ \neg B_2 \qquad \text{and} \qquad \neg B_1 \ \sqsubseteq\ B_2, \tag{Krom}$$

(here and below all the B_i and B are basic concepts). $\mathcal{T}$ will be called a *Horn TBox* if its concept inclusions are restricted to:

$$\bigsqcap_k B_k \ \sqsubseteq\ B. \tag{Horn}$$

Finally, we call $\mathcal{T}$ a *core TBox* if its concept inclusions are restricted to:

$$B_1 \ \sqsubseteq\ B_2 \qquad \text{and} \qquad B_1 \sqcap B_2 \ \sqsubseteq\ \bot. \tag{core}$$

[1] The Krom fragment of first-order logic consists of all formulas in prenex normal form whose quantifier-free part is a conjunction of binary clauses.

As $B_1 \sqsubseteq \neg B_2$ is equivalent to $B_1 \sqcap B_2 \sqsubseteq \bot$, core TBoxes can be regarded as sitting in the intersection of Krom and Horn TBoxes. In this paper we study the following logics, for $\alpha \in \{\mathit{core}, \mathit{krom}, \mathit{horn}, \mathit{bool}\}$:

***DL-Lite*$_{krom}^{\mathcal{HNA}}$, *DL-Lite*$_{horn}^{\mathcal{HNA}}$, *DL-Lite*$_{core}^{\mathcal{HNA}}$** are the fragments of *DL-Lite*$_{bool}^{\mathcal{HNA}}$ with Krom, Horn, and core TBoxes, respectively;

***DL-Lite*$_{\alpha}^{\mathcal{HN}}$** is the fragment of *DL-Lite*$_{\alpha}^{\mathcal{HNA}}$ without attributes and datatypes;

***DL-Lite*$_{\alpha}^{\mathcal{NA}}$** is the fragment of *DL-Lite*$_{\alpha}^{\mathcal{HNA}}$ without role and attribute inclusions.

As shown in [3], reasoning in *DL-Lite*$_{\alpha}^{\mathcal{HN}}$ is already rather costly (ExpTime-complete) due to the interaction between role inclusions and number restrictions. However, both of these constructs turn out to be useful for the purposes of conceptual modeling. By limiting their interplay one can get languages with a better computational properties [8, 3]. Before presenting such limitations we need to introduce some notation. For a role R, let $\mathit{inv}(R) = P_k^-$ if $R = P_k$ and $\mathit{inv}(R) = P_k$ if $R = P_k^-$. Given a TBox $\mathcal{T}$ we denote by $\sqsubseteq_{\mathcal{T}}^*$ the reflexive and transitive closure of the relation $\{(R, R'), (\mathit{inv}(R), \mathit{inv}(R')) \mid R \sqsubseteq R' \in \mathcal{T}\}$. We say that $R \equiv_{\mathcal{T}}^* R'$ iff $R \sqsubseteq_{\mathcal{T}}^* R'$ and $R' \sqsubseteq_{\mathcal{T}}^* R$. Say that R' is a *proper sub-role* of R in $\mathcal{T}$ if $R' \sqsubseteq_{\mathcal{T}}^* R$ and $R \not\sqsubseteq_{\mathcal{T}}^* R'$. A proper sub-role R' of R is said to be a *direct sub-role* of R if there is no other proper sub-role R'' of R such that R' is a proper sub-role of R''; the set of direct sub-roles of R is denoted as $\mathit{dsub}_{\mathcal{T}}(R)$.

The language *DL-Lite*$_{\alpha}^{(\mathcal{HN})}$ [3] is the result of imposing the following syntactic restriction on *DL-Lite*$_{\alpha}^{\mathcal{HN}}$ TBoxes $\mathcal{T}$:

(inter) if R has a proper sub-role in $\mathcal{T}$ then $\mathcal{T}$ contains no negative occurrences of number restrictions $\geq q\ R$ or $\geq q\ \mathit{inv}(R)$ with $q \geq 2$

(an occurrence of a concept on the right-hand (left-hand) side of a concept inclusion is called *negative* if it is in the scope of an odd (even) number of negations $\neg$; otherwise it is called *positive*). We will formulate two alternative versions of restriction **(inter)**.

Definition 1. *Given a TBox $\mathcal{T}$ and a role $R \in \mathit{role}^{\pm}(\mathcal{T})$, we define the following parameters:*

$$\mathit{ubound}(R,\mathcal{T}) = \min\big(\{\infty\} \cup \{\, q-1 \mid q \geq 2 \text{ and } \geq q\, R \text{ occurs negatively in } \mathcal{T}\}\big),$$
$$\mathit{lbound}(R,\mathcal{T}) = \max\big(\{0\} \cup \{q \mid \geq q\, R \text{ occurs positively in } \mathcal{T}\}\big),$$
$$\mathit{rank}(R,\mathcal{T}) = \max\big(\mathit{lbound}(R,\mathcal{T}), \textstyle\sum_{R' \in \mathit{dsub}_{\mathcal{T}}(R)} \mathit{rank}(R',\mathcal{T})\big),$$
$$\mathit{rank}(R,\mathcal{A}) = \max\big(\{0\} \cup \{n \mid R_i(a,a_i) \in \mathcal{A}, R_i \sqsubseteq_{\mathcal{T}}^* R, \text{ for distinct } a_1,\dots,a_n\}\big).$$

Consider the languages obtained from *DL-Lite*$_{\alpha}^{\mathcal{HN}}$ by imposing one of the following two restrictions:

(inter1) for every $R \in \mathit{role}^{\pm}(\mathcal{T})$, if R has a proper sub-role in $\mathcal{T}$ then $\mathit{ubound}(R,\mathcal{T}) \geq \mathit{rank}(R,\mathcal{T})$;

(inter2) for every $R \in \mathit{role}^{\pm}(\mathcal{T})$, if R has a proper sub-role in $\mathcal{T}$ then $\mathit{ubound}(R,\mathcal{T}) \geq \mathit{rank}(R,\mathcal{T}) + \max\{1, \mathit{rank}(R,\mathcal{A})\}$.

language	**(inter)** [3]	**(inter1)**	**(inter2)**	**non-restrict.**
$DL\text{-}Lite^{\mathcal{HN}}_{core}$	NLogSpace [3]	$\geq$NP [Th.2]	NLogSpace [Th.1]	ExpTime [3]
$DL\text{-}Lite^{\mathcal{HN}}_{horn}$	PTime [3]		PTime [Th.1]	
$DL\text{-}Lite^{\mathcal{HN}}_{krom}$	NLogSpace [3]		NLogSpace [Th.1]	
$DL\text{-}Lite^{\mathcal{HN}}_{bool}$	NP [3]		NP [Th.1]	
$DL\text{-}Lite^{\mathcal{HNA}}_{core}$	NLogSpace [Th.3]	$\geq$NP [Th.2]	NLogSpace [Th.3]	ExpTime
$DL\text{-}Lite^{\mathcal{HNA}}_{horn}$	PTime [Th.3]		PTime [Th.3]	
$DL\text{-}Lite^{\mathcal{HNA}}_{krom}$	NP [Th.4]		NP [Th.4]	
$DL\text{-}Lite^{\mathcal{HNA}}_{bool}$	NP [Th.3]		NP [Th.3]	
$DL\text{-}Lite^{\mathcal{NA}}_{core}$	NA	NA	NA	NLogSpace [Th.3]
$DL\text{-}Lite^{\mathcal{NA}}_{horn}$				PTime [Th.3]
$DL\text{-}Lite^{\mathcal{NA}}_{krom}$				NP [Th. 4]
$DL\text{-}Lite^{\mathcal{NA}}_{bool}$				NP [Th.3]

Table 1: Complexity of *DL-Lite* logics (NA = Non-Applicable).

These new restrictions are in some way weaker than **(inter)** and, for example, allow for the specialization of functional roles: KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{\geq 2\,R \sqsubseteq \bot,\ R_1 \sqsubseteq R_2,\ R_2 \sqsubseteq R\}$, and $\mathcal{A} = \{R(a,b), R_1(a_1,b_1), R_2(a_2,b_2)\}$ does not satisfy **(inter)**, but it satisfies both **(inter1)** and **(inter2)**. Finally, the above restrictions can also be applied to sub-attributes in the languages $DL\text{-}Lite^{\mathcal{HNA}}_{\alpha}$. Table 1 summarizes the obtained complexity results (with numerical parameters q coded in binary).

3 Reasoning in $DL\text{-}Lite^{\mathcal{HN}}_{\alpha}$

In this section, we investigate the complexity of deciding KB satisfiability in languages $DL\text{-}Lite^{\mathcal{HN}}_{\alpha}$ under the restrictions **(inter1)** and **(inter2)**, respectively.

We adapt the proof presented in [3], where a $DL\text{-}Lite^{\mathcal{HN}}_{bool}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is encoded into a sentence $\mathcal{K}^{\ddagger_e}$ in the one-variable first-order logic $\mathcal{QL}^1$. We use a slightly longer but simpler encoding. Every $a_i \in ob(\mathcal{A})$ is associated to the individual constant a_i of $\mathcal{QL}^1$, and every concept name A_i to the unary predicate $A_i(x)$. For each concept $\geq q\,R$ in $\mathcal{K}$ we introduce a fresh unary predicate $E_qR(x)$. For each role name $P_k \in role^{\pm}(\mathcal{K})$, two individual constants dp_k and dp_k^- are introduced, as representatives of the objects in the domain and range of P_k, respectively. The encoding C^* of a concept C is defined inductively:

$$\begin{array}{lll} \bot^* = \bot, & (A_i)^* = A_i(x), & (\geq q\,R)^* = E_qR(x), \\ \top^* = \top, & (\neg C)^* = \neg C^*(x), & (C_1 \sqcap C_2)^* = C_1^*(x) \wedge C_2^*(x). \end{array}$$

The $\mathcal{QL}^1$ sentence encoding the knowledge base $\mathcal{K}$ is defined as follows:

$$\mathcal{K}^{\ddagger_e} \quad = \quad \forall x \big[\mathcal{T}^*(x) \ \wedge\ \mathcal{T}^{\mathcal{R}}(x) \ \wedge \bigwedge_{R \in role^{\pm}(\mathcal{K})} \big(\epsilon_R(x) \ \wedge\ \delta_R(x)\big)\big] \ \wedge\ \mathcal{A}^{\ddagger_e}.$$

Formulas $\mathcal{T}^*(x)$, the $\delta_R(x)$, for $R \in \mathit{role}^{\pm}(\mathcal{K})$, and $T^{\mathcal{R}}(x)$ encode the TBox $\mathcal{T}$:

$$\mathcal{T}^*(x) = \bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \big(C_1^*(x) \to C_2^*(x)\big), \qquad \delta_R(x) = \bigwedge_{q,q' \in Q^R_{\mathcal{T}},\ q' > q} \big(E_{q'}R(x) \to E_qR(x)\big),$$

$$\mathcal{T}^{\mathcal{R}}(x) = \bigwedge_{R \sqsubseteq^*_{\mathcal{T}} R'} \bigwedge_{q \in Q^R_{\mathcal{T}}} \big(E_qR(x) \to E_qR'(x)\big),$$

where $Q^R_{\mathcal{T}}$ contains 1, all q such that $\geq q\,R$ occurs in $\mathcal{T}$ and all $Q^{R'}_{\mathcal{T}}$, for $R' \sqsubseteq^*_{\mathcal{T}} R$. Sentence $\mathcal{A}^{\ddagger_{\mathsf{e}}}$ encodes the ABox $\mathcal{A}$:

$$\mathcal{A}^{\ddagger_{\mathsf{e}}} = \bigwedge_{A_k(a_i) \in \mathcal{A}} A_k(a_i) \ \wedge \bigwedge_{\neg A_k(a_i) \in \mathcal{A}} \neg A_k(a_i) \ \wedge \bigwedge_{\substack{a,a' \in ob(\mathcal{A}) \\ R' \sqsubseteq^*_{\mathcal{T}} R,\ R'(a,a') \in \mathcal{A}}} E_{q^{\mathsf{e}}_{R,a}}R(a) \ \wedge \bigwedge_{\substack{\neg P_k(a_i,a_j) \in \mathcal{A} \\ R(a_i,a_j) \in \mathcal{A},\ R \sqsubseteq^*_{\mathcal{T}} P_k}} \bot,$$

where $q^{\mathsf{e}}_{R,a}$ is the maximum number in $Q^R_{\mathcal{T}}$ such that there are $q^{\mathsf{e}}_{R,a}$ many distinct a_i with $R_i(a,a_i) \in \mathcal{A}$ and $R_i \sqsubseteq^*_{\mathcal{T}} R$. For each $R \in \mathit{role}^{\pm}(\mathcal{K})$, we also need the following formula expressing the fact that the range of R is not empty whenever its domain is non-empty:

$$\epsilon_R(x) = E_1R(x) \to \mathit{inv}(E_1R(dr)),$$

where $\mathit{inv}(E_1R(dr))$ is $E_1P_k^-(dp_k^-)$ if $R = P_k$ and $E_1P_k(dp_k)$ if $R = P_k^-$.

Lemma 1. *A DL-Lite$^{\mathcal{HN}}_{bool}$ knowledge base under restriction* **(inter2)** *is satisfiable iff the $\mathcal{QL}^1$-sentence $\mathcal{K}^{\ddagger_{\mathsf{e}}}$ is satisfiable.*

Proof. (Sketch) The only challenging direction is ($\Leftarrow$). To prove it, we adapt the proofs of Theorem 5.2 and Lemma 5.14 in [3]. The idea of the proof is to construct a *DL-Lite*$^{\mathcal{HN}}_{bool}$ interpretation $\mathcal{I}$, from $\mathfrak{M}$, the minimal Herbrand model of $\mathcal{K}^{\ddagger_{\mathsf{e}}}$. We denote the interpretations of unary predicates P and constants a of $\mathcal{QL}^1$ in $\mathfrak{M}$ by $P^{\mathfrak{M}}$ and $a^{\mathfrak{M}}$, respectively. Let $D = ob(\mathcal{A}) \cup \{dp_k, dp_k^- \mid P_k \in \mathit{role}(\mathcal{K})\}$ be the domain of $\mathfrak{M}$. Then $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is defined inductively: $\Delta^{\mathcal{I}} = \bigcup_{m=0}^{\infty} W_m$, such that W_0 is the set $D_0 = ob(\mathcal{A})$, and for every $a_i \in ob(\mathcal{A})$, $a_i^{\mathcal{I}} = a_i^{\mathfrak{M}}$. Each set W_{m+1}, $m \geq 0$, is constructed by adding to W_m fresh *copies* of certain elements from $D \setminus ob(\mathcal{A})$. The extensions $A_k^{\mathcal{I}}$ of concept names A_k are defined by taking

$$A_k^{\mathcal{I}} = \{w \in \Delta^{\mathcal{I}} \mid \mathfrak{M} \models A_k^*[cp(w)]\}, \tag{1}$$

where $cp(w)$ is the element $d \in D$ of which w is a copy.

The interpretation for each role P_k, is defined inductively as $P_k^{\mathcal{I}} = \bigcup_{m=0}^{\infty} P_k^m$, where $P_k^m \subseteq W_m \times W_m$, along with the construction of $\Delta^{\mathcal{I}}$. The initial interpretation for each role name P_k is defined as follows:

$$P_k^0 = \{(a_i^{\mathfrak{M}}, a_j^{\mathfrak{M}}) \in W_0 \times W_0 \mid R(a_i, a_j) \in \mathcal{A} \text{ and } R \sqsubseteq^*_{\mathcal{T}} P_k\}. \tag{2}$$

For every $R \in \mathit{role}^{\pm}(\mathcal{K})$, the *required R-rank* $r(R,d)$ of $d \in D$ is defined by taking $r(R,d) = \max\big(\{0\} \cup \{q \in Q^R_{\mathcal{T}} \mid \mathfrak{M} \models E_qR[d]\}\big)$. The *actual R-rank* $r_m(R,w)$ of a point $w \in \Delta^{\mathcal{I}}$ at step m is

$$r_m(R,w) = \begin{cases} \sharp\{w' \in W_{m+1} \mid (w,w') \in P_k^{m+1}\}, & \text{if } R = P_k, \\ \sharp\{w' \in W_{m+1} \mid (w',w) \in P_k^{m+1}\}, & \text{if } R = P_k^-. \end{cases}$$

Assume that W_m and P_k^m, $m \geq 0$, have been already defined. Let $W_{m+1} = \emptyset$ and $P_k^{m+1} = \emptyset$, for each role name P_k. If we had $r_m(R, w) = r(R, cp(w))$, for each role R and $w \in W_m$, then the interpretation we need would be constructed. However, the actual rank of some points could still be smaller than the required rank. We cure these defects by adding R-successors for them. Note that the 'curing' process for a given w and R, not only increases the actual R-rank of w, but also all its R'-ranks, for all $R \sqsubseteq_{\mathcal{T}}^* R'$. At this point we adapt the construction in [3] to obtain the interpretation $\mathcal{I}$ we are intending. For each $P_k \in \mathit{role}(\mathcal{K})$, we consider two sets of defects in P_k^m: $\Lambda_k^m = \{w \in W_m \setminus W_{m-1} \mid r_m(P_k, w) < r(P_k, cp(w))\}$ and $\Lambda_k^{m-} = \{w \in W_m \setminus W_{m-1} \mid r_m(P_k^-, w) < r(P_k^-, cp(w))\}$.

In each equivalence class $[R] = \{S \mid S \equiv_{\mathcal{T}}^* R\}$ we select a single role, a *representative*. Let $G = (\mathit{Rep}_{\mathcal{T}}^*, E)$ be a directed graph such that $\mathit{Rep}_{\mathcal{T}}^*$ is the set of representatives and $(R, R') \in E$ iff R is a proper sub-role of R'. Clearly, G is a directed acyclic graph and so, by a topological sort, one can assign to each representative a unique number smaller than the number of all its descendants in G. We use the ascending total order induced on G when choosing an element P_k in $\mathit{Rep}_{\mathcal{T}}^*$, and extend in that way W_m and P_k^m to W_{m+1} and P_k^{m+1}, respectively.

(Λ_k^m) Let $w \in \Lambda_k^m$, $q = r(P_k, cp(w)) - r_m(P_k, w)$, $d = cp(w)$. There is $q' \geq q > 0$ with $\mathfrak{M} \models E_{q'} P_k[d]$. Then, $\mathfrak{M} \models E_1 P_k[d]$ and $\mathfrak{M} \models E_1 P_k^-[dp_k^-]$. In this case we take q *fresh* copies $w_1', \dots, w_q'$ of dp_k^-, add them to W_{m+1} and for each $1 \leq i \leq q$, set $cp(w_i') = dp_k^-$, add the pairs (w, w_i') to each P_j^{m+1} with $P_k \sqsubseteq_{\mathcal{T}}^* P_j$ and the pairs (w_i', w) to each P_j^{m+1} with $P_k^- \sqsubseteq_{\mathcal{T}}^* P_j$ (note that by adding pairs to P_j^{m+1} we change its the actual rank);

(Λ_k^{m-}) This rule is the mirror image of (Λ_k^m): P_k and dp_k^- are replaced with P_k^- and dp_k, respectively.

We need to show that, for all $w \in \Delta^{\mathcal{I}}$ and all $\geq q\, R$ in $\mathcal{T}$,

(a$_1$) if $\geq q\, R$ occurs positively in $\mathcal{T}$ then $\mathfrak{M} \models E_q R[cp(w)]$ implies $w \in (\geq q\, R)^{\mathcal{I}}$;

(a$_2$) if $\geq q\, R$ occurs negatively in $\mathcal{T}$ then $w \in (\geq q\, R)^{\mathcal{I}}$ implies $\mathfrak{M} \models E_q R[cp(w)]$.

Consider first $w \in W_0$. It should be clear that actual R-rank of w

$$r_0(R, w) \quad \leq \quad \mathit{rank}(R, \mathcal{A}) + \textstyle\sum_{R' \in \mathit{dsub}_{\mathcal{T}}(R)} \mathit{rank}(R', \mathcal{T})$$

and so, by **(inter2)**, the total number of R-successors before we cure the defects does not exceed $\mathit{ubound}(R, \mathcal{T})$. If $\mathit{ubound}(R, \mathcal{T}) = \infty$ then there are no negative occurrences of $\geq q\, R$ with $q \geq 2$ and, although may have $r_m(R, w) \geq r(R, cp(w))$ after curing the defects of R, both **(a$_1$)** and **(a$_2$)** hold. Otherwise, we have $\mathit{ubound}(R, \mathcal{T}) + 1 \in Q_{\mathcal{T}}^R$ and so, by **(inter2)**, $\max Q_{\mathcal{T}}^R > \mathit{rank}(R, \mathcal{T}) + \mathit{rank}(R, \mathcal{A})$, whence $r_0(R, w) < \max Q_{\mathcal{T}}^R$. So, as $r(R, cp(w)) \leq \mathit{lbound}(R, \mathcal{T})$ and $\mathit{lbound}(R, \mathcal{T}) < \mathit{ubound}(R, \mathcal{T}) < \max Q_{\mathcal{T}}^R$, after curing the defect, we will have $r_m(R, w) = r(R, cp(w))$, for all $m > 0$, and both **(a$_1$)** and **(a$_2$)** hold. The case with $w \in W_{m_0} \setminus W_{m_0 - 1}$, for $m_0 > 0$ is similar, only now

$$r_{m_0}(R, w) \quad \leq \quad 1 + \textstyle\sum_{R' \in \mathit{dsub}_{\mathcal{T}}(R)} \mathit{rank}(R', \mathcal{T}).$$

Finally, we show that $\mathcal{I} \models \varphi$ for each $\varphi \in \mathcal{K}$. For $\varphi = A_k(a_i)$, $\varphi = \neg A_k(a_i)$ the claim is by the definition of $A_k^{\mathcal{I}}$. For $\varphi = \neg P_k(a_i, a_j)$, we have $(a_i, a_j) \in P_k^{\mathcal{I}}$ iff($a_i, a_j) \in P_k^0$ iff $R(a_i, a_j) \in \mathcal{A}$ and $R \sqsubseteq_{\mathcal{T}}^* P_k$. By induction on the structure of concepts and (a_1) and (a_2), one can show that $\mathcal{I} \models C_1 \sqsubseteq C_2$ whenever $\mathfrak{M} \models \forall x(C_1^*(x) \to C_2^*(x))$, for each $\varphi = C_1 \sqsubseteq C_2$. Finally, $\mathcal{I} \models \varphi$ holds by definition in case $\varphi = R_1 \sqsubseteq R_2 \in \mathcal{T}$.

Theorem 1. *Under restriction* **(inter2)***, checking KB satisfiability is* NP*-complete in DL-Lite$_{bool}^{\mathcal{HN}}$,* PTime*-complete in DL-Lite$_{horn}^{\mathcal{HN}}$ and* NLogSpace*-complete in both DL-Lite$_{krom}^{\mathcal{HN}}$ and DL-Lite$_{core}^{\mathcal{HN}}$.*

We now consider the case where the restriction **(inter1)** is imposed on the interaction between sub-roles and number restrictions. In presence of an ABox, **(inter2)** restricts the number of R-successors in the ABox, which appears to be a strong constraint on the instances of the ABox. On the other hand, the less restrictive condition **(inter1)**, which does not impose any bound on R-successors in the ABox, does not come for free, as shown by the following theorem:

Theorem 2. *Under restriction* **(inter1)***, checking KB satisfiability is* NP*-hard even in DL-Lite$_{core}^{\mathcal{HN}}$.*

Proof. We show that graph 3-colorability can be reduced to KB satisfiability. Let $G = (V, E)$ be a graph with vertices V and edges E and $\{r, g, b\}$ be three colors. Consider the following KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with role names v_i and w and object names o, r, g, b and the x_i, for each vertex $v_i \in V$:

$$\begin{aligned}\mathcal{T} =& \{\geq (|V| + 4)\, w \sqsubseteq \bot\} \cup \{v_i \sqsubseteq w,\ B_1 \sqsubseteq \exists v_i,\ B_2 \sqcap \exists v_i^- \sqsubseteq \bot \mid v_i \in V\} \cup \\ &\{\exists v_i^- \sqcap \exists v_j^- \sqsubseteq \bot \mid (v_i, v_j) \in E\},\\ \mathcal{A} =& \{B_1(o), w(o, r), w(o, g), w(o, b)\} \cup \{w(o, x_i), B_2(x_i) \mid v_i \in V\}.\end{aligned}$$

It can be shown that $\mathcal{K}$ is satisfiable iff G is 3-colorable.

4 Reasoning with Attributes

In this section we study the effect of extending *DL-Lite* with attributes. In particular, we show that for the Bool, Horn and core cases the addition of attributes does not change the complexity of KB satisfiability.

Theorem 3. *KB satisfiability is* NP*-complete in DL-Lite$_{bool}^{\mathcal{NA}}$,* PTime*-complete in DL-Lite$_{horn}^{\mathcal{NA}}$ and* NLogSpace*-complete in DL-Lite$_{core}^{\mathcal{NA}}$.*

Under restriction **(inter2)***, checking KB satisfiability is* NP*-complete in DL-Lite$_{bool}^{\mathcal{HNA}}$,* PTime*-complete in DL-Lite$_{horn}^{\mathcal{HNA}}$ and* NLogSpace*-complete in DL-Lite$_{core}^{\mathcal{HNA}}$.*

Proof. (Sketch) We encode a *DL-Lite$_{\alpha}^{\mathcal{HNA}}$* KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in a $\mathcal{QL}^1$ sentence $\mathcal{K}^{\ddagger a}$ in a way similar to the translation used in Lemma 1. Denote by $val(\mathcal{A})$ the set of all value names that occur in $\mathcal{A}$. Similarly to roles, we define the sets $Q_{\mathcal{T}}^U$

of natural numbers for all occurrences of $\geq q\,U$ (including sub-attributes). We need a unary predicate $E_qU(x)$, for each attribute name U and $q \in Q^U_{\mathcal{T}}$, denoting the set of objects with at least q values of attribute U. We also need, for each attribute name U and each datatype T, a unary predicates $UT(x)$, denoting all objects that may have attribute U values only of datatype T. Following this intuition, we extend $\cdot^*$ by the following two statements:

$$(\geq q\,U)^* = E_qU(x) \qquad \text{and} \qquad (\forall U.T)^* = UT(x).$$

The $\mathcal{QL}^1$ sentence encoding the KB $\mathcal{K}$ is defined as follows:

$$\mathcal{K}^{\ddagger_{\mathbf{a}}} \;=\; \mathcal{K}^{\ddagger_{\mathbf{e}}} \;\wedge\; \forall x\,\bigl[\mathcal{T}^{\mathcal{U}}(x) \wedge \bigwedge_{U \in att(\mathcal{K})} \bigl(\delta_U(x) \wedge \alpha^1_U(x) \wedge \alpha^2_U(x)\bigr)\bigr] \;\wedge\; \mathcal{A}^{\ddagger_{\mathbf{a}}} \;\wedge\; \mathcal{A}^{\ddagger^2_{\mathbf{a}}},$$

where $\mathcal{K}^{\ddagger_{\mathbf{e}}}$ is as before, $\mathcal{T}^{\mathcal{U}}(x)$, $\delta_U(x)$ and $\mathcal{A}^{\ddagger_{\mathbf{a}}}$ are similar to $\mathcal{T}^{\mathcal{R}}(x)$, $\delta_R(x)$ and $\mathcal{A}^{\ddagger_{\mathbf{e}}}$, but rephrased for attributes and their inclusions. The new types of ABox assertions require the following formula:

$$\mathcal{A}^{\ddagger^2_{\mathbf{a}}} = \bigwedge_{U_k(a_i,v_j)\in\mathcal{A}} \;\bigwedge_{\text{datatype } T} \bigl(UT(a_i) \to Tv_j\bigr) \quad\wedge\quad \bigwedge_{T(v_j)\in\mathcal{A}} Tv_j,$$

where Tv_j is a propositional variable for each datatype T and each $v_j \in val(\mathcal{A})$. The two additional formulas, $\alpha^1_U(x)$ and $\alpha^2_U(x)$, capturing datatype inclusions and disjointness constraints are:

$$\begin{aligned}
\alpha^1_U(x) &= \bigwedge_{T \sqsubseteq T' \in \mathcal{T}} \bigl(UT(x) \to UT'(x)\bigr),\\
\alpha^2_U(x) &= \bigwedge_{T \sqcap T' \sqsubseteq \bot \in \mathcal{T}} \bigl[\bigl(UT(x) \wedge UT'(x) \wedge E_1U(x) \to \bot\bigr) \wedge \bigwedge_{v \in val(\mathcal{A})} \bigl(Tv \wedge T'v \to \bot\bigr)\bigr].
\end{aligned}$$

We would like to note here that the formula $\alpha^2_U(x)$ for disjoint datatypes demonstrates a subtle interaction between attribute range constraints $\forall U.T$ and minimal cardinality constraints $\exists U$.

We show that $\mathcal{K}$ is satisfiable iff the $\mathcal{QL}^1$-sentence $\mathcal{K}^{\ddagger_{\mathbf{a}}}$ is satisfiable. For ($\Leftarrow$), let $\mathfrak{M} \models \mathcal{K}^{\ddagger_{\mathbf{a}}}$. We construct a model $\mathcal{I} = (\Delta^{\mathcal{I}}_O \cup \Delta^{\mathcal{I}}_V, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ similarly to the way we proved Lemma 1 but this time datatypes will have to be taken into account: let $\Delta^{\mathcal{I}}_O$ be defined inductively as before and $\Delta^{\mathcal{I}}_O = val(\mathcal{A}) \cup V$. The set V will be constructed starting from $val(\mathcal{A})$ in order to 'cure' the attribute successors as follows. For each datatype T and each attribute U, let

$$T^0 = \{v \in val(\mathcal{A}) \mid \mathfrak{M} \models Tv\} \quad \text{and} \quad U^0 = \{(a,v) \mid U(a,v) \in \mathcal{A}\}.$$

For every attribute $U \in att(\mathcal{K})$, we can define the required U-rank $r(U,d)$ of $d \in D$ and the actual U-rank $r_0(R,w)$ of $w \in \Delta^{\mathcal{I}}_O$ as before, treating U as a role name (the only difference is that there will be only one step, and so, the actual rank is needed only for step 0). We can also consider the equivalence relation induced by the sub-attribute relation in $\mathcal{T}$, then we can choose representatives and a linear order on them respecting the sub-attribute relation of $\mathcal{T}$. We can start from the smaller attributes and 'cure' their defects. Let $w \in \Delta_O^{\mathcal{I}}$ and $q = r(U, cp(w)) - r_0(U, w) > 0$. Take q fresh elements $v_1, \dots, v_q$, add those fresh values to V, add pairs $(w, v_1), \dots, (w, v_q)$ to U^0 and add $v_1, \dots, v_q$ to T^0 for each datatype T with $\mathfrak{M} \models UT[cp(w)]$. Let $U^{\mathcal{I}}$ and $T^{\mathcal{I}}$ be the resulting relations. Now, it can be shown that if $\mathfrak{M} \models \mathcal{K}^{\ddagger\mathbf{a}}$ then $\mathcal{I} \models \varphi$ for every $\varphi \in \mathcal{K}$. We only note here that fresh values v_j cannot be added to two disjoint datatypes T and T' because of formula $\alpha_U^2(x)$.

Now, given a KB with a Bool or Horn TBox, $\mathcal{K}^{\ddagger\mathbf{a}}$ is a universal one-variable formula or a universal one-variable Horn formula, respectively, which immediately gives the NP and PTime upper complexity bounds for the Bool and Horn fragments. The NLogSpace upper bound for KBs with core TBoxes is not so straightforward because $\alpha_U^2(x)$ is not a binary clause. In this case we note that $\mathcal{K}^{\ddagger\mathbf{a}}$ is still a universal one-variable Horn formula and therefore, $\mathcal{K}^{\ddagger\mathbf{a}}$ is satisfiable iff it is true in the 'minimal' model. The minimal model can be constructed in the bottom-to-top fashion by using only positive clauses of $\mathcal{K}^{\ddagger\mathbf{a}}$ (i.e., clauses of the form $\forall x\,(B_1(x) \land \dots \land B_k(x) \to H(x)))$ and then checking whether the negative clauses of $\mathcal{K}^{\ddagger\mathbf{a}}$ (i.e., clauses of he form $\forall x\,(B_1(x) \land \dots \land B_k(x) \to \bot))$ hold in the constructed model. By inspection of the structure of $\mathcal{K}^{\ddagger\mathbf{a}}$, one can see that all its positive clauses are in fact binary, and therefore, whether an atom is true in its minimal model or not can be checked in NLogSpace.

It is of interest to note that the complexity of KB satisfiability increases in the case of Krom TBoxes:

Theorem 4. *KB satisfiability is* NP*-complete in DL-Lite*$_{krom}^{\mathcal{NA}}$*, and so, in DL-Lite*$_{krom}^{\mathcal{HNA}}$ *even under* **(inter)** *and* **(inter2)**.

Proof. (Sketch) The proof exploits the ternary disjointness formula $\alpha_U^2(x)$ in $\mathcal{K}^{\ddagger\mathbf{a}}$. In fact, if $T \sqcap T' \sqsubseteq \bot \in \mathcal{T}$ then the following concept inclusion, although not in the syntax of *DL-Lite*$_{krom}^{\mathcal{NA}}$, is a logical consequence of $\mathcal{T}$ (cf. $\alpha_U^2(x)$):

$$\forall U.T \sqcap \forall U.T' \sqcap \exists U \sqsubseteq \bot.$$

Using such ternary intersections one can encode 3SAT. Let $\varphi = \bigwedge_{i=1}^{m} C_i$ be a 3CNF, where the C_i are ternary clauses over variables $p_1, \dots, p_n$. Now, suppose $p_{j_i^1} \lor \neg p_{j_i^2} \lor p_{j_i^3}$ is the ith clause of φ. It is equivalent to $\neg p_{j_i^1} \land p_{j_i^2} \land \neg p_{j_i^3} \to \bot$ and so, can be encoded as follows:

$$T_i^1 \sqcap T_i^2 \sqsubseteq \bot, \qquad \neg A_{j_i^1} \sqsubseteq \forall U_i.T_i^1, \qquad A_{j_i^2} \sqsubseteq \forall U_i.T_i^2, \qquad \neg A_{j_i^3} \sqsubseteq \exists U_i,$$

where the $A_1, \dots, A_n$ are concept names for the variables $p_1, \dots, p_n$, and U_i is an attribute and T_i^1 and T_i^2 are datatypes for the ith clause (note that Krom concept inclusions of the form $\neg B \sqsubseteq B'$ are required, which is not allowed in the core TBoxes). Let $\mathcal{T}$ consist of all such inclusions for clauses in φ. It can be seen that φ is satisfiable iff $\mathcal{T}$ is satisfiable.

5 Conclusions

We studied two different extensions of the *DL-Lite* logics. First, *local attributes* allow to use the same attribute associated to different concepts with different datatype range restrictions. We showed that the extension with attributes is harmless with the only notable exception of the Krom fragment, where the complexity rises from NLogSpace to NP.

Second, we consider weak syntactic restrictions on interaction between cardinality constraints and role inclusions and study their impact on the complexity of satisfiability. For example, under **(inter)** [3], roles with sub-roles cannot have maximum cardinality constraints. We present two alternative restrictions, which coincide without ABoxes, and show that the complexity of TBox satisfiability under them coincides with the complexity of TBox satisfiability without role inclusions. However, if we want to preserve complexity of KB reasoning, condition **(inter2)** imposes a bound on the number R-successors in the ABox. Indeed, under the weaker condition **(inter1)** complexity of KB satisfiability rises to at least NP (even for the core fragment).

As a future work, we intend to fill the gaps in Table 1 and, in particular, to see whether the NP-hardness results have a matching upper bound. We are also working on query answering in the languages with attributes.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. *DL-Lite* in the light of first-order logic. In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 361–366, 2007.
2. A. Artale, D. Calvanese, and A. Ibáñez-García. Full satisfiability of UML class diagrams. In *Proc. of the* 29^{th} *International Conference on Conceptual Modeling (ER-10)*, 2010.
3. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
4. A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Complexity of reasoning over temporal data models. In *Proc. of the* 29^{th} *International Conference on Conceptual Modeling (ER-10)*, 2010.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. *DL-Lite*: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in P2P data integration: An epistemic logic approach. *Information Systems*, 33(4):360–384, 2008.
8. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking Data to Ontologies. *J. on Data Semantics*, X:133–173, 2008.

Unification in the Description Logic $\mathcal{EL}$ without the Top Concept

Franz Baader[1]*, Nguyen Thanh Binh[2], Stefan Borgwardt[1]*, and Barbara Morawska[1]*

[1] TU Dresden, Germany, {baader,stefborg,morawska}@tcs.inf.tu-dresden.de
[2] ETH Zürich, Switzerland, thannguy@inf.ethz.ch

Abstract. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. The inexpressive Description Logic $\mathcal{EL}$ is of particular interest in this context since, on the one hand, several large biomedical ontologies are defined using $\mathcal{EL}$. On the other hand, unification in $\mathcal{EL}$ has recently been shown to be NP-complete, and thus of considerably lower complexity than unification in other DLs of similarly restricted expressive power. However, $\mathcal{EL}$ allows the use of the top concept ($\top$), which represents the whole interpretation domain, whereas the large medical ontology SNOMED CT makes no use of this feature. Surprisingly, removing the top concept from $\mathcal{EL}$ makes the unification problem considerably harder. More precisely, we will show that unification in $\mathcal{EL}$ without the top concept is PSpace-complete.

1 Introduction

Unification in DLs has been proposed in [7] as a novel inference service that can, for example, be used to detect redundancies in ontologies. In this paper, we will look at unification in ontologies expressed in $\mathcal{EL}$. For example, assume that one knowledge engineer defines the concept of *female professors* as

$$\mathsf{Person} \sqcap \mathsf{Female} \sqcap \exists \mathsf{job}.\mathsf{Professor},$$

whereas another knowledge engineer represent this notion in a somewhat different way, e.g., by using the concept term

$$\mathsf{Woman} \sqcap \exists \mathsf{job}.(\mathsf{Teacher} \sqcap \mathsf{Researcher}).$$

These two concept terms are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by substituting the concept name Professor in the first term by the concept term Teacher ⊓ Researcher and the concept name Woman in the second term by the concept term Person ⊓ Female. We call a substitution that makes two concept terms equivalent a *unifier* of the two terms. Such a unifier proposes definitions

* Supported by DFG under grant BA 1122/14-1

for the concept names that are used as variables. In our example, we know that, if we define Woman as Person $\sqcap$ Female and Professor as Teacher $\sqcap$ Researcher, then the two concept terms from above are equivalent w.r.t. these definitions.

In [7] it was shown that, for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions, deciding unifiability is an ExpTime-complete problem. In [4], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is "only" NP-complete. The original unification algorithm for $\mathcal{EL}$ introduced in [4] was a brutal "guess and then test" NP-algorithm, but we have since then also developed more practical algorithms. On the one hand, in [6] we describe a goal-oriented unification algorithm for $\mathcal{EL}$, in which non-deterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. On the other hand, in [5], we present an algorithm that is based on a reduction to satisfiability in propositional logic (SAT), and thus allows us to employ highly optimized state-of-the-art SAT solvers for implementing an $\mathcal{EL}$-unification algorithm.

However, the large medical ontology SNOMED CT is not formulated in $\mathcal{EL}$, but rather in its sub-logic $\mathcal{EL}^{-\top}$, which differs from $\mathcal{EL}$ in that the use of the top concept is disallowed. If we employ $\mathcal{EL}$-unification to detect redundancies in (extensions of) SNOMED CT, then a unifier may introduce concept terms that contain the top concept, and thus propose definitions for concept names that are of a form that is not used in SNOMED CT. Apart from this practical motivation for investigating unification in $\mathcal{EL}^{-\top}$, we also found it interesting to see how such a small change in the logic influences the unification problem. Surprisingly, it turned out that the complexity of the problem increases considerably (from NP to PSpace). In addition, compared to $\mathcal{EL}$-unification, quite different methods had to be developed to actually solve $\mathcal{EL}^{-\top}$-unification problems. In particular, we will show in this paper, that—similar to the case of $\mathcal{FL}_0$-unification—$\mathcal{EL}^{-\top}$-unification can be reduced to solving certain language equations. In contrast to the case of $\mathcal{FL}_0$-unification, these language equations can be solved in PSpace rather than ExpTime, which we show by a reduction to the emptiness problem for alternating automata on finite words.

Complete proofs of the results presented in this paper can be found in [1]. There we also show PSpace-hardness of $\mathcal{EL}^{-\top}$-unification by a reduction of the intersection emptiness problem for finite automata [11, 8]. An extended version of this paper will be published as [2].

2 The Description Logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

In this paper, we deal with the description logic $\mathcal{EL}$ in which concept terms are built from concept names (N_C) and role names (N_R) using the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$) and the top concept ($\top$). In the restricted description logic $\mathcal{EL}^{-\top}$, concept terms may not contain $\top$. As usual, these concepts are interpreted as sets over some domain [3].

An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name $A \in N_C$ or an existential restriction $\exists r.D$. Concept names and existential restrictions

$\exists r.D$, where D is a concept name or $\top$, are called *flat atoms*. The set $\mathrm{At}(C)$ of *atoms of an $\mathcal{EL}$-concept term C* consists of all the subterms of C that are atoms. For example, $C = A \sqcap \exists r.(B \sqcap \exists r.\top)$ has the atom set $\mathrm{At}(C) = \{A, \exists r.(B \sqcap \exists r.\top), B, \exists r.\top\}$. Obviously, any $\mathcal{EL}$-concept term C is a conjunction $C = C_1 \sqcap \ldots \sqcap C_n$ of atoms and $\top$. We call the atoms among $C_1, \ldots, C_n$ the *top-level atoms* of C. The $\mathcal{EL}$-concept term C is called *flat* if all its top-level atoms are flat. Subsumption in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ can be characterized as follows [6]:

Lemma 1. *Let $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_l \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$ be two $\mathcal{EL}$-concept terms, where $A_1, \ldots, A_k, B_1, \ldots, B_l$ are concept names. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_l\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j \in \{1, \ldots, n\}$ there exists an $i \in \{1, \ldots, m\}$ such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

In particular, this means that $C \sqsubseteq D$ iff for every top-level atom D' of D there is a top-level atom C' of C such that $C' \sqsubseteq D'$.

Modulo equivalence, the subsumption relation is a partial order on concept terms. In $\mathcal{EL}$, the top concept $\top$ is the greatest element w.r.t. this order. In $\mathcal{EL}^{-\top}$, there are many incomparable maximal concept terms. We will see below that these are exactly the $\mathcal{EL}^{-\top}$-concept terms of the form $\exists r_1.\cdots\exists r_n.A$ for $n \geq 0$ role names $r_1, \ldots, r_n$ and a concept name A. We call such concept terms *particles*. The set $\mathrm{Part}(C)$ of all particles of a given $\mathcal{EL}^{-\top}$-concept term C is defined as

- $\mathrm{Part}(C) := \{C\}$ if C is a concept name,
- $\mathrm{Part}(C) := \{\exists r.E \mid E \in \mathrm{Part}(D)\}$ if $C = \exists r.D$,
- $\mathrm{Part}(C) := \mathrm{Part}(C_1) \cup \mathrm{Part}(C_2)$ if $C = C_1 \sqcap C_2$.

For example, the particles of $C = A \sqcap \exists r.(A \sqcap \exists r.B)$ are $A, \exists r.A, \exists r.\exists r.B$. Such particles will play an important role in our $\mathcal{EL}^{-\top}$-unification algorithm. The next lemma states that particles are indeed the maximal concept terms w.r.t. to subsumption in $\mathcal{EL}^{-\top}$, and that the particles subsuming an $\mathcal{EL}^{-\top}$-concept term C are exactly the particles of C.

Lemma 2. *Let C be an $\mathcal{EL}^{-\top}$-concept term and B a particle.*

1. *If $B \sqsubseteq C$, then $B \equiv C$.*
2. *$B \in \mathrm{Part}(C)$ iff $C \sqsubseteq B$.*

3 Unification in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

To define unification in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ simultaneously, let $\mathcal{L} \in \{\mathcal{EL}, \mathcal{EL}^{-\top}\}$. When defining unification in $\mathcal{L}$, we assume that the set of concepts names is partitioned into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). An *$\mathcal{L}$-substitution σ* is a mapping from N_v into the set of all $\mathcal{L}$-concept terms. This mapping is extended to concept terms in the usual way, i.e., by replacing all occurrences of variables in the term by their σ-images. An

$\mathcal{L}$-concept term is called *ground* if it contains no variables, and an $\mathcal{L}$-substitution σ is called *ground* if the concept terms $\sigma(X)$ are ground for all $X \in N_v$.

Unification tries to make concept terms equivalent by applying a substitution.

Definition 1. *An* $\mathcal{L}$-unification problem *is of the form* $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$*, where* $C_1, D_1, \ldots C_n, D_n$ *are* $\mathcal{L}$*-concept terms. The* $\mathcal{L}$*-substitution* σ *is an* $\mathcal{L}$-unifier *of* Γ *iff it solves all the equations* $C_i \equiv^? D_i$ *in* Γ*, i.e., iff* $\sigma(C_i) \equiv \sigma(D_i)$ *for* $i = 1, \ldots, n$*. In this case,* Γ *is called* $\mathcal{L}$-unifiable.

In the following, we will use the subsumption $C \sqsubseteq^? D$ as an abbreviation for the equation $C \sqcap D \equiv^? C$. Obviously, σ solves this equation iff $\sigma(C) \sqsubseteq \sigma(D)$.

Clearly, every $\mathcal{EL}^{-\top}$-unification problem Γ is also an $\mathcal{EL}$-unification problem. Whether Γ is $\mathcal{L}$-unifiable or not may depend, however, on whether $\mathcal{L} = \mathcal{EL}$ or $\mathcal{L} = \mathcal{EL}^{-\top}$. As an example, consider the problem $\Gamma := \{A \sqsubseteq^? X, B \sqsubseteq^? X\}$, where A, B are distinct concept constants and X is a concept variable. Obviously, the substitution that replaces X by $\top$ is an $\mathcal{EL}$-unifier of Γ. However, Γ does not have an $\mathcal{EL}^{-\top}$-unifier. In fact, for such a unifier σ, the $\mathcal{EL}^{-\top}$-concept term $\sigma(X)$ would need to satisfy $A \sqsubseteq \sigma(X)$ and $B \sqsubseteq \sigma(X)$. Since A and B are particles, Lemma 2 would imply $A \equiv \sigma(X) \equiv B$ and thus $A \equiv B$, which is not the case.

It is easy to see that, for both $\mathcal{L} = \mathcal{EL}$ and $\mathcal{L} = \mathcal{EL}^{-\top}$, an $\mathcal{L}$-unification problem Γ has an $\mathcal{L}$-unifier iff it has a ground $\mathcal{L}$-unifier σ that uses only concept and role names occurring in Γ,[3] i.e., for all variables X, the $\mathcal{L}$-concept term $\sigma(X)$ is a ground term that contains only such concept and role names. In addition, we may without loss of generality restrict our attention to *flat* $\mathcal{L}$*-unification problems*, i.e., unification problems in which the left- and right-hand sides of equations are flat $\mathcal{L}$-concept terms (see, e.g., [6]).

Given a flat $\mathcal{L}$-unification problem Γ, we denote by $\mathrm{At}(\Gamma)$ the set of all atoms of Γ, i.e., the union of all sets of atoms of the concept terms occurring in Γ. By $\mathrm{Var}(\Gamma)$ we denote the variables that occur in Γ, and by $\mathrm{NV}(\Gamma) := \mathrm{At}(\Gamma) \backslash \mathrm{Var}(\Gamma)$ the set of all *non-variable atoms* of Γ.

$\mathcal{EL}$-unification by guessing acyclic assignments

The NP-algorithm for $\mathcal{EL}$-unification introduced in [4] guesses, for every variable X occurring in Γ, a set $S(X)$ of non-variable atoms of Γ. Given such an *assignment* of sets of non-variable atoms to the variables in Γ, we say that the variable X *directly depends on* the variable Y if Y occurs in an atom of $S(X)$. Let *depends on* be the transitive closure of *directly depends on*. If there is no variable that depends on itself, then we call this assignment *acyclic*. In case the guessed assignment is not acyclic, this run of the NP-algorithm returns "fail." Otherwise, there exists a strict linear order $>$ on the variables occurring in Γ such that $X > Y$ if X depends on Y. One can then define the substitution γ^S induced by the assignment S along this linear order:

– If X is the least variable w.r.t. $>$, then $\gamma^S(X)$ is the conjunction of the elements of $S(X)$, where the empty conjunction is $\top$.

[3] Without loss of generality, we assume that Γ contains at least one concept name.

– Assume $\gamma^S(Y)$ is defined for all variables $Y < X$. If $S(X) = \{D_1, \ldots, D_n\}$, then $\gamma^S(X) := \gamma^S(D_1) \sqcap \ldots \sqcap \gamma^S(D_n)$.

The algorithm then tests whether the substitution γ^S computed this way is a unifier of Γ. If this is the case, then this run returns γ^S; otherwise, it returns "fail." In [4] it is shown that Γ is unifiable iff there is a run of this algorithm on input Γ that returns a substitution (which is then an $\mathcal{EL}$-unifier of Γ).

Why this does not work for $\mathcal{EL}^{-\top}$

The $\mathcal{EL}$-unifiers returned by the $\mathcal{EL}$-unification algorithm sketched above need not be $\mathcal{EL}^{-\top}$-unifiers since some of the sets $S(X)$ in the guessed assignment may be empty, in which case $\gamma^S(X) = \top$. This suggests the following simple modification of the above algorithm: require that the guessed assignment is such that all sets $S(X)$ are nonempty. If such an assignment S is acyclic, then the induced substitution γ^S is actually an $\mathcal{EL}^{-\top}$-substitution, and thus the substitutions returned by the modified algorithm are indeed $\mathcal{EL}^{-\top}$-unifiers. However, this modified algorithm does not always detect $\mathcal{EL}^{-\top}$-unifiability, i.e., it may return no substitution although the input problem is $\mathcal{EL}^{-\top}$-unifiable.

As an example, consider the $\mathcal{EL}^{-\top}$-unification problem

$$\Gamma := \{A \sqcap B \equiv^? Y,\ B \sqcap C \equiv^? Z,\ \exists r.Y \sqsubseteq^? X,\ \exists r.Z \sqsubseteq^? X\},$$

where X, Y, Z are concept variables and A, B, C are distinct concept constants. We claim that, up to equivalence, the substitution that maps X to $\exists r.B$, Y to $A \sqcap B$, and Z to $B \sqcap C$ is the only $\mathcal{EL}^{-\top}$-unifier of Γ. In fact, any $\mathcal{EL}^{-\top}$-unifier γ of Γ must map Y to $A \sqcap B$ and Z to $B \sqcap C$, and thus satisfy $\exists r.(A \sqcap B) \sqsubseteq \gamma(X)$ and $\exists r.(B \sqcap C) \sqsubseteq \gamma(X)$. Lemma 1 then yields that the only possible top-level atom of $\gamma(X)$ is $\exists r.B$. However, there is no non-variable atom $D \in \mathrm{NV}(\Gamma)$ such that $\gamma(D)$ is equivalent to $\exists r.B$. This shows that Γ has an $\mathcal{EL}^{-\top}$-unifier, but this unifier cannot be computed by the modified algorithm sketched above.

The main idea underlying the $\mathcal{EL}^{-\top}$-unification algorithm introduced in the next section is that one starts with an $\mathcal{EL}$-unifier, and then conjoins "appropriate" particles to the images of the variables that are replaced by $\top$ by this unifier. It is, however, not so easy to decide which particles can be added this way without turning the $\mathcal{EL}$-unifier into an $\mathcal{EL}^{-\top}$-substitution that no longer solves the unification problem.

4 An $\mathcal{EL}^{-\top}$-Unification Algorithm

In the following, let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem. Without loss of generality we assume that Γ consists of subsumptions of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ for atoms $C_1, \ldots, C_n, D$. Our decision procedure for $\mathcal{EL}^{-\top}$-unifiability proceeds in four steps.

Step 1. If S is an acyclic assignment guessed by the $\mathcal{EL}$-unification algorithm sketched above, then $D \in S(X)$ implies that the subsumption $\gamma^S(X) \sqsubseteq \gamma^S(D)$

holds for the substitution γ^S induced by S. Instead of guessing just subsumptions between variables and non-variable atoms, our $\mathcal{EL}^{-\top}$-unification algorithm starts with guessing subsumptions between arbitrary atoms of Γ. To be more precise, it guesses a mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0,1\}$, which specifies which subsumptions between atoms of Γ should hold for the $\mathcal{EL}^{-\top}$-unifier that it tries to generate: if $\tau(D_1, D_2) = 1$ for $D_1, D_2 \in \mathrm{At}(\Gamma)$, then this means that the search for a unifier is restricted (in this branch of the search tree) to substitutions γ satisfying $\gamma(D_1) \sqsubseteq \gamma(D_2)$. Obviously, any such mapping τ also yields an assignment

$$S^\tau(X) := \{D \in \mathrm{NV}(\Gamma) \mid \tau(X, D) = 1\},$$

and we require that this assignment is acyclic and induces an $\mathcal{EL}$-unifier of Γ.

Definition 2. *The mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0,1\}$ is called a* subsumption mapping for Γ *if it satisfies the following three conditions:*

1. *It respects the properties of subsumption in $\mathcal{EL}$:*
   (a) *$\tau(D, D) = 1$ for each $D \in \mathrm{At}(\Gamma)$.*
   (b) *$\tau(A_1, A_2) = 0$ for distinct concept constants $A_1, A_2 \in \mathrm{At}(\Gamma)$.*
   (c) *$\tau(\exists r.C_1, \exists s.C_2) = 0$ for distinct $r, s \in N_R$ with $\exists r.C_1, \exists s.C_2 \in \mathrm{At}(\Gamma)$.*
   (d) *$\tau(A, \exists r.C) = \tau(\exists r.C, A) = 0$ for each constant $A \in \mathrm{At}(\Gamma)$, role name r and variable or constant C with $\exists r.C \in \mathrm{At}(\Gamma)$.*
   (e) *If $\exists r.C_1, \exists r.C_2 \in \mathrm{At}(\Gamma)$, then $\tau(\exists r.C_1, \exists r.C_2) = \tau(C_1, C_2)$.*
   (f) *For all atoms $D_1, D_2, D_3 \in \mathrm{At}(\Gamma)$, if $\tau(D_1, D_2) = \tau(D_2, D_3) = 1$, then $\tau(D_1, D_3) = 1$.*
2. *It induces an $\mathcal{EL}$-substitution, i.e., the assignment S^τ is acyclic and thus induces a substitution γ^{S^τ}, which we will simply denote by γ^τ.*
3. *It respects the subsumptions of Γ, i.e., it satisfies the following conditions for each subsumption $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ in Γ:*
   (a) *If D is a non-variable atom, then there is at least one C_i such that $\tau(C_i, D) = 1$.*
   (b) *If D is a variable and $\tau(D, C) = 1$ for a non-variable atom $C \in \mathrm{NV}(\Gamma)$, then there is at least one C_i with $\tau(C_i, C) = 1$.*

Though this is not really necessary for the proof of correctness of our $\mathcal{EL}^{-\top}$-unification algorithm, it can be shown that the substitution γ^τ induced by a subsumption mapping τ for Γ is indeed an $\mathcal{EL}$-unifier of Γ. It should be noted that γ^τ need not be an $\mathcal{EL}^{-\top}$-unifier of Γ. In addition, γ^τ need not agree with τ on every subsumption between atoms of Γ. The reason for this is that τ specifies subsumptions which should hold in the $\mathcal{EL}^{-\top}$-unifier of Γ to be constructed. To turn γ^τ into such an $\mathcal{EL}^{-\top}$-unifier, we may have to add certain particles, and these additions may invalidate subsumptions that hold for γ^τ. However, we will ensure that no subsumption claimed by τ is invalidated.

Step 2. In this step, we use τ to turn Γ into a unification problem that has only variables on the right-hand sides of subsumptions. More precisely, we define $\Delta_{\Gamma,\tau} := \Delta_\Gamma \cup \Delta_\tau$, where

$$\Delta_\Gamma := \{C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? X \in \Gamma \mid X \text{ is a variable of } \Gamma\},$$
$$\Delta_\tau := \{C \sqsubseteq^? X \mid X \text{ is a variable and } C \text{ an atom of } \Gamma \text{ with } \tau(C, X) = 1\}.$$

For an arbitrary $\mathcal{EL}^{-\top}$-substitution σ, we define

$$S^\sigma(X) := \{D \in \mathrm{NV}(\varGamma) \mid \sigma(X) \sqsubseteq \sigma(D)\},$$

and write $S^\tau \leq S^\sigma$ if $S^\tau(X) \subseteq S^\sigma(X)$ for every variable X. The following lemma states the connection between $\mathcal{EL}^{-\top}$-unifiability of $\varGamma$ and of $\varDelta_{\varGamma,\tau}$, using the notation that we have just introduced.

Lemma 3. *Let $\varGamma$ be a flat $\mathcal{EL}^{-\top}$-unification problem. Then the following statements are equivalent for any $\mathcal{EL}^{-\top}$-substitution σ:*

1. *σ is an $\mathcal{EL}^{-\top}$-unifier of $\varGamma$.*
2. *There is a subsumption mapping $\tau : \mathrm{At}(\varGamma)^2 \to \{0,1\}$ for $\varGamma$ such that σ is an $\mathcal{EL}^{-\top}$-unifier of $\varDelta_{\varGamma,\tau}$ and $S^\tau \leq S^\sigma$.*

Step 3. In this step, we characterize which particles can be added in order to turn γ^τ into an $\mathcal{EL}^{-\top}$-unifier σ of $\varDelta_{\varGamma,\tau}$ satisfying $S^\tau \leq S^\sigma$. Recall that particles are of the form $\exists r_1.\cdots\exists r_n.A$ for $n \geq 0$ role names $r_1,\ldots,r_n$ and a concept name A. We write such a particle as $\exists w.A$, where $w = r_1\cdots r_n$ is viewed as a word over the alphabet N_R of all role names. If $n = 0$, then w is the empty word ε and $\exists\varepsilon.A$ is just A.

Admissible particles are determined by solutions of a system of linear language inclusions. These *linear inclusions* are of the form

$$X_i \subseteq L_0 \cup L_1X_1 \cup \ldots \cup L_nX_n, \tag{1}$$

where $X_1,\ldots,X_n$ are indeterminates, $i \in \{1,\ldots,n\}$, and each L_i ($i \in \{0,\ldots,n\}$) is a subset of $N_R \cup \{\varepsilon\}$. A *solution* θ of such an inclusion assigns sets of words $\theta(X_i) \subseteq N_R^*$ to the indeterminates X_i such that $\theta(X_i) \subseteq L_0 \cup L_1\theta(X_1) \cup \ldots \cup L_n\theta(X_n)$.

The unification problem $\varDelta_{\varGamma,\tau}$ induces a finite system $\mathcal{I}_{\varGamma,\tau}$ of such inclusions. The indeterminates of $\mathcal{I}_{\varGamma,\tau}$ are of the form X_A, where $X \in N_v$ and $A \in N_c$. For each constant $A \in N_c$ and each subsumption of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? X \in \varDelta_{\varGamma,\tau}$, we add the following inclusion to $\mathcal{I}_{\varGamma,\tau}$:

$$X_A \subseteq f_A(C_1) \cup \ldots \cup f_A(C_n), \text{ where}$$
$$f_A(C) := \begin{cases} \{r\}f_A(C') & \text{if } C = \exists r.C' \\ Y_A & \text{if } C = Y \text{ is a variable} \\ \{\varepsilon\} & \text{if } C = A \\ \emptyset & \text{if } C \in N_c \setminus \{A\} \end{cases}$$

Since $\varDelta_{\varGamma,\tau}$ contains only flat atoms, these inclusion are indeed of the form (1).

We call a solution θ of $\mathcal{I}_{\varGamma,\tau}$ *admissible* if, for every variable $X \in N_v$, there is a constant $A \in N_c$ such that $\theta(X_A)$ is nonempty. This condition will ensure that we can add enough particles to turn γ^τ into an $\mathcal{EL}^{-\top}$-substitution. In order to obtain a substitution at all, only finitely many particles can be added. Thus, we are interested in *finite* solutions of $\mathcal{I}_{\varGamma,\tau}$, i.e., solutions θ such that all the sets $\theta(X_A)$ are finite.

Lemma 4. *Let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem and τ a subsumption mapping for Γ. Then $\Delta_{\Gamma,\tau}$ has an $\mathcal{EL}^{-\top}$-unifier σ with $S^\tau \leq S^\sigma$ iff $\mathcal{I}_{\Gamma,\tau}$ has a finite, admissible solution.*

Proof sketch. Given a ground $\mathcal{EL}^{-\top}$-unifier σ of $\Delta_{\Gamma,\tau}$ with $S^\tau \leq S^\sigma$, we define for each concept variable X and concept constant A occurring in Γ:

$$\theta(X_A) := \{w \in N_R^* \mid \exists w.A \in \mathrm{Part}(\sigma(X))\}.$$

It can then be shown that θ is a solution of $\mathcal{I}_{\Gamma,\tau}$. This solution is finite since any concept term has only finitely many particles, and it is admissible since σ is an $\mathcal{EL}^{-\top}$-substitution.

Conversely, let θ be a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$. We define the substitution σ by induction on the dependency order $>$ induced by S^τ as follows. Let X be a variable of Γ and assume that $\sigma(Y)$ has already been defined for all variables Y with $X > Y$. Then we set

$$\sigma(X) := \bigsqcap_{D \in S^\tau(X)} \sigma(D) \sqcap \bigsqcap_{A \in N_c} \bigsqcap_{w \in \theta(X_A)} \exists w.A.$$

Since θ is finite and admissible, σ is a well-defined $\mathcal{EL}^{-\top}$-substitution. It can be shown that $\sigma(X)$ is indeed an $\mathcal{EL}^{-\top}$-unifier of $\Delta_{\Gamma,\tau}$ with $S^\tau \leq S^\sigma$. $\square$

Step 4. In this step we show how to test whether the system $\mathcal{I}_{\Gamma,\tau}$ of linear language inclusions constructed in the previous step has a finite, admissible solution or not. The main idea is to consider the greatest solution of $\mathcal{I}_{\Gamma,\tau}$.

To be more precise, given a system of linear language inclusions $\mathcal{I}$, we can order the solutions of $\mathcal{I}$ by defining $\theta_1 \subseteq \theta_2$ iff $\theta_1(X) \subseteq \theta_2(X)$ for all indeterminates X of $\mathcal{I}$. Since $\theta_\emptyset$, which assigns the empty set to each indeterminate of $\mathcal{I}$, is a solution of $\mathcal{I}$ and solutions are closed under argument-wise union, the following clearly defines the (unique) greatest solution θ^* of $\mathcal{I}$ w.r.t. this order:

$$\theta^*(X) := \bigcup_{\theta \text{ solution of } \mathcal{I}} \theta(X).$$

Lemma 5. *Let X be an indeterminate in $\mathcal{I}$ and θ^* the maximal solution of $\mathcal{I}$. If $\theta^*(X)$ is nonempty, then there is a finite solution θ of $\mathcal{I}$ such that $\theta(X)$ is nonempty.*

Proof. Let $w \in \theta^*(X)$. We construct the finite solution θ of $\mathcal{I}$ by keeping only the words of length $|w|$: for all indeterminates Y occurring in $\mathcal{I}$ we define

$$\theta(Y) := \{u \in \theta^*(Y) \mid |u| \leq |w|\}.$$

By definition, we have $w \in \theta(X)$. To show that θ is indeed a solution of $\mathcal{I}$, consider an arbitrary inclusion $Y \subseteq L_0 \cup L_1 X_1 \cup \ldots \cup L_n X_n$ in $\mathcal{I}$, and assume that $u \in \theta(Y)$. We must show that $u \in L_0 \cup L_1\theta(X_1) \cup \ldots \cup L_n\theta(X_n)$. Since $u \in \theta^*(Y)$ and θ^* is a solution of $\mathcal{I}$, we have (i) $u \in L_0$ or (ii) $u \in L_i\theta^*(X_i)$ for some $i, 1 \leq i \leq n$. In the first case, we are done. In the second case, $u = \alpha u'$ for some $\alpha \in L_i \subseteq N_R \cup \{\varepsilon\}$ and $u' \in \theta^*(X_i)$. Since $|u'| \leq |u| \leq |w|$, we have $u' \in \theta(X_i)$, and thus $u \in L_i\theta(X_i)$. $\square$

Lemma 6. *There is a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$ iff the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$ is admissible.*

Proof. If $\mathcal{I}_{\Gamma,\tau}$ has a finite, admissible solution θ, then the maximal solution of $\mathcal{I}_{\Gamma,\tau}$ contains this solution, and is thus also admissible.

Conversely, if θ^* is admissible, then (by Lemma 5) for each $X \in \mathrm{Var}(\Gamma)$ there is a constant $A(X)$ and a finite solution θ_X of $\mathcal{I}_{\Gamma,\tau}$ such that $\theta_X(X_{A(X)}) \neq \emptyset$. The union of these solutions θ_X for $X \in \mathrm{Var}(\Gamma)$ is the desired finite, admissible solution. □

Given this lemma, it remains to show how we can test admissibility of the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$. For this purpose, it is obviously sufficient to be able to test, for each indeterminate X_A in $\mathcal{I}_{\Gamma,\tau}$, whether $\theta^*(X_A)$ is empty or not. This can be achieved by representing the languages $\theta^*(X_A)$ using *alternating finite automata with ε-transitions (ε-AFA)*, which are a special case of two-way alternating finite automata. In fact, as shown in [10], the emptiness problem for two-way alternating finite automata (and thus also for ε-AFA) is in PSpace.

Lemma 7. *For each indeterminate X_A in $\mathcal{I}_{\Gamma,\tau}$, we can construct in polynomial time in the size of $\mathcal{I}_{\Gamma,\tau}$ an ε-AFA $\mathcal{A}(X,A)$ such that the language $L(\mathcal{A}(X,A))$ accepted by $\mathcal{A}(X,A)$ is equal to $\theta^*(X_A)$, where θ^* denotes the maximal solution of $\mathcal{I}_{\Gamma,\tau}$.*

This finishes the description of our $\mathcal{EL}^{-\top}$-unification algorithm. It remains to argue why it is a PSpace decision procedure for $\mathcal{EL}^{-\top}$-unifiability.

Theorem 1. *The problem of deciding unifiability in $\mathcal{EL}^{-\top}$ is* PSpace*-complete.*

Proof. Here we only show that the problem is in NPSpace, which is equal to PSpace by Savitch's theorem [13]. PSpace-hardness is shown in [1, 2].

Let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem. By Lemma 3, Lemma 4, and Lemma 6, we know that Γ is $\mathcal{EL}^{-\top}$-unifiable iff there is a subsumption mapping τ for Γ such that the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$ is admissible.

Thus, we first guess a mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0,1\}$ and test whether τ is a subsumption mapping for Γ. Guessing τ can clearly be done in NPSpace. For a given mapping τ, the test whether it is a subsumption mapping for Γ can be done in polynomial time.

From τ we can first construct $\Delta_{\Gamma,\tau}$ and then $\mathcal{I}_{\Gamma,\tau}$ in polynomial time. Given $\mathcal{I}_{\Gamma,\tau}$, we then construct the (polynomially many) ε-AFA $\mathcal{A}(X,A)$, and test them for emptiness. Since the emptiness problem for ε-AFA is in PSpace, this can be achieved within PSpace. Given the results of these emptiness tests, we can then check in polynomial time whether, for each concept variable X of Γ there is a concept constant A of Γ such that $\theta^*(X_A) = L(\mathcal{A}(X,A)) \neq \emptyset$. If this is the case, then θ^* is admissible, and thus Γ is $\mathcal{EL}^{-\top}$-unifiable. □

5 Conclusion

Unification in $\mathcal{EL}$ was introduced in [4] as an inference service that can support the detection of redundancies in large biomedical ontologies, which are frequently written in this DL. Motivated by the fact that the large medical ontology SNOMED CT actually does not use the top concept available in $\mathcal{EL}$, we have in this paper investigated unification in $\mathcal{EL}^{-\top}$, which is obtained from $\mathcal{EL}$ by removing the top concept. More precisely, SNOMED CT is a so-called acyclic $\mathcal{EL}^{-\top}$-TBox,[4] rather than a collection of $\mathcal{EL}^{-\top}$-concept terms. However, as shown in [6], acyclic TBoxes can be easily handled by a unification algorithm for concept terms.

Surprisingly, it has turned out that the complexity of unification in $\mathcal{EL}^{-\top}$ (PSpace) is considerably higher than of unification in $\mathcal{EL}$ (NP). From a theoretical point of view, this result is interesting since it provides us with a natural example where reducing the expressiveness of a given DL (in a rather minor way) increases the complexity of the unifiability problem. Regarding the complexity of unification in more expressive DLs, not much is known. If we add negation to $\mathcal{EL}$, then we obtain the well-known DL $\mathcal{ALC}$, which corresponds to the basic (multi-)modal logic K [14]. Decidability of unification in K is a long-standing open problem. Recently, undecidability of unification in some extensions of K (for example, by the universal modality) was shown in [17]. These undecidability results also imply undecidability of unification in some expressive DLs (e.g., in $\mathcal{SHIQ}$ [9]).

Apart from its theoretical interest, the result of this paper also has practical implications. Whereas practically rather efficient unification algorithm for $\mathcal{EL}$ can readily be obtained by a translation into SAT [5], it is not so clear how to turn the PSpace algorithm for $\mathcal{EL}^{-\top}$-unification introduced in this paper into a practically useful algorithm. One possibility could be to use a SAT modulo theories (SMT) approach [12]. The idea is that the SAT solver is used to generate all possible subsumption mappings for Γ, and that the theory solver tests the system $\mathcal{I}_{\Gamma,\tau}$ induced by τ for the existence of a finite, admissible solution. How well this works will mainly depend on whether we can develop such a theory solver that satisfies well all the requirements imposed by the SMT approach.

Another topic for future research is how to actually compute $\mathcal{EL}^{-\top}$-unifiers for a unifiable $\mathcal{EL}^{-\top}$-unification problem. In principle, our decision procedure is constructive in the sense that, from appropriate successful runs of the ε-AFA $\mathcal{A}(X, A)$, one can construct a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$, and from this an $\mathcal{EL}^{-\top}$-unifier of Γ. However, this needs to be made more explicit, and we need to investigate what kind of $\mathcal{EL}^{-\top}$-unifiers can be computed this way.

[4] Note that the right-identity rules in SNOMED CT [15] are actually not expressed using complex role inclusion axioms, but through the SEP-triplet encoding [16]. Thus, complex role inclusion axioms are not relevant here.

References

1. Franz Baader, Nguyen Thanh Binh, Stefan Borgwardt, and Barbara Morawska. Unification in the description logic $\mathcal{EL}$ without the top concept. LTCS-Report 11-01, TU Dresden, Dresden, Germany, 2011.
   See http://lat.inf.tu-dresden.de/research/reports.html.
2. Franz Baader, Nguyen Thanh Binh, Stefan Borgwardt, and Barbara Morawska. Unification in the description logic $\mathcal{EL}$ without the top concept. In *Proc. of the 23rd Int. Conf. on Automated Deduction (CADE 23)*, Springer LNCS, 2011. To appear.
3. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. In *Proc. of the 20th Int. Conf. on Rewriting Techniques and Applications (RTA 2009)*, Springer LNCS 5595, pages 350–364, 2009.
5. Franz Baader and Barbara Morawska. SAT encoding of unification in $\mathcal{EL}$. In *Proc. of the 17th Int. Conf. on Logic for Programming, Artifical Intelligence, and Reasoning (LPAR-17)*, Springer LNCS 6397, pages 97–111, 2010.
6. Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010.
7. Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *Journal of Symbolic Computation*, 31(3):277–305, 2001.
8. Michael R. Garey and David S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (USA), 1979.
9. Ian Horrocks, Ulrike Sattler, and Stefan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.
10. Tao Jiang and B. Ravikumar. A note on the space complexity of some decision problems for finite automata. *Information Processing Letters*, 40:25–31, 1991.
11. Dexter Kozen. Lower bounds for natural proof systems. *Annual IEEE Symposium on Foundations of Computer Science*, pages 254–266, 1977.
12. Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.
13. Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.
14. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.
15. Kent A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue.
16. Boontawee Suntisrivaraporn, Franz Baader, Stefan Schulz, and Kent Spackman. Replacing SEP-triplets in SNOMED CT using tractable description logic operators. In *Proc. of the 11th Conf. on Artificial Intelligence in Medicine (AIME'07)*, Springer LNCS 4594, pages 287–291, 2007.
17. Frank Wolter and Michael Zakharyaschev. Undecidability of the unification and admissibility problems for modal and description logics. *ACM Transactions on Computational Logic*, 9(4), 2008.

GCIs Make Reasoning in Fuzzy DL with the Product T-norm Undecidable

Franz Baader and Rafael Peñaloza

Theoretical Computer Science, TU Dresden, Germany
{baader,penaloza}@tcs.inf.tu-dresden.de

1 Introduction

Fuzzy variants of Description Logics (DLs) were introduced in order to deal with applications where not all concepts can be defined in a precise way. A great variety of fuzzy DLs have been investigated in the literature [12,8]. In fact, compared to crisp DLs, fuzzy DLs offer an additional degree of freedom when defining their expressiveness: in addition to deciding which concept constructors (like conjunction, disjunction, existential restriction) and which TBox formalism (like no TBox, acyclic TBox, general concept inclusions) to use, one must also decide how to interpret the concept constructors by appropriate functions on the domain of fuzzy values $[0, 1]$. For example, conjunction can be interpreted by different t-norms (such as Gödel, Łukasiewicz, and product) and there are also different options for how to interpret negation (such as involutive negation and residual negation). In addition, one can either consider all models or only so-called witnessed models [10] when defining the semantics of fuzzy DLs.

Decidability of fuzzy DLs is often shown by adapting the tableau-based algorithms for the corresponding crisp DL to the fuzzy case. This was first done for the case of DLs without general concept inclusion axioms (GCIs) [19,17,14,6], but then also extended to GCIs [16,15,18,4,5]. Usually, these tableau algorithms reason w.r.t. witnessed models.[1] It should be noted, however, that in the presence of GCIs there are different ways of extending the notion of witnessed models from [10], depending on whether the witnessed property is required to apply also to GCIs (in which case we talk about strongly witnessed models) or not (in which case we talk about witnessed models).

The paper [4] considers the case of reasoning w.r.t. fuzzy GCIs in the setting of a logic with product t-norm and involutive negation. More precisely, the tableau algorithm introduced in that paper is supposed to check whether an ontology consisting of fuzzy GCIs and fuzzy ABox assertions expressed in this DL has a strongly witnessed model or not.[2] Actually, the proof of correctness of this algorithm given in [4] implies that, whenever such an ontology has a strongly witnessed model, then it has a finite model. However, it was recently shown in [2]

[1] In fact, witnessed models were introduced in [10] to correct the proof of correctness for the tableau algorithm presented in [19].

[2] Note that the authors of [4] actually use the term "witnessed models" for what we call "strongly witnessed models."

that this is not the case in the presence of general concept inclusion axioms, i.e., there is an ontology written in this logic that has a strongly witnessed model, but does not have a finite model. Of course, this does not automatically imply that the algorithm itself is wrong. In fact, if one applies the algorithm from [4] to the ontology used in [2] to demonstrate the failure of the finite model property, then one obtains the correct answer, and in [2] the authors actually conjecture that the algorithm is still correct. However, incorrectness of the algorithm has now independently been shown in [3] and in [1]. Thus, one can ask whether the fuzzy DL considered in [4] is actually decidable. Though this question is not answered in [1], the paper gives strong indications that the answer might in fact be "no." More precisely, [1] contains a proof of undecidability for a variant of the fuzzy DL considered in [4], which (i) additionally allows for strict GCIs, i.e., GCIs whose fuzzy value is required to be *strictly greater* than a given rational number; and (ii) where the notion of strongly witnessed models used in [4] is replaced by the weaker notion of witnessed models.

In this paper, we consider a different fuzzy DL with product t-norm, where disjunction and involutive negation are replaced by the constructor implication, which is interpreted as the residuum. In this logic, residual negation can be expressed, but neither involutive negation nor disjunction. It was introduced in [10], where decidability of reasoning w.r.t. witnessed models was shown for the case without GCIs. In [7], an analogous decidability result was shown for the case of reasoning w.r.t. so-called quasi-witnessed models. Following [7], we call this logic $*\text{-}\mathcal{ALE}$. In the present paper we show that adding GCIs makes reasoning in $*\text{-}\mathcal{ALE}$ undecidable w.r.t. several variants of the notion of witnessed models (including witnessed, quasi-witnessed, and strongly witnessed models).

2 Preliminaries

In this section, we introduce the logic $*\text{-}\mathcal{ALE}$ and some of the properties that will be useful throughout the paper.

The syntax of this logic is slightly different from standard description logics, as it allows for an implication constructor, and no negation or disjunction. $*\text{-}\mathcal{ALE}$ *concepts* are built through the syntactic rule

$$C ::= A \mid \bot \mid \top \mid C_1 \sqcap C_2 \mid C_1 \to C_2 \mid \exists r.C \mid \forall r.C$$

where A is a *concept name* and r is a *role name.*

A $*\text{-}\mathcal{ALE}$ *ABox* is a finite set of *assertion axioms* of the form $\langle a : C \rhd q\rangle$ or $\langle (a, b) : r \rhd q\rangle$, where C is a $*\text{-}\mathcal{ALE}$ concept, $r \in \mathcal{N}_\mathcal{R}$, q is a rational number in $[0, 1]$, a, b are *individual names* and $\rhd$ is either $\geq$ or $=$. A $*\text{-}\mathcal{ALE}$ *TBox* is a finite set of *concept inclusion axioms* of the form $\langle C \sqsubseteq D \geq q\rangle$, where C, D are $*\text{-}\mathcal{ALE}$ concepts and q is a rational number in $[0, 1]$. A $*\text{-}\mathcal{ALE}$ *ontology* is a tuple $(\mathcal{A}, \mathcal{T})$, where $\mathcal{A}$ is a $*\text{-}\mathcal{ALE}$ ABox and $\mathcal{T}$ a $*\text{-}\mathcal{ALE}$ TBox. In the following we will often drop the prefix $*\text{-}\mathcal{ALE}$, and speak simply of e.g. *TBoxes* and *ontologies.*

The semantics of this logic extends the classical DL semantics by interpreting concepts and roles as fuzzy sets over an interpretation domain. Given a non-empty domain Δ, a *fuzzy set* is a function $F : \Delta \to [0, 1]$, with the intuition that an element $\delta \in \Delta$ belongs to F with *degree* $F(\delta)$. Here, we focus on the *product t-norm* semantics, where logical constructors are interpreted using the product t-norm $\otimes$ and its *residuum* $\Rightarrow$ defined, for every $\alpha,\beta \in [0,1]$, as follows:

$$\alpha \otimes \beta := \alpha \cdot \beta,$$
$$\alpha \Rightarrow \beta := \begin{cases} 1 & \text{if } \alpha \leq \beta \\ \beta/\alpha & \text{otherwise.} \end{cases}$$

The semantics of $*$-$\mathcal{ALE}$ is based on interpretations. An *interpretation* is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty set, called the *domain*, and the function $\cdot^\mathcal{I}$ maps each individual name a to an element of $\Delta^\mathcal{I}$, each concept name A to a function $A^\mathcal{I} : \Delta^\mathcal{I} \to [0,1]$ and each role name r to a function $r^\mathcal{I} : \Delta^\mathcal{I} \times \Delta^\mathcal{I} \to [0,1]$. The interpretation function is extended to arbitrary $*$-$\mathcal{ALE}$ concepts as follows. For every $\delta \in \Delta^\mathcal{I}$,

$$\begin{aligned}
\top^\mathcal{I}(\delta) &= 1,\\
\bot^\mathcal{I}(\delta) &= 0,\\
(C_1 \sqcap C_2)^\mathcal{I}(\delta) &= C_1^\mathcal{I}(\delta) \otimes C_2^\mathcal{I}(\delta)\\
(C_1 \to C_2)^\mathcal{I}(\delta) &= C_1^\mathcal{I}(\delta) \Rightarrow C_2^\mathcal{I}(\delta)\\
(\exists r.C)^\mathcal{I}(\delta) &= \sup_{\gamma \in \Delta^\mathcal{I}} r^\mathcal{I}(\delta,\gamma) \otimes C^\mathcal{I}(\gamma)\\
(\forall r.C)^\mathcal{I}(\delta) &= \inf_{\gamma \in \Delta^\mathcal{I}} r^\mathcal{I}(\delta,\gamma) \Rightarrow C^\mathcal{I}(\gamma).
\end{aligned}$$

The interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ satisfies the assertional axiom $\langle a : C \rhd q\rangle$ iff $C^\mathcal{I}(a^\mathcal{I}) \rhd q$, it satisfies $\langle (a,b) : r \rhd q\rangle$ iff $r^\mathcal{I}(a^\mathcal{I}, b^\mathcal{I}) \rhd q$ and it satisfies the concept inclusion $\langle C \sqsubseteq D \geq q\rangle$ iff $\inf_{\delta\in\Delta^\mathcal{I}}(C^\mathcal{I}(\delta) \Rightarrow D^\mathcal{I}(\delta)) \geq q$. This interpretation is called a *model* of the ontology $\mathcal{O}$ if it satisfies all the axioms in $\mathcal{O}$.

In fuzzy DLs, reasoning is often restricted to witnessed models [10]. An interpretation $\mathcal{I}$ is called *witnessed* if it satisfies the following two conditions:

(wit1) for every $\delta \in \Delta^\mathcal{I}$, role r and concept C there exists $\gamma \in \Delta^\mathcal{I}$ such that $(\exists r.C)^\mathcal{I}(\delta) = r^\mathcal{I}(\delta,\gamma) \otimes C^\mathcal{I}(\gamma)$, and

(wit2) for every $\delta \in \Delta^\mathcal{I}$, role r and concept C there exists $\gamma \in \Delta^\mathcal{I}$ such that $(\forall r.C)^\mathcal{I}(\delta) = r^\mathcal{I}(\delta,\gamma) \Rightarrow C^\mathcal{I}(\gamma)$.

This model is called *weakly witnessed* if it satisfies **(wit1)** and *quasi-witnessed* if it satisfies **(wit1)** and the condition

(wit2') for every $\delta \in \Delta^\mathcal{I}$, role r and concept C, either $(\forall r.C)^\mathcal{I} = 0$ or there exists $\gamma \in \Delta^\mathcal{I}$ such that $(\forall r.C)^\mathcal{I}(\delta) = r^\mathcal{I}(\delta,\gamma) \Rightarrow C^\mathcal{I}(\gamma)$.

In the presence of GCIs, witnessed interpretations are sometimes further restricted [6,2,8] to satisfy

(wit3) for every two concepts C, D, there is a γ such that

$$\inf_{\eta\in\Delta^\mathcal{I}}(C^\mathcal{I}(\eta) \Rightarrow D^\mathcal{I}(\eta)) = C^\mathcal{I}(\gamma) \Rightarrow D^\mathcal{I}(\gamma).$$

Witnessed interpretations that satisfy this third restriction **(wit3)** are called *strongly witnessed* interpretations.

We say that an ontology $\mathcal{O}$ is *consistent* (resp. *weakly witnessed consistent*, *quasi-witnessed consistent*, *witnessed consistent*, *strongly witnessed consistent*) if it has a model (resp. a weakly witnessed model, a quasi-witnessed model, a witnessed model, a strongly witnessed model). Obviously, strongly witnessed consistency implies witnessed consistency, which implies quasi-witnessed consistency, which itself implies weakly witnessed consistency. The converse implications, however, need not hold; for instance, a quasi-witnessed consistent $*$-$\mathcal{ALE}$ ontology that has no witnessed models can be derived from the example in [7].

We now describe some properties of t-norms and axioms that will be useful for the rest of the paper. For every $\alpha,\beta \in [0,1]$ it holds that $\alpha \Rightarrow \beta = 1$ iff $\alpha \leq \beta$. Thus, given two concepts C, D, the axiom $\langle C \sqsubseteq D \geq 1\rangle$ expresses that $C^{\mathcal{I}}(\delta) \leq D^{\mathcal{I}}(\delta)$ for all $\delta \in \Delta^{\mathcal{I}}$. Additionally, $1 \Rightarrow \beta = \beta$ and $0 \Rightarrow \beta = 1$ for all $\beta \in [0,1]$, and $\alpha \Rightarrow 0 = 0$ for all $\alpha \in (0,1]$.

In the following, we will use the expression $\langle C \overset{r}{\rightsquigarrow} D\rangle$ to abbreviate the axioms $\langle C \sqsubseteq \forall r.D \geq 1\rangle, \langle \exists r.D \sqsubseteq C \geq 1\rangle$. To understand this abbreviation, consider an interpretation $\mathcal{I}$ satisfying $\langle C \overset{r}{\rightsquigarrow} D\rangle$ and let $\delta,\gamma \in \Delta^{\mathcal{I}}$ with $r^{\mathcal{I}}(\delta,\gamma) = 1$. From the first axiom it follows that

$$\begin{aligned} C^{\mathcal{I}}(\delta) \leq (\forall r.D)^{\mathcal{I}}(\delta) &= \inf_{\eta\in\Delta^{\mathcal{I}}} r^{\mathcal{I}}(\delta,\eta) \Rightarrow D^{\mathcal{I}}(\eta) \\ &\leq r^{\mathcal{I}}(\delta,\gamma) \Rightarrow D^{\mathcal{I}}(\gamma) = 1 \Rightarrow D^{\mathcal{I}}(\gamma) = D^{\mathcal{I}}(\gamma). \end{aligned}$$

From the second axiom it follows that

$$\begin{aligned} C^{\mathcal{I}}(\delta) \geq (\exists r.D)^{\mathcal{I}}(\delta) &= \sup_{\eta\in\Delta^{\mathcal{I}}} r^{\mathcal{I}}(\delta,\eta) \cdot D^{\mathcal{I}}(\eta) \\ &\geq r^{\mathcal{I}}(\delta,\gamma) \cdot D^{\mathcal{I}}(\gamma) = D^{\mathcal{I}}(\gamma), \end{aligned}$$

and hence, both axioms together imply that $C^{\mathcal{I}}(\delta) = D^{\mathcal{I}}(\gamma)$. In other words, $\langle C \overset{r}{\rightsquigarrow} D\rangle$ expresses that the value of $C^{\mathcal{I}}(\delta)$ is propagated to the valuation of the concept D on all r successors with degree 1 of δ. Conversely, given an interpretation $\mathcal{I}$ such that $r^{\mathcal{I}}(\delta,\gamma) \in \{0,1\}$ for all $\delta,\gamma \in \Delta^{\mathcal{I}}$, if $r^{\mathcal{I}}(\delta,\gamma) = 1$ implies $C^{\mathcal{I}}(\delta) = D^{\mathcal{I}}(\gamma)$, then $\mathcal{I}$ is a model of $\langle C \overset{r}{\rightsquigarrow} D\rangle$.

For a concept C, and a natural number $n \geq 1$, the expression C^n will denote the concatenation of C with itself n times; that is, $C^n := \bigsqcap_{j=1}^{n} C$. The semantics of $\sqcap$ yields $(C^n)^{\mathcal{I}}(\delta) = (C^{\mathcal{I}}(\delta))^n$, for every model $\mathcal{I}$ and $\delta \in \Delta^{\mathcal{I}}$.

We will show that consistency of $*$-$\mathcal{ALE}$ ontologies w.r.t. the different variants of witnessed models introduced above is undecidable. We will show this using a reduction from the Post correspondence problem, which is well-known to be undecidable [13].

Definition 1 (PCP). *Let $(v_1, w_1), \ldots, (v_m, w_m)$ be a finite list of pairs of words over an alphabet $\Sigma = \{1, \ldots, s\}, s > 1$. The* Post correspondence problem *(PCP)* *asks whether there is a non-empty sequence $i_1, i_2, \dots, i_k$, $1 \leq i_j \leq m$ such that $v_{i_1} v_{i_2} \cdots v_{i_k} = w_{i_1} w_{i_2} \cdots w_{i_k}$. If such a sequence exists, then the word $i_1 i_2 \cdots i_k$ is called a* solution *of the problem.*

We assume w.l.o.g. that there is no pair v_i, w_i where both words are empty. For a word $\mu = i_1 i_2 \cdots i_k \in \{1, \dots, m\}^*$, we will denote as v_μ and w_μ the words $v_{i_1} v_{i_2} \cdots v_{i_k}$ and $w_{i_1} w_{i_2} \cdots w_{i_k}$, respectively.

The alphabet Σ consists of the first s positive integers. We can thus view every word in Σ^* as a natural number represented in base $s+1$ in which 0 never occurs. Using this intuition, we will express the empty word as the number 0.

In the following reductions, we will encode the word w in Σ^* using the number $2^{-w} \in [0,1]$. We will construct an ontology whose models encode the search for a solution. The interpretation of two designated concept names A and B at a node will correspond to the words v_μ, w_μ, respectively, for $\mu \in \{1, \dots, m\}^*$.

3 Undecidability w.r.t. Witnessed Models

We will show undecidability of consistency w.r.t. witnessed models by constructing, for a given instance $\mathcal{P} = ((v_1, w_1), \dots, (v_m, w_m))$ of the PCP, an ontology $\mathcal{O}_\mathcal{P}$ such that for every witnessed model $\mathcal{I}$ of $\mathcal{O}_\mathcal{P}$ and every $\mu \in \{1, \dots, m\}^*$ there is an element $\delta_\mu \in \Delta^\mathcal{I}$ with $A^\mathcal{I}(\delta_\mu) = 2^{-v_\mu}$ and $B^\mathcal{I}(\delta_\mu) = 2^{-w_\mu}$. Additionally, we will show that this ontology has a witnessed model whose domain has only these elements. Then, $\mathcal{P}$ has a solution iff for every witnessed model $\mathcal{I}$ of $\mathcal{O}_\mathcal{P}$ there exist a $\delta \in \Delta^\mathcal{I}$ such that $A^\mathcal{I}(\delta) = B^\mathcal{I}(\delta)$.

Let $\delta \in \Delta^\mathcal{I}$ encode the words $v, w \in \Sigma^*$; i.e., $A^\mathcal{I}(\delta) = 2^{-v}$ and $B^\mathcal{I}(\delta) = 2^{-w}$, and let $i, 1 \leq i \leq m$. Assume additionally that we have concept names V_i, W_i with $V_i^\mathcal{I}(\delta) = 2^{-v_i}$ and $W_i^\mathcal{I}(\delta) = 2^{-w_i}$. We want to ensure the existence of a node γ that encodes the concatenation of the words v, w with the i-th pair from $\mathcal{P}$; i.e. vv_i and ww_i. This is done through the TBox

$$\mathcal{T}_\mathcal{P}^i := \{\langle \top \sqsubseteq \exists r_i.\top \geq 1\rangle, \langle (V_i \sqcap A^{(s+1)^{|v_i|}}) \overset{r_i}{\rightsquigarrow} A\rangle, \langle (W_i \sqcap B^{(s+1)^{|w_i|}}) \overset{r_i}{\rightsquigarrow} B\rangle\}.$$

Recall that we are viewing words in Σ^* as natural numbers in base $s+1$. Thus, the concatenation of two words u, u' corresponds to the operation $u \cdot (s+1)^{|u'|} + u'$. We then have that

$$(V_i \sqcap A^{(s+1)^{|v_i|}})^\mathcal{I}(\delta) = V_i^\mathcal{I}(\delta) \cdot (A^\mathcal{I}(\delta))^{(s+1)^{|v_i|}} = 2^{-vv_i}.$$

If $\mathcal{I}$ is a witnessed model of $\mathcal{T}_\mathcal{P}^i$, then from the first axiom it follows that $(\exists r_i.\top)^\mathcal{I}(\delta) = 1$, and according to **(wit1)**, there must exist a $\gamma \in \Delta^\mathcal{I}$ with $r^\mathcal{I}(\delta, \gamma) = 1$. The last two axioms then ensure that $A^\mathcal{I}(\gamma) = 2^{-vv_i}$ and $B^\mathcal{I}(\gamma) = 2^{-ww_i}$; thus, the concept names A and B encode, at node γ, the words vv_i and ww_i, as desired. If we want to use this construction to recursively construct all the pairs of concatenated words defined by $\mathcal{P}$, we need to ensure also that $V_j^\mathcal{I}(\gamma) = 2^{-v_j}$, $W_j^\mathcal{I}(\gamma) = 2^{-w_j}$ hold for every $j, 1 \leq j \leq m$. This can be done through the axioms

$$\mathcal{T}_\mathcal{P}^0 := \{\langle V_j \overset{r_i}{\rightsquigarrow} V_j\rangle, \langle W_j \overset{r_i}{\rightsquigarrow} W_j\rangle \mid 1 \leq i, j \leq m\}.$$

It only remains to ensure that there is a node δ_ε where $A^{\mathcal{I}}(\delta_\varepsilon) = B^{\mathcal{I}}(\delta_\varepsilon) = 1 = 2^0$ (that is, where A and B encode the empty word) and $V_j^{\mathcal{I}}(\delta_\varepsilon) = 2^{-v_j}$, $W_j^{\mathcal{I}}(\delta_\varepsilon) = 2^{-w_j}$ hold for every $j, 1 \leq i \leq m$. This condition is easily enforced through the ABox[3]

$$\begin{aligned}\mathcal{A}_{\mathcal{P}}^0 := \{&\langle a : A = 1\rangle, \langle a : B = 1\rangle\}\cup \\ &\{\langle a : V_i = 2^{-v_i}\rangle, \langle a : W_i = 2^{-w_i}\rangle \mid 1 \leq i \leq m\}.\end{aligned}$$

Finally, we include a concept name H that must be interpreted as 0.5 in every domain element. This is enforced by the following axioms:

$$\begin{aligned}\mathcal{A}_0 &:= \{\langle a : H = 0.5\rangle\}, \\ \mathcal{T}_0 &:= \{\langle H \overset{r_i}{\rightsquigarrow} H\rangle \mid 1 \leq i \leq m\}.\end{aligned}$$

This concept name will later be used to detect whether $\mathcal{P}$ has a solution (see Theorem 3).

Let now $\mathcal{O}_{\mathcal{P}} := (\mathcal{A}_{\mathcal{P}}, \mathcal{T}_{\mathcal{P}})$ where $\mathcal{A}_{\mathcal{P}} = \mathcal{A}_{\mathcal{P}}^0 \cup \mathcal{A}_0$ and $\mathcal{T}_{\mathcal{P}} := \mathcal{T}_0 \cup \bigcup_{i=0}^m \mathcal{T}_{\mathcal{P}}^i$. We define the interpretation $\mathcal{I}_{\mathcal{P}} := (\Delta^{\mathcal{I}_{\mathcal{P}}}, \cdot^{\mathcal{I}_{\mathcal{P}}})$ as follows:

- $\Delta^{\mathcal{I}_{\mathcal{P}}} = \{1, \ldots, m\}^*$,
- $a^{\mathcal{I}_{\mathcal{P}}} = \varepsilon$,

for every $\mu \in \Delta^{\mathcal{I}_{\mathcal{P}}}$,

- $A^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-v_\mu}$, $B^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-w_\mu}$, $H^{\mathcal{I}_{\mathcal{P}}}(\mu) = 0.5$,

and for all $j, 1 \leq j \leq m$

- $V_j^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-v_j}$, $W_j^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-w_j}$, and
- $r_j^{\mathcal{I}_{\mathcal{P}}}(\mu, \mu j) = 1$ and $r_j^{\mathcal{I}_{\mathcal{P}}}(\mu, \mu') = 0$ if $\mu' \neq \mu j$.

It is easy to see that $\mathcal{I}_{\mathcal{P}}$ is in fact a witnessed model of $\mathcal{O}_{\mathcal{P}}$, since every node has exactly one r_i successor with degree greater than 0, for every $i, 1 \leq i \leq m$. More interesting, however, is that for every witnessed model $\mathcal{I}$ of $\mathcal{O}_{\mathcal{P}}$, there is an homomorphism from $\mathcal{I}_{\mathcal{P}}$ to $\mathcal{I}$ as described in the following lemma.

Lemma 2. *Let $\mathcal{I}$ be a witnessed model of $\mathcal{O}_{\mathcal{P}}$. Then there exists a function $f : \Delta^{\mathcal{I}_{\mathcal{P}}} \to \Delta^{\mathcal{I}}$ such that, for every $\mu \in \Delta^{\mathcal{I}_{\mathcal{P}}}$, $C^{\mathcal{I}_{\mathcal{P}}}(\mu) = C^{\mathcal{I}}(f(\mu))$ holds for every concept name C and $r_i^{\mathcal{I}}(f(\mu), f(\mu i)) = 1$ holds for every $i, 1 \leq i \leq m$.*

Proof. The function f is built inductively on the length of μ. First, as $\mathcal{I}$ is a model of $\mathcal{A}_{\mathcal{P}}$, there must be a $\delta \in \Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} = \delta$. Notice that $\mathcal{A}_{\mathcal{P}}$ fixes the interpretation of all concept names on δ and hence $f(\varepsilon) = \delta$ satisfies the condition of the lemma.

[3] Notice that equality is necessary for this construction; since there is no negation constructor, it is not possible to express $\langle a : X = q\rangle$ with $q < 1$ using only axioms of the form $\langle a : Y \geq q'\rangle$.

Let now μ be such that $f(\mu)$ has already been defined. By induction, we can assume that $A^{\mathcal{I}}(f(\mu)) = 2^{-v_\mu}, B^{\mathcal{I}}(f(\mu)) = 2^{-w_\mu}, H^{\mathcal{I}}(f(\mu)) = 0.5$, and for every j, $1 \leq j \leq m$, $V_j^{\mathcal{I}}(f(\mu)) = 2^{-v_j}, W_j^{\mathcal{I}}(f(\mu)) = 2^{-w_j}$. Since $\mathcal{I}$ is a witnessed model of $\langle \top \sqsubseteq \exists r_i.\top \geq 1\rangle$, for all $i, 1 \leq i \leq m$ there exists a $\gamma \in \Delta^{\mathcal{I}}$ with $r^{\mathcal{I}}(f(\mu), \gamma) = 1$, and as $\mathcal{I}$ satisfies all the axioms of the form $\langle C \overset{r}{\rightsquigarrow} D\rangle \in \mathcal{T}_{\mathcal{P}}$, it follows that

$$A^{\mathcal{I}}(\gamma) = 2^{-v_\mu v_i} = 2^{-v_{\mu i}}, \quad B^{\mathcal{I}}(\gamma) = 2^{-w_\mu w_i} = 2^{-w_{\mu i}},$$

$H^{\mathcal{I}}(\gamma) = 0.5$ and for all $j, 1 \leq j \leq m$, $V_j^{\mathcal{I}}(\gamma) = 2^{-v_j}, W_j^{\mathcal{I}}(\gamma) = 2^{-w_j}$. Setting $f(\mu i) = \gamma$ thus satisfies the required property. □

From this lemma it follows that, if the PCP $\mathcal{P}$ has a solution μ for some $\mu \in \{1, \ldots, m\}^+$, then every witnessed model $\mathcal{I}$ of $\mathcal{O}_{\mathcal{P}}$ contains a node $\delta = f(\mu)$ such that $A^{\mathcal{I}}(\delta) = B^{\mathcal{I}}(\delta)$; that is, where A and B encode the same word. Conversely, if every witnessed model contains such a node, then in particular $\mathcal{I}_{\mathcal{P}}$ does, and thus $\mathcal{P}$ has a solution. The question is now how to detect whether a node with this characteristics exists in every model. We will extend $\mathcal{O}_{\mathcal{P}}$ with axioms that further restrict $\mathcal{I}_{\mathcal{P}}$ to satisfy $A^{\mathcal{I}_{\mathcal{P}}}(\mu) \neq B^{\mathcal{I}_{\mathcal{P}}}(\mu)$ for every $\mu \in \{1, \ldots, m\}^+$. This will ensure that the extended ontology will have a model iff $\mathcal{P}$ has no solution.

Suppose for now that, for some $\mu \in \{1, \ldots, m\}^*$, it holds that

$$2^{-v_\mu} = A^{\mathcal{I}_{\mathcal{P}}}(\mu) > B^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-w_\mu}.$$

We then have that $v_\mu < w_\mu$ and hence $w_\mu - v_\mu \geq 1$. It thus follows that

$$(A \to B)^{\mathcal{I}_{\mathcal{P}}}(\mu) = 2^{-w_\mu}/2^{-v_\mu} = 2^{-(w_\mu - v_\mu)} \leq 2^{-1} = 0.5$$

and thus $((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu) \leq 0.5$. Likewise, if $A^{\mathcal{I}_{\mathcal{P}}}(\mu) < B^{\mathcal{I}_{\mathcal{P}}}(\mu)$, we also get $((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu) \leq 0.5$. Additionally, if $A^{\mathcal{I}_{\mathcal{P}}}(\mu) = B^{\mathcal{I}_{\mathcal{P}}}(\mu)$, then it is easy to verify that $((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu) = 1$. From all this it follows that, for every $\mu \in \{1, \ldots, m\}^*$,

$$A^{\mathcal{I}_{\mathcal{P}}}(\mu) \neq B^{\mathcal{I}_{\mathcal{P}}}(\mu) \quad \text{iff} \quad ((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu) \leq 0.5. \tag{1}$$

Thus, the instance $\mathcal{P}$ has no solution iff for every $\mu \in \{1, \ldots, m\}^+$ it holds that $((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu) \leq 0.5$.

We define now the ontology $\mathcal{O}'_{\mathcal{P}} := (\mathcal{A}_{\mathcal{P}}, \mathcal{T}'_{\mathcal{P}})$ where

$$\mathcal{T}'_{\mathcal{P}} := \mathcal{T}_{\mathcal{P}} \cup \{ \langle \top \sqsubseteq \forall r_i.(((A \to B) \sqcap (B \to A)) \to H) \geq 1\rangle \mid 1 \leq i \leq m\}.$$

Theorem 3. *The instance $\mathcal{P}$ of the PCP has a solution iff the ontology $\mathcal{O}'_{\mathcal{P}}$ is not witnessed consistent.*

Proof. Assume first that $\mathcal{P}$ has a solution $\mu = i_1 \cdots i_k$ and let $u = v_\mu = w_\mu$ and $\mu' = i_1 i_2 \cdots i_{k-1} \in \{1, \ldots, m\}^*$. Suppose there is a witnessed model $\mathcal{I}$ of $\mathcal{O}'_{\mathcal{P}}$. Since $\mathcal{O}_{\mathcal{P}} \subseteq \mathcal{O}'_{\mathcal{P}}$, $\mathcal{I}$ must also be a model of $\mathcal{O}_{\mathcal{P}}$. From Lemma 2 it then follows that there are nodes $\delta,\delta' \in \Delta^{\mathcal{I}}$ such that $A^{\mathcal{I}}(\delta) = A^{\mathcal{I}_{\mathcal{P}}}(\mu) = B^{\mathcal{I}_{\mathcal{P}}}(\mu) = B^{\mathcal{I}}(\delta)$ and $r^{\mathcal{I}}_{i_k}(\delta',\delta) = 1$. Then, $((A \to B) \sqcap (B \to A))^{\mathcal{I}}(\delta) = 1$ and hence

$$(((A \to B) \sqcap (B \to A)) \to H)^{\mathcal{I}}(\delta) = 1 \Rightarrow 0.5 = 0.5.$$

This then means that $(\forall r_{i_k}.(((A \to B) \sqcap (B \to A)) \to H))^{\mathcal{I}}(\delta') \leq 0.5$, violating one of the axioms in $\mathcal{T}'_{\mathcal{P}}$. Hence $\mathcal{I}$ is cannot be a model of $\mathcal{O}'_{\mathcal{P}}$.

For the converse, assume that $\mathcal{O}'_{\mathcal{P}}$ is not witnessed consistent. Then $\mathcal{I}_{\mathcal{P}}$ is not a model of $\mathcal{O}'_{\mathcal{P}}$. Since it is a model of $\mathcal{O}_{\mathcal{P}}$, there must exist an $i, 1 \leq i \leq m$ such that $\mathcal{I}_{\mathcal{P}}$ violates the axiom $\langle \top \sqsubseteq \forall r_i.(((A \to B) \sqcap (B \to A)) \to H) \geq 1\rangle$. This means that there is some $\mu \in \{1,\ldots,m\}^*$ such that

$$(\forall r_i.(((A \to B) \sqcap (B \to A)) \to H))^{\mathcal{I}_{\mathcal{P}}}(\mu) < 1.$$

Since $r_i^{\mathcal{I}_{\mathcal{P}}}(\mu,\mu') = 0$ for all $\mu' \neq \mu i$ and $r_i^{\mathcal{I}_{\mathcal{P}}}(\mu,\mu i) = 1$, this implies that $(((A \to B) \sqcap (B \to A)) \to H)^{\mathcal{I}_{\mathcal{P}}}(\mu i) < 1$, i.e. $((A \to B) \sqcap (B \to A))^{\mathcal{I}_{\mathcal{P}}}(\mu i) > 0.5$. From (1) it follows that $A^{\mathcal{I}_{\mathcal{P}}}(\mu i) = B^{\mathcal{I}_{\mathcal{P}}}(\mu i)$ and hence μi is a solution of $\mathcal{P}$. $\square$

Corollary 4. *Witnessed consistency of $*$-$\mathcal{ALE}$ ontologies is undecidable.*

Notice that in the proofs of Lemma 2 and Theorem 3, the second condition of the definition of witnessed models was never used. Moreover, the witnessed interpretation $\mathcal{I}_{\mathcal{P}}$ is obviously also weakly witnessed. We thus have the following corollary.

Corollary 5. *Weakly witnessed consistency and quasi-witnessed consistency of $*$-$\mathcal{ALE}$ ontologies are undecidable.*

4 Undecidability w.r.t. Strongly Witnessed Models

Unfortunately, the model $\mathcal{I}_{\mathcal{P}}$ constructed in the previous section is not a strongly witnessed model of $\mathcal{O}_{\mathcal{P}}$ since, for instance, $\inf_{\eta \in \Delta^{\mathcal{I}_{\mathcal{P}}}}(\top^{\mathcal{I}_{\mathcal{P}}}(\eta) \Rightarrow A^{\mathcal{I}_{\mathcal{P}}}(\eta)) = 0$, but there is no $\delta \in \Delta^{\mathcal{I}_{\mathcal{P}}}$ with $A^{\mathcal{I}_{\mathcal{P}}}(\delta) = 0$. Thus, the construction of $\mathcal{O}_{\mathcal{P}}$ does not yield an undecidability result for strongly witnessed consistency in $*$-$\mathcal{ALE}$.

Thus, we need a new reduction that proves undecidability of strongly witnessed consistency. This reduction will follow a similar idea to the one used in the previous section, in which models describe a search for a solution of the PCP $\mathcal{P}$. However, rather than building the whole search tree, models will describe only individual branches of this tree. The condition **(wit3)** will be used to ensure that at some point in this branch a solution is found.

Before describing the reduction in detail, we recall a property of t-norms. From a t-norm $\otimes$ and residuum $\Rightarrow$, one can express the minimum and maximum operators as follows [9]:

- $\min(\alpha,\beta) = \alpha \otimes (\alpha \Rightarrow \beta)$,
- $\max(\alpha,\beta) = \min(((\alpha \Rightarrow \beta) \Rightarrow \beta), ((\beta \Rightarrow \alpha) \Rightarrow \alpha))$.

We can thus introduce w.l.o.g. the $*$-$\mathcal{ALE}$ concept constructor max with the obvious semantics. We will use this constructor to simulate the non-deterministic choices in the search tree as described next.

Given an instance $\mathcal{P} = ((v_1, w_1), \dots, (v_m, w_m))$ of the PCP, we define the ABox $\mathcal{A}^0_{\mathcal{P}}$ and the TBox $\mathcal{T}^0_{\mathcal{P}}$ as in the previous section, and for every $i, 1 \leq i \leq m$ we construct the TBox

$$\mathcal{T}^{s}{}^{i}_{\mathcal{P}} := \{\langle C_i \sqsubseteq \exists r_i.\top \geq 1\rangle, \langle V_i \sqcap A^{(s+1)^{|v_i|}} \overset{r_i}{\rightsquigarrow} A\rangle, \langle W_i \sqcap B^{(s+1)^{|w_i|}} \overset{r_i}{\rightsquigarrow} B\rangle\}.$$

The only difference between the TBoxes $\mathcal{T}^i_{\mathcal{P}}$ and $\mathcal{T}^{s}{}^{i}_{\mathcal{P}}$ is in the first axiom. Intuitively, the concept names C_i encode the choice of the branch in the tree to be expanded. If $C_i^{\mathcal{I}}(\delta) = 1$, there will be an r_i successor with degree 1, and the i-th branch of the tree will be explored. For this intuition to work, we need to ensure that at least one of the C_is is interpreted as 1 in every node. On the other hand, we can stop expanding the tree once a solution has been found. Using this intuition, we define the ontology $\mathcal{O}^s_{\mathcal{P}} := (\mathcal{A}^s_{\mathcal{P}}, \mathcal{T}^s_{\mathcal{P}})$ where

$$\begin{aligned}
\mathcal{A}^s_{\mathcal{P}} &:= \mathcal{A}^0_{\mathcal{P}} \cup \{a : \max(C_1, \dots, C_m) = 1\},\\
\mathcal{T}^s_{\mathcal{P}} &:= \mathcal{T}^0_{\mathcal{P}} \cup \bigcup_{i=1}^{m} \mathcal{T}^{s}{}^{i}_{\mathcal{P}} \cup \{\langle\, (A \sqcap B) \to \bot \sqsubseteq \bot \geq 1\rangle\} \cup \\
&\qquad \{\langle \top \sqsubseteq \forall r_i. \max((A \to B) \sqcap (B \to A), C_1, \dots, C_m) \geq 1\rangle \mid 1 \leq i \leq m\}.
\end{aligned}$$

Theorem 6. *The instance $\mathcal{P}$ of the PCP has a solution iff the ontology $\mathcal{O}^s_{\mathcal{P}}$ is strongly witnessed consistent.*

Proof. Let $\nu = i_1 i_2 \cdots i_k$ be a solution of $\mathcal{P}$ and let $\mathsf{pre}(\nu)$ denote the set of all prefixes of ν. We build the finite interpretation $\mathcal{I}^s_{\mathcal{P}}$ as follows:

- $\Delta^{\mathcal{I}^s_{\mathcal{P}}} := \mathsf{pre}(\nu)$,
- $a^{\mathcal{I}^s_{\mathcal{P}}} = \varepsilon$,

for all $\mu \in \Delta^{\mathcal{I}^s_{\mathcal{P}}}$,

- $A^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 2^{-v_\mu}$, $B^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 2^{-w_\mu}$,

and for all $j, 1 \leq j \leq m$

- $V_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 2^{-v_j}$, $W_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 2^{-w_j}$,
- $C_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 1$ if $\mu j \in \mathsf{pre}(\nu)$ and $C_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu) = 0$ otherwise, and
- $r_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu, \mu j) = 1$ if $\mu j \in \mathsf{pre}(\nu)$ and $r_j^{\mathcal{I}^s_{\mathcal{P}}}(\mu, \mu') = 0$ if $\mu' \in \mathsf{pre}(\nu)$ and $\mu' \neq \mu j$.

We show now that $\mathcal{I}^s_{\mathcal{P}}$ is a model of $\mathcal{O}^s_{\mathcal{P}}$. Since $\mathcal{I}^s_{\mathcal{P}}$ is finite, it follows immediately that it is also strongly witnessed. Clearly $\mathcal{I}^s_{\mathcal{P}}$ satisfies all axioms in $\mathcal{A}^0_{\mathcal{P}}$; additionally, we have that $C_{i_1}^{\mathcal{I}^s_{\mathcal{P}}}(\varepsilon) = 1$ and thus, $\mathcal{I}^s_{\mathcal{P}}$ satisfies $\mathcal{A}^s_{\mathcal{P}}$. The axiom $\langle (A \sqcap B) \to \bot \sqsubseteq \bot \geq 1\rangle$ expresses that $(A \sqcap B)^{\mathcal{I}^s_{\mathcal{P}}}(\mu) \Rightarrow 0 = 0$, and hence $(A \sqcap B)^{\mathcal{I}^s_{\mathcal{P}}}(\mu) > 0$ for all $\mu \in \mathsf{pre}(\nu)$, which clearly holds. We now show that the rest of the axioms are also satisfied for every $\mu \in \mathsf{pre}(\nu)$. Let

$\mu \in \mathsf{pre}(\nu) \setminus \{\nu\}$. Then we know that there exists $i, 1 \le i \le m$ such that $C_i^{\mathcal{I}^s_\mathcal{P}}(\mu) = 1$ and $r_i^{\mathcal{I}^s_\mathcal{P}}(\mu, \mu i) = 1$; thus $\mathcal{I}^s_\mathcal{P}$ satisfies the axioms in ${\mathcal{T}^s}^i_\mathcal{P}$. Moreover, $C_j^{\mathcal{I}^s_\mathcal{P}}(\mu) = 0 = r_j^{\mathcal{I}^s_\mathcal{P}}(\mu, \mu')$ for all $j \neq i$ and all $\mu' \in \mathsf{pre}(\nu)$ which means that $\mathcal{I}^s_\mathcal{P}$ trivially satisfies all axioms in ${\mathcal{T}^s}^j_\mathcal{P}$.

If $\mu i = \nu$, then as ν is a solution $((A \to B) \sqcap (B \to A))^{\mathcal{I}^s_\mathcal{P}}(\mu i) = 1$; otherwise, there is a $j, 1 \le j \le m$ with $\mu i j \in \mathsf{pre}(\nu)$ and thus $C_j^{\mathcal{I}^s_\mathcal{P}}(\mu i) = 1$. This means that $\mathcal{I}^s_\mathcal{P}$ satisfies the last axioms in $\mathcal{T}^s_\mathcal{P}$. Finally, if $\mu = \nu$, then $r_i^{\mathcal{I}^s_\mathcal{P}}(\mu, \mu') = 0$ and $C_i(\mu) = 0$, for all $\mu' \in \mathsf{pre}(\nu), 1 \le i \le m$, and thus the axioms are all trivially satisfied.

For the converse, let $\mathcal{I}$ be a strongly witnessed model of $\mathcal{O}^s_\mathcal{P}$. Then, there must be an element $\delta_0 \in \Delta^\mathcal{I}$ with $a^\mathcal{I} = \delta_0$. Since $\mathcal{I}$ must satisfy all axioms in $\mathcal{A}^s_\mathcal{P}$, there is an $i_1, 1 \le i_1 \le m$ such that $C_{i_1}^\mathcal{I}(\delta_0) = 1$. Since it must satisfy the axioms in ${\mathcal{T}^s}^{i_1}_\mathcal{P}$, there must exist a $\delta_1 \in \Delta^\mathcal{I}$ with $r_{i_1}^\mathcal{I}(\delta_0, \delta_1) = 1$, $A^\mathcal{I}(\delta_1) = 2^{-v_{i_1}}$, and $B^\mathcal{I}(\delta_1) = 2^{-w_{i_1}}$. If $A^\mathcal{I}(\delta_1) = B^\mathcal{I}(\delta_1)$, then i_1 is a solution of $\mathcal{P}$. Otherwise, from the last set of axioms in $\mathcal{T}^s_\mathcal{P}$, there must exist an $i_2, 1 \le i_2 \le m$ with $C_{i_2}^\mathcal{I}(\delta_1) = 1$. We can then iterate this same process to generate a sequence $i_3, i_4, \ldots$ of indices and $\delta_2, \delta_3, \ldots \in \Delta^\mathcal{I}$ where $A^\mathcal{I}(\delta_k) = 2^{-v_{i_1} \cdots v_{i_k}}$, and $B^\mathcal{I}(\delta_k) = 2^{-w_{i_1} \cdots w_{i_k}}$.

If there is some k such that $A^\mathcal{I}(\delta_k) = B^\mathcal{I}(\delta_k)$, then $i_1 \cdots i_k$ is a solution of $\mathcal{P}$. Assume now that no such k exists. We then have an infinite sequence of indices $i_1, i_2, \ldots$ and since for every $i, 1 \le i \le m$ either $v_i \neq 0$ or $w_i \neq 0$, then at least one of the sequences $v_{i_1} \cdots v_{i_k}, w_{i_1} \cdots w_{i_k}$ diverges. Thus, for every natural number n there is a k such that either $v_{i_1} \cdots v_{i_k} > n$ or $w_{i_1} \cdots w_{i_k} > n$; equivalently, $(A \sqcap B)^\mathcal{I}(\delta_k) < 1/n$. This implies that

$$\inf_{\eta \in \Delta^\mathcal{I}} (\top^\mathcal{I}(\eta) \Rightarrow (A \sqcap B)^\mathcal{I}(\eta)) = 0$$

and since $\mathcal{I}$ is strongly witnessed, there must exist a $\gamma \in \Delta^\mathcal{I}$ with

$$0 = \top^\mathcal{I}(\gamma) \Rightarrow (A \sqcap B)^\mathcal{I}(\gamma) = (A \sqcap B)^\mathcal{I}(\gamma).$$

But from this it follows that $((A \sqcap B) \to \bot)^\mathcal{I}(\gamma) \Rightarrow 0 = 0$, contradicting the axiom $\langle (A \sqcap B) \to \bot \sqsubseteq \bot \ge 1 \rangle$ of $\mathcal{T}^s_\mathcal{P}$. Thus, $\mathcal{P}$ has a solution. $\square$

Notice that, if $\mathcal{P}$ has no solution, then $\mathcal{O}^s_\mathcal{P}$ still has witnessed models, but no strongly witnessed models. It is also relevant to point out that $\mathcal{O}^s_\mathcal{P}$ has a strongly witnessed model iff it has a finite model. In fact, the condition of strongly witnessed was only used for ensuring finiteness of the model, and hence, that a solution is indeed found.

Corollary 7. *For $*$-$\mathcal{ALE}$ ontologies, strongly witnessed consistency and consistency w.r.t. finite models are undecidable.*

5 Conclusions

We have shown that consistency of $*$-$\mathcal{ALE}$ ontologies w.r.t. a wide variety of models, ranging from finite models to weakly witnessed models, is undecidable if the product t-norm semantics are used. Whether consistency in general, that is, without restricting the class of interpretations used, is also undecidable is still an open problem. In [11] it was shown that, if only *crisp* axioms are used, then consistency is equivalent to quasi-witnessed consistency. However, it is unclear how to extend this result to the fuzzy axioms used in this paper.

As future work we plan to study whether these undecidability results still hold if the disjunction and negation constructors are used in place of the implication considered in this paper. Additionally, we will study the decidability status of these logics if different t-norms are chosen for the semantics.

References

1. F. Baader and R. Peñaloza. Are fuzzy description logics with general concept inclusion axioms decidable? In *Proc. of Fuzz-IEEE 2011*. IEEE, 2011. To appear.
2. F. Bobillo, F. Bou, and U. Straccia. On the failure of the finite model property in some fuzzy description logics. *CoRR*, abs/1003.1588, 2010.
3. F. Bobillo, F. Bou, and U. Straccia. On the failure of the finite model property in some fuzzy description logics. *Fuzzy Sets and Systems*, 172(23):1–12, 2011.
4. F. Bobillo and U. Straccia. A fuzzy description logic with product t-norm. In *Proc. of FUZZ-IEEE 2007*, pages 1–6. IEEE, 2007.
5. F. Bobillo and U. Straccia. On qualified cardinality restrictions in fuzzy description logics underŁ ukasiewicz semantics. In *Proc. of IPMU-08*, pages 1008–1015, 2008.
6. F. Bobillo and U. Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets and Systems*, 160(23):3382–3402, 2009.
7. M. Cerami, F. Esteva, and F. Bou. Decidability of a description logic over infinite-valued product logic. In *Proceedings of KR 2010*. AAAI Press, 2010.
8. A. García-Cerdaña, E. Armengol, and F. Esteva. Fuzzy description logics and t-norm based fuzzy logics. *Int. J. of Approx. Reasoning*, 51:632 – 655, July 2010.
9. P. Hájek. *Metamathematics of Fuzzy Logic (Trends in Logic)*. Springer, 2001.
10. P. Hájek. Making fuzzy description logic more general. *Fuzzy Sets and Systems*, 154(1):1–15, 2005.
11. M. C. Laskowski and S. Malekpour. Provability in predicate product logic. *Arch. Math. Log.*, 46(5-6):365–378, 2007.
12. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.
13. E. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.
14. G. Stoilos, G. B. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks. The fuzzy description logic f-SHIN. In *Proc. of URSW'05*, pages 67–76, 2005.
15. G. Stoilos, U. Straccia, G. B. Stamou and J. Z. Pan. General concept inclusions in fuzzy description logics. In *Proc. of ECAI'06*, pages 457–461, 2006.
16. U. Straccia. Reasoning within fuzzy description logics. *JAIR*, 14:137–166, 2001.
17. U. Straccia. Description logics with fuzzy concrete domains. In *Proc. of UAI'05*, pages 559–567. AUAI Press, 2005.
18. U. Straccia and F. Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. In *Proc. of 5th EUSFLAT Conf.*, pages 213–220, 2007.
19. C. Tresp and R. Molitor. A description logic for vague knowledge. In *Proc. of ECAI'98*, pages 361–365, Brighton, UK, 1998. J. Wiley and Sons.

Verification of Conjunctive-Query Based Semantic Artifacts[⋆]

Babak Bagheri Hariri[1], Diego Calvanese[1],
Giuseppe De Giacomo[2], and Riccardo De Masellis[2]

[1] KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
{bagheri,calvanese}@inf.unibz.it

Dip. di Informatica e Sistemistica
Sapienza Università di Roma
Via Ariosto, 25, 00185 Rome, Italy
lastname@dis.uniroma1.it

Abstract. We introduce semantic artifacts, which are a mechanism that provides both a semantically rich representation of the information on the domain of interest in terms of an ontology, including the underlying data, and a set of actions to change such information over time. In this paper, the ontology is specified as a DL-Lite TBox together with an ABox that may contain both (known) constants and unknown individuals (labeled nulls, represented as Skolem terms). Actions are specified as sets of conditional effects, where conditions are based on conjunctive queries over the ontology (TBox and ABox), and effects are expressed in terms of new ABoxes. In this setting, which is obviously not finite state, we address the verification of temporal/dynamic properties expressed in μ-calculus. Notably, we show decidability of verification, under a suitable restriction inspired by the notion of acyclicity in data exchange.

1 Introduction

The artifact-centric approach to service modeling, introduced recently, considers both data and processes as first-class citizens in service design and analysis. Artifacts are advocated as a sort of middle ground between a conceptual formalization of a dynamic system and an actual implementation of the system itself [1]. The verification of temporal properties in the presence of data represents a significant challenge for the research community, since the system becomes infinite-state, and hence the usual analysis based on model checking of finite-state systems [2] is impossible in general. Recently, there have been some advancements on this issue, considering suitably constrained relational database settings for the data component (which acts also as a data storage for the process), see e.g., [3,4].

In this paper, we follow the line of [3], and rely on the work in knowledge representation to propose a more conceptual treatment of artifacts. We do so by enriching artifacts with a semantic layer constituted by a full-fledged ontology expressed in description logics. In particular, our *semantic artifacts* include an ontology specified in *DL-Lite*$_\mathcal{R}$ [5], which is the core of the web ontology language OWL2 QL[1], and it is particularly well suited for data management. The TBox of the ontology captures intensional information

[⋆] This work has been supported by the EU FP7-ICT Project ACSI (257593).

[1] http://www.w3.org/TR/owl2-profiles/

on the domain of interest, similarly to conceptual data models, such as UML class diagram, though as a software component to be used at runtime. The ABox, which acts as the artifact state, maintains the data of interest, which are accessed by relying on query answering through ontologies. As a query language, we use unions of conjunctive queries, possibly composing their certain answers through full FOL constructs [6]. A semantic artifact has associated actions whose execution changes the state of the artifact, i.e., its ABox. Such actions are specified as sets of conditional effects, where conditions are queries over the ontology and effects are expressed in terms of new ABoxes. To capture data that are acquired from external users/environments during the execution of actions, such ABoxes may contain special constants called labeled nulls, represented as *Skolem terms*. These represent individuals that the artifact does not know, but on which it knows some facts. Actions have no pre-condition, instead processes over the semantic artifact are used to specify which actions can be executed at each step. We model such processes as condition/action rules, where the condition is again expressed as a query over the ontology.

In this setting, which is obviously not finite state, we address the verification of temporal/dynamic properties expressed in μ-calculus [7], where atomic formulas are queries over the ontology that can refer only to known individuals. Unsurprisingly, we show that even for very simple semantic artifacts and dynamic properties verification is undecidable. However, we also show that for a very rich class of semantic artifacts, verification is indeed decidable and reducible to finite-state model checking. To obtain this result, we rely on recent results on the finiteness of the chase of tuple-generating dependencies in the data exchange literature [8].

2 Preliminaries

As ontology language, we make use of *DL-Lite*$_\mathcal{R}$, a member of the *DL-Lite* family [5], and hence a tractable DL particularly suited for dealing with ontologies (or KBs) with very large ABoxes, which can be managed through relational database technology. *DL-Lite*$_\mathcal{R}$ is also the core of the standard web ontology language OWL2 QL. In *DL-Lite*$_\mathcal{R}$, concepts and roles are formed according to the following syntax:

$$\begin{array}{rcl} B & ::= & N \mid \exists U \\ C & ::= & B \mid \neg B \mid \exists U.B \end{array} \qquad\qquad \begin{array}{rcl} U & ::= & P \mid P^- \\ V & ::= & U \mid \neg U \end{array}$$

N, B, and C respectively denote a *concept name*, a *basic concept*, and an *arbitrary concept*. P, P^-, U, and V respectively denote a *role name*, an *inverse role*, a *basic role*, and an *arbitrary role*. A *DL-Lite*$_\mathcal{R}$ ontology is a pair (T, A), where T is a TBox, i.e., a finite set of (concept and role) *inclusion assertions* of the forms $B \sqsubseteq C$ and $U \sqsubseteq V$; and A is an ABox, i.e., a finite set of *facts* (also called membership assertions) of the forms $N(c_1)$ and $P(c_1, c_2)$, where N and P occur in T, and c_1 and c_2 are constants. We denote with $\mathcal{C}_A$ the set of constants appearing in A. The semantics of a *DL-Lite*$_\mathcal{R}$ ontology is the usual one for DLs, see [5]. Notice that in *DL-Lite*$_\mathcal{R}$ the unique name assumption is immaterial, since there is no way of deducing facts about equality. We say that A is *consistent wrt* T if (T, A) is satisfiable, i.e., admits at least one model.

A *union of conjunctive queries* (UCQ) q over a *DL-Lite*$_\mathcal{R}$ TBox T and constants $\mathcal{C}$ is a FOL formula of the form $\exists \boldsymbol{y}_1.conj_1(\boldsymbol{x}, \boldsymbol{y}_1) \lor \cdots \lor \exists \boldsymbol{y}_n.conj_n(\boldsymbol{x}, \boldsymbol{y}_n)$, with free variables $\boldsymbol{x}$, existentially quantified variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. Each $conj_i(\boldsymbol{x}, \boldsymbol{y_i})$ in q is a conjunction of atoms of the form $N(z)$, $P(z, z')$, $z = z'$, where N and P respectively denote a concept and a role name occurring in T, and z, z' are constants in $\mathcal{C}$ or variables in $\boldsymbol{x}$ or $\boldsymbol{y_i}$, for some $i \in \{1, \ldots, n\}$. Given constants $\mathcal{C}$ (typically $\mathcal{C}_A$), the *(certain-)answers to q over (T, A) wrt $\mathcal{C}$* is the set $ans_{\mathcal{C}}(q, T, A)$ of substitutions[2] θ of the free variables of q with constants in $\mathcal{C}$ such that $q\theta$ evaluates to true in every model of (T, A). We also consider an extension of UCQs, called ECQs, which are the range-restricted queries of the query language *EQL-Lite*(UCQ) [6], that is, the FOL query language whose atoms are UCQs evaluated according to the certain answer semantics above. Formally, an *ECQ* over T and $\mathcal{C}$ is a possibly open formula of the form

$$Q \ ::= \ q \mid Q_1 \wedge Q_2 \mid \neg Q \mid \exists x.Q,$$

where q denotes a UCQ over T and $\mathcal{C}$, with the proviso that every free variable of a negative subquery, i.e., of the form $\neg Q$, must occur in a positive subquery (to guarantee range-restriction). Given constants $\mathcal{C}$ the *answer to Q over (T, A) wrt $\mathcal{C}$*, is the set $ans_{\mathcal{C}}(Q, T, A)$ of tuples of constants in $\mathcal{C}$ obtained by *evaluating* the FOL formula Q in the standard way, while considering each UCQ q appearing in it as a primitive predicate with extension $ans_{\mathcal{C}}(q, T, A)$. For the connection with epistemic logic, see [6].

3 Semantic Artifacts

A *semantic artifact* $\mathcal{S} = (T, A_0, R)$ is a stateful device constituted by the information ontology (T, A_0) and the action specification R.

- T is a *DL-Lite*$_\mathcal{R}$ TBox, fixed once and for all, and capturing the intensional knowledge about the domain modeled by the semantic artifact.
- A_0 is a *DL-Lite*$_\mathcal{R}$ ABox, which expresses extensional information, and constitutes the *initial state* of the artifact. We call *proper constants* the constants $\mathcal{C}_{A_0}$ in A_0, and *labeled nulls* all new constants introduced by action execution.
- R is a set of *actions*, which change the state of the semantic artifact, i.e., the extensional information component.

An *action* ρ is constituted by a signature and an effect specification. The *action signature* is constituted by a name and a list of individual *input parameters*. Such parameters need to be substituted by constants for the execution of the action. Given a substitution θ for the input parameters, we denote by $\rho\theta$ the action with actual parameters.[3]

The *effect specification* consists of a set $\{e_1, \ldots, e_n\}$ of effects, which are assumed to take place simultaneously. Their formalization is inspired by the notion of mappings in data exchange [8]. Specifically, an *effect* e_i has the form $q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ where:

- $q_i^+ \wedge Q_i^-$ is a query over T and $\mathcal{C}_{A_0}$ with $\boldsymbol{x}$ as free variables, that may include some of the input parameters as query constants, q_i^+ is a UCQ, and Q_i^- is an arbitrary ECQ whose free variables are included in those of q_i^+, namely in $\boldsymbol{x}$. Intuitively, q_i^+ selects the tuples to instantiate the effect, and Q_i^- filters aways some of them.

[2] As customary, we can view each substitution simply as a tuple of constants, assuming some ordering of the free variables of q.

[3] We disregard a specific treatment of the output to the user, and assume instead that the user can freely pose queries to the ontology and thus extract implicit or explicit information from the states through which the semantic artifact evolves.

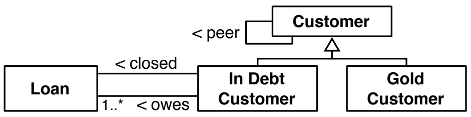


Fig. 1. A semantic artifact ontology

- A'_i is a set of facts over T, which include as constants: constants in $\mathcal{C}_{A_0}$, parameters, free variables of q_i^+, and implicitly existentially quantified variables.

Given a state A of $\mathcal{S}$, and a substitution σ for the parameters of the action ρ, the effect e_i extracts from A the set $ans_{\mathcal{C}_A}((q_i^+ \wedge Q_i^-)\sigma, T, A)$ of tuples of constants (proper constants and labeled nulls), and for each such tuple θ asserts a set $A'_i\sigma\theta$ of facts obtained from $A'_i\sigma$ by applying the substitution θ for the free variables of q_i^+. For each existentially quantified variable z in $A'_i\sigma$, a labeled null is introduced having the form $f_{z,e_i}(\boldsymbol{x})\sigma\theta$, where $f_{z,e_i}(\boldsymbol{x})$ is a Skolem function, depending on the existential variable z and the effect e_i, having as arguments the free variables $\boldsymbol{x}$ of q_i^+. We denote by $e_i\sigma(A)$ the overall set of facts, i.e., $e_i\sigma(A) = \bigcup_{\theta \in ans_{\mathcal{C}_A}(Q_i\sigma,T,A)} A'_i\sigma\theta$. The overall *effect* of the action ρ with parameter substitution σ over A is a new state $do(\rho\sigma, T, A) = \bigcup_{1 \leq i \leq n} e_i\sigma(A)$ for $\mathcal{S}$. Notice that $do(\rho\sigma, T, A)$ may be inconsistent wrt T. In this case, the action ρ with σ over A is *not executable*.

Let's make some observations on such actions. The effects of an action are naturally a form of update of the previous state, and not of belief revision [9]. That is, we never learn new facts on the state in which an action is executed, but only on the state resulting from the action execution. We also observe that existentially quantified variables introduced by actions effects are used as witnesses of values chosen by the external user/environment when executing the action. We assume that such a choice depends only on the specific effects and the information retrieved by the query in the premises. We model such a choice by introducing suitable labeled nulls generated by appropriate Skolem functions. Finally, we observe that we do not make any persistence (or frame) assumption in our formalization [10]. In principle at every move we substitute the whole old state, i.e., ABox, with a new one. On the other hand, it should be clear that we can easily write effect specifications that *copy* big chunks of the old state into the new one. For example, $P(x,y) \rightsquigarrow P(x,y)$ copies the entire set of assertions involving the role P.

Example 1. Let us consider a semantic artifact $\mathcal{S} = (T, A_0, R)$, where T is the *DL-Lite*$_\mathcal{R}$ formalization of the UML diagram in Figure 1, which can be described as follows. A bank considers two specific types of customers: in-debt customers have a loan with the bank, while gold customers have access to special privileges. In-debt customers may have closed their loan. A relationship $\mathsf{peer}(c,p)$ between two customers denotes that p can vouch for c. The initial state is $A_0 = \{\mathsf{Gold}(\mathsf{john}), \mathsf{Cust}(\mathsf{ann}), \mathsf{peer}(\mathsf{mark}, \mathsf{john})\}$. R includes the following actions (we use brackets to isolate UCQs in ECQs):

$$\mathsf{GetLoan}(c) : \{\ [\exists p.\mathsf{peer}(c,p) \wedge \mathsf{Gold}(p)] \rightsquigarrow \{\mathsf{owes}(c, \mathit{newl}(c))\}, \quad \mathit{CopyAll}\ \}$$

That is, customer c gets a loan provided that s/he has a peer that is "gold". We use $\mathit{CopyAll}$ as a shortcut to denote effects that copy all concepts and roles.

$$\mathsf{CloseAllLoans}(c) : \{\ [\mathsf{owes}(c,l)] \rightsquigarrow \{\mathsf{closed}(c,l)\}, \quad \mathit{CopyAll}\ \}$$

That is, customer c closes all his/her loans which are moved to the closed relation.

$$\begin{aligned}\mathsf{UpdateDebts} : \{ & [\mathsf{owes}(x,l)] \land \neg[\mathsf{closed}(x,l)] \rightsquigarrow \{\mathsf{owes}(x,l)\},\\ & [\mathsf{InDebt}(x)] \land \forall l.[\mathsf{owes}(x,l)] \supset [\mathsf{closed}(x,l)] \rightsquigarrow \{\mathsf{Cust}(x)\},\\ & \mathit{CopyAllExceptOwesClosedInDebt} \ \}\end{aligned}$$

That is, "remove" from owes those tuples that are in closed, and remove in-debt customers whose loans are all closed from InDebt, keeping them in Cust. *CopyAllExceptOwesClosedInDebt* copies everything except owes, closed, and InDebt. ■

4 Processes

Notice that semantic artifacts include information and actions to change such information. However, they do not say anything about how or when to apply a certain action. In other words, semantic artifact are agnostic to processes that use them. Processes over a semantic artifact $\mathcal{S} = (T, A_0, R)$ are (possibly nondeterministic) programs that use the state of $\mathcal{S}$ (accessed through T) to store their (intermediate and final) computation results, and the actions in R as atomic instructions. The state A can be arbitrarily queried through query answering over T, while it can be updated only through the actions in R. There are many ways to specify processes over $\mathcal{S}$. Here we adopt a rule-based specification.

A *condition/action rule* π for a semantic artifact $\mathcal{S}$ is an expression of the form $Q \mapsto \rho$, where ρ is an action in R and Q is an ECQ over T and $\mathcal{C}_{A_0}$, whose free variables are exactly the parameters of ρ. Such a rule expresses that, for each tuple θ for which condition Q holds, the action ρ with actual parameters θ *can* be executed.

A *process* is a finite set $\Pi = \{\pi_1, \ldots, \pi_n\}$ of rules. Notice that processes don't force the execution of actions but constrain them: the user of the process will be able to choose any of the actions that the rules forming the process allow. Notice also that our processes inherit entirely their states from the semantic artifact $\mathcal{S}$, see e.g., [1]. Other choices are also possible: the process could maintain its own state besides the one of the semantic artifact. As long as such an additional state is finite, or embeddable into the semantic artifact itself, the results here would easily extend to such a case.

The *execution* of a process Π over a semantic artifact $\mathcal{S}$ is defined as follows: we start from the initial state A_0 of the artifact, and for each rule $Q \mapsto \rho$ in Π, evaluate Q, and for each tuple θ returned execute $\rho\theta$, obtaining a new state $A' = do(\rho\theta, T, A_0)$ if A' is consistent wrt T (i.e., $\rho\theta$ is actually executable), and so on. In this way we build a *transition system* $\Upsilon(\Pi, \mathcal{S})$ whose states represent possible artifact states and where each transition represents the execution of an instantiated action that is allowed according to the process. $\Upsilon(\Pi, \mathcal{S})$ captures the behavior of the process Π over the artifact $\mathcal{S}$. In principle we can model-check such a transition system to verify dynamic properties. This is exactly what we are going to do next. However, one has to consider that in general such a transition system is infinite, so the classical results on model checking [2], which are developed for finite transition systems, do not apply.

Example 2. Referring to the example above, a process Π might include the following rules:

- $[\mathsf{Cust}(x)] \land \neg[\exists y.\mathsf{owes}(x,y)] \mapsto \mathsf{GetLoan}(x)$, i.e., a customer can get a loan if she does not have one already;
- $\exists y.([\mathsf{owes}(x,y)] \land \neg[\mathsf{closed}(x,y)]) \mapsto \mathsf{CloseAllLoans}(x)$, i.e., a customer that owes loans that are not closed, can close them all at once;
- $\mathsf{true} \mapsto \mathsf{UpdateDebts}$, i.e., it is always possible to perform UpdateDebts. ■

5 Verification Formalism

We turn to the verification formalism to be used to specify dynamic properties of processes running over semantic artifacts. Several choices are possible. Here we focus on a variant of μ-calculus [7], which is one of the most powerful temporal logics that subsumes both linear time logics, such as LTL and PSL, and branching time logics such as CTL and CTL* [2]. In particular, we introduce a variant of μ-calculus, called $\mu\mathcal{L}$ that conforms with the basic assumption of our formalism, namely the use of ECQs to talk about the semantic information contained in the state of the artifact. Formally, given a semantic artifact $\mathcal{S} = (T, A_0, R)$, formulas of $\mu\mathcal{L}$ over $\mathcal{S}$ have the following form:

$$\Phi ::= Q \mid \neg\Phi \mid \Phi_1 \land \Phi_2 \mid \exists x.\Phi \mid \Box\Phi \mid \Diamond\Phi \mid \mu Z.\Phi \mid \nu Z.\Phi \mid Z$$

where Q is an ECQ over T and $\mathcal{C}_{A_0}$ (but not labeled nulls), and Z is a predicate variable.

The symbols μ and ν can be considered as quantifiers, and we make use of the notions of scope, bound and free occurrences of variables, closed formulas, etc. The definitions of these notions are the same as in first-order logic, treating μ and ν as quantifiers. In fact, we are interested only in closed formulas as specification of temporal properties to verify. For formulas of the form $\mu Z.\Phi$ and $\nu Z.\Phi$, we require the *syntactic monotonicity* of Φ wrt Z: every occurrence of the variable Z in Φ must be within the scope of an even number of negation signs. In μ-calculus, given the requirement of syntactic monotonicity, the least fixpoint $\mu Z.\Phi$ and the greatest fixpoint $\nu Z.\Phi$ always exist. In order to define the meaning of such formulas we resort to transition systems. Let $\mathfrak{A} = \Upsilon(\Pi, \mathcal{S})$ be the transition system for a process Π over a semantic artifact $\mathcal{S} = (T, A_0, R)$. We denote by $\Sigma_{\mathfrak{A}}$ the states of $\mathfrak{A}$. Let $\mathcal{V}$ be a predicate and individual variable valuation on $\mathfrak{A}$, i.e., a mapping from the predicate variables Z to subsets of the states $\Sigma_{\mathfrak{A}}$, and from individual variables to constants in $\mathcal{C}_{A_0}$. Then, we assign meaning to μ-calculus formulas by associating to $\Upsilon(\Pi, \mathcal{S})$ and $\mathcal{V}$ an *extension function* $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$, which maps μ-calculus formulas to subsets of $\Sigma_{\mathfrak{A}}$. The extension function $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$ is defined inductively as follows:

$$\begin{array}{ll}
(Q)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid ans_{\mathcal{C}_{A_0}}(Q\mathcal{V}, T, A)\} \\
(Z)^{\mathfrak{A}}_{\mathcal{V}} & = Z\mathcal{V} \subseteq \Sigma_{\mathfrak{A}} \\
(\neg\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \Sigma_{\mathfrak{A}} - (\Phi)^{\mathfrak{A}}_{\mathcal{V}} \\
(\Phi_1 \land \Phi_2)^{\mathfrak{A}}_{\mathcal{V}} & = (\Phi_1)^{\mathfrak{A}}_{\mathcal{V}} \cap (\Phi_2)^{\mathfrak{A}}_{\mathcal{V}} \\
(\exists x.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{(\Phi)^{\mathfrak{A}}_{\mathcal{V}[x/c]} \mid c \in \mathcal{C}_{A_0}\} \\
(\Diamond\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid \exists A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ and } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\Box\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \{A \in \Sigma_{\mathfrak{A}} \mid \forall A'.\ A \Rightarrow_{\mathfrak{A}} A' \text{ implies } A' \in (\Phi)^{\mathfrak{A}}_{\mathcal{V}}\} \\
(\mu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcap\{\mathcal{E} \subseteq \Sigma_{\mathfrak{A}} \mid (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]} \subseteq \mathcal{E}\} \\
(\nu Z.\Phi)^{\mathfrak{A}}_{\mathcal{V}} & = \bigcup\{\mathcal{E} \subseteq \Sigma_{\mathfrak{A}} \mid \mathcal{E} \subseteq (\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]}\}
\end{array}$$

Here $A \Rightarrow_{\mathfrak{A}} A'$ holds iff there exists a rule $Q \mapsto \rho$ in Π such that there exists a $\theta \in ans_{\mathcal{C}_A}(Q, T, A)$ and $A' = do(\rho\theta, T, A')$. That is, there exist a rule in Π that can fire on A and produce an instantiated action $\rho\theta$, which applied on A has A' as effect.

Intuitively, the extension function $(\cdot)^{\mathfrak{A}}_{\mathcal{V}}$ assigns to the various μ-calculus constructs the following meanings. The boolean connectives have the expected meaning, while quantification is restricted to constants of $\mathcal{C}_{A_0}$, which are the only constants that the ECQs in the formula can retrieve. We also use the usual FOL abbreviations. The extension of $\Diamond\Phi$ consists of the states A such that for *some* state A' with $A \Rightarrow_{\mathfrak{A}} A'$, we have that Φ holds in A'. While the extension of $\Box\Phi$ consists of the states A such that for *all* states A' with $A \Rightarrow_{\mathfrak{A}} A'$, we have that Φ holds in A'. The extension of $\mu Z.\Phi$ is the *smallest subset* $\mathcal{E}_\mu$ of $\Sigma_{\mathfrak{A}}$ such that, assigning to Z the extension $\mathcal{E}_\mu$, the resulting extension of Φ is contained in $\mathcal{E}_\mu$. That is, the extension of $\mu X.\Phi$ is the *least fixpoint* of the operator $\lambda\mathcal{E}.(\Phi)^{\mathfrak{A}}_{\mathcal{V}[Z/\mathcal{E}]}$ (here $\mathcal{V}[Z/\mathcal{E}]$ denotes the predicate valuation obtained from $\mathcal{V}$ by forcing the valuation of Z to be $\mathcal{E}$). Similarly, the extension of $\nu X.\Phi$ is the *greatest subset* $\mathcal{E}_\nu$ of $\Sigma_{\mathfrak{A}}$ such that, assigning to X the extension $\mathcal{E}_\nu$, the resulting extension of Φ contains $\mathcal{E}_\nu$. That is, the extension of $\nu X.\Phi$ is the *greatest fixpoint* of the operator $\lambda\mathcal{E}.(\Phi)^{\mathfrak{A}}_{\mathcal{V}[X/\mathcal{E}]}$. When Φ is a closed formula, $(\Phi)^{\mathfrak{A}}_{\mathcal{V}}$ does not depend on $\mathcal{V}$, and we denote it by $\Phi^{\mathfrak{A}}$.

The reasoning problem we are interested in is *model checking*, i.e., verify whether a *$\mu\mathcal{L}$ closed formula Φ holds for the process Π over the semantic artifact $\mathcal{S}$*. Formally, such a problem is defined as checking whether $A_0 \in \Phi^{\mathfrak{A}}$, that is, whether Φ is true in the root of the initial state A_0 of transition system $\Upsilon(\Pi, \mathcal{S})$.

Example 3. Continuing on our running example, we consider the following simple safety property: It is always true that gold customers in A_0 remain so. This property can be written as:

$$\forall x.([\mathsf{Gold}(x)] \supset \nu Z.([\mathsf{Gold}(x)] \wedge \Box Z)).$$

This formula is true, since no action (among the ones above) removes individuals from being Gold.

Next, we consider a simple liveness property: It is possible to reach a state in which a gold customer is also an in-debt customer.

$$\mu Z.([\exists x.\mathsf{Gold}(x) \wedge \mathsf{InDebt}(x)] \vee \Diamond Z)$$

This formula is true, because the ontology implies that if x participates to owes then x is an instance of InDebt; and we can reach a state in which $\exists x.\mathsf{Gold}(x) \wedge \mathsf{owes}(x, y)$ holds by firing the action GetLoan(john), which is allowed by the process. ■

6 Homomorphism and Bisimulation

We want to understand when two ABoxes A_1 and A_2 over a common *DL-Lite*$_{\mathcal{R}}$ TBox T provide the same answers to all EQL queries. Given two relational structures $\mathcal{I}_1$ and $\mathcal{I}_2$ over the same set of relations, and a set $\mathcal{C}$ of constants, a *$\mathcal{C}$-homomorphism* h from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a mapping from the domain of $\mathcal{I}_1$ to the domain of $\mathcal{I}_2$ that preserves constants in $\mathcal{C}$ and relations, i.e., $h(c^{\mathcal{I}_1}) = c^{\mathcal{I}_2}$ for each $c \in \mathcal{C}$, and if $(d_1, \ldots, d_n) \in r^{\mathcal{I}_1}$, then $(h(d_1), \ldots, h(d_n)) \in r^{\mathcal{I}_2}$, for each relation r. Then, we say that there is a *$\mathcal{C}$-homomorphism from A_1 to A_2 wrt T*, denoted $A_1 \to^{\mathcal{C}}_{T} A_2$, iff there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to each model of (T, A_2), where $\mathcal{I}_{A_1}$ is the structure whose domain is $\mathcal{C}_{A_1}$, whose constants are interpreted as themselves, and $N^{\mathcal{I}_{A_1}} = \{c \mid N(c) \in A_1\}$ for each concept name N, similarly for role names. This property can be checked by resorting to query answering over ontologies. For the ABox A_1, let Q_{A_1} be the boolean conjunctive query obtained as the conjunction of all facts in A_1, in which the constants not in $\mathcal{C}$ are treated as existentially quantified variables.

Lemma 1. *Given a DL-Lite$_{\mathcal{R}}$ TBox T, two ABoxes A_1 and A_2 over T, and a set $\mathcal{C}$ of constants, we have that $A_1 \to^{\mathcal{C}}_{T} A_2$ iff $ans_{\mathcal{C}}(Q_{A_1}, T, A_2) = \mathsf{true}$.*

Proof (sketch). We remind that a *DL-Lite*$_{\mathcal{R}}$ ontology (T, A) has a unique (up to renaming of domain elements) canonical-model [5], which is the model that has a $\mathcal{C}_A$-homomorphism to each model of (T, A). By composing homomorphisms, we have that $A_1 \to_T^{\mathcal{C}} A_2$ iff there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to the canonical model of (T, A_2). The claim then follows by considering that there is a $\mathcal{C}$-homomorphism from $\mathcal{I}_{A_1}$ to the canonical model of (T, A_2) iff $ans_{\mathcal{C}}(Q_{A_1}, T, A_2) = \mathsf{true}$ [11,5]. □

A_1 and A_2 are said *$\mathcal{C}$-homomorphically equivalent wrt T* if $A_1 \to_T^{\mathcal{C}} A_2$ and $A_2 \to_T^{\mathcal{C}} A_1$.

Theorem 1. *Let $\mathcal{C}$ be a set of constants, and A_1, A_2 two ABoxes that are $\mathcal{C}$-homomorphically equivalent wrt a TBox T. Then, for every ECQ Q over T using only constants in $\mathcal{C}$, we have that $ans_{\mathcal{C}}(Q, T, A_1) = ans_{\mathcal{C}}(Q, T, A_2)$.*

Next we want to capture when two states of a single transition system or more generally of two transition systems, possibly obtained by applying different processes to different semantic artifacts sharing the same TBox and constants in the initial state, can be considered *behaviourally equivalent*, in the sense that they satisfy exactly the same $\mu\mathcal{L}$ formulas. To formally capture such an equivalence, we make use of the notion of *bisimulation* [12], suitably extended to deal with query answering over ontologies.

Given two artifact transition systems $\mathfrak{A}_1 = \Upsilon(\Pi_1, \mathcal{S}_1)$ (with states $\Sigma_{\mathfrak{A}_1}$) and $\mathfrak{A}_2 = \Upsilon(\Pi_2, \mathcal{S}_2)$ (with states $\Sigma_{\mathfrak{A}_2}$) such that $\mathcal{S}_1 = (T, A_{10}, R_1)$ and $\mathcal{S}_2 = (T, A_{20}, R_2)$ share the same TBox T and the same constants $\mathcal{C} = \mathcal{C}_{A_{10}} = \mathcal{C}_{A_{20}}$, a *bisimulation* is a relation $\mathcal{B} \subseteq \Sigma_{\mathfrak{A}_1} \times \Sigma_{\mathfrak{A}_2}$ such that: $(A_1, A_2) \in \mathcal{B}$ implies that:

1. A_1 and A_2 are $\mathcal{C}$-homomorphically equivalent wrt T;
2. if $A_1 \Rightarrow_{\mathfrak{A}_1} A_1'$ then there exists A_2' such that $A_2 \Rightarrow_{\mathfrak{A}_2} A_2'$ and $(A_1', A_2') \in \mathcal{B}$;
3. if $A_2 \Rightarrow_{\mathfrak{A}_2} A_2'$ then there exists A_1' such that $A_1 \Rightarrow_{\mathfrak{A}_1} A_1'$ and $(A_1', A_2') \in \mathcal{B}$.

We say that two states A_1 and A_2 are *bisimilar*, if there exists a bisimulation $\mathcal{B}$ such that $(A_1, A_2) \in \mathcal{B}$. Two transition systems $\mathfrak{A}_1$ with initial state A_{10} and $\mathfrak{A}_2$ with initial state A_{20} are *bisimilar* if $(A_{10}, A_{20}) \in \mathcal{B}$.

The following theorem states that the formula evaluation in $\mu\mathcal{L}$ is indeed invariant wrt bisimulation, so we can equivalently check any bisimilar transition systems.

Theorem 2. *Let $\mathfrak{A}_1$ and $\mathfrak{A}_2$ be two transition systems obtained from two semantic artifacts sharing the same TBox and constants. Then, for two states A_1 of $\mathfrak{A}_1$ and A_2 of $\mathfrak{A}_2$ (including the initial ones) are bisimilar iff for all $\mu\mathcal{L}$ closed formulas Φ over the two semantic artifacts, we have that $A_1 \in (\Phi)^{\mathfrak{A}_1}$ iff $A_2 \in (\Phi)^{\mathfrak{A}_2}$.*

Proof. The proof is analogous to the standard proof of bisimulation invariance of μ-calculus [7], though taking into account our specific definition of bisimulation, using Theorem 1 to guarantee that ECQs are evaluated identically over bisimilar states. □

7 Undecidability and Decidability

We now show that, not surprisingly, verification in the infinite state setting we considered is undecidable, but that it becomes decidable under some interesting conditions inspired by the recent literature on data exchange [8]. Our results rely on the possibility of building special semantic artifacts that we call "inflationary approximates". For such special artifacts there exists a tight correspondence between the application of an action and a step in the chase of a set of tuple-generating dependencies (TGDs) [13,8]

Given a semantic artifact $\mathcal{S} = (T, A_0, R)$, its *inflationary approximate* is the semantic artifact $\mathcal{S}^+ = (T^+, A_0, R^+)$ defined as follows. T^+ is obtained from T by dropping all assertions involving negation on the right-hand side, thus obtaining a TBox formed by positive inclusions only. R^+ is formed by one action specification ρ^+ for each action specification $\rho \in R$, where ρ^+ is obtained from ρ by:

- removing all input parameters from the signature – note that the variables in q_i^+ that used to be parameters in ρ, become free variables in ρ^+;
- substituting each effect $e_i : q_i^+ \wedge Q_i^- \rightsquigarrow A_i'$ with $e_i : q_i^+ \rightsquigarrow A_i'$ – note that we need to preserve the Skolem functions name in the transformation;
- adding effects to copy all concept and role names, namely adding an effect $N(x) \rightsquigarrow N(x)$ for each concept name N of T, and an effect $P(x, y) \rightsquigarrow P(x, y)$ for each role name P of T.

Observe that executing actions in $\mathcal{S}^+$ can never give rise to an inconsistency, since T^+ does not contain any negative information [5].

We also consider the *generic process $\Pi_\top$*, in which all condition/action rules have the trivially true condition. Hence, $\Pi_\top$ allows for executing every action at every step. With these notions in place, it is easy to prove that verification in this setting is undecidable.

Theorem 3. *$\mu\mathcal{L}$ model checking of processes over semantic artifacts is undecidable.*

Proof (sketch). We show that it is already undecidable to check, given a semantic artifact $\mathcal{S}_\emptyset^+ = (\emptyset, A_0, R)$, of the form of an inflationary approximate of an artifact with an empty TBox, and the transition system $\mathfrak{A} = \Upsilon(\Pi_\top, \mathcal{S}_\emptyset^+)$, whether $A_0 \in \mu Z(q \vee \Diamond Z)^{\mathfrak{A}}$, where q is a boolean UCQ. We observe that the set of all actions in $\mathcal{S}^+$ can be seen as a set of TGDs, indeed it suffices to consider one TGD for each disjunct in the UCQ on the left-hand side of an effect specification. So, we can reduce to the above model checking problem the problem of answering boolean UCQs in a relational database under a set of TGDs, which is undecidable [14] □

Next, we observe a notable property of the transition system $\Upsilon(\Pi_\top, \mathcal{S}^+)$ generated by the generic process $\Pi_\top$ over the inflationary approximate $\mathcal{S}^+$ of a semantic artifact $\mathcal{S}$. Namely, each $do(\rho^+, T, \cdot)$ is a monotonic operator. This implies that by repeatedly applying such operators starting from the ABox A_0 in $\mathcal{S}^+$, we get at the limit (possibly transfinite) a single ABox A_{max}, which is the *least fixpoint* of such operators taken collectively [15,16]. Such an ABox contains, every fact generated by repeatedly executing actions from the inflationary approximate $\mathcal{S}^+$, that is every state A^+ of $\Upsilon(\Pi_\top, \mathcal{S}^+)$ is such that A^+ is contained in A_{max}. More interestingly, we show next that A_{max} contains also every A generated by repeatedly executing actions from the original $\mathcal{S}$.

Lemma 2. *Let $\mathcal{S} = (T, A_0, R)$ be a semantic artifact and Π a process over $\mathcal{S}$. Then every state A of the transition system $\Upsilon(\Pi, \mathcal{S})$ is a subset of A_{max} defined as above.*

Proof (sketch). We can show by induction that, for every sequence of actions $\rho_1\theta_1, \rho_2\theta_2, \ldots, \rho_n\theta_n$ generated by the process Π starting from A_0, the resulting state $do(\rho_n\theta_n, T, do(\cdots do(\rho_2\theta_2, T, do(\rho_1\theta_1, T, A_0))\cdots))$ is a subset of the corresponding resulting state $do(\rho_n^+ T^+, do(\cdots do(\rho_2^+, T^+, do(\rho_1^+, T^+, A_0))\cdots))$ of the inflationary approx., which is a subset of A_{max}. □

In other words, A_{max} generated by the generic process $\Pi_\top$ running over the inflationary approximate $\mathcal{S}^+$ of a semantic artifact $\mathcal{S}$, bounds all states A that any process Π can generate by running on $\mathcal{S}$. Hence if for any reason A_{max} is finite, then the transition system $\Upsilon(\Pi, \mathcal{S})$ is *finite*. Hence, being model checking of finite transition system decidable (in fact polynomial in the size of the transition system), we get that also model checking of $\Upsilon(\Pi, \mathcal{S})$ is decidable.

To get finiteness guarantees on A_{max}, we take advantage of the correspondence between action execution and TGDs chase steps, as in the proof of Theorem 3. We build on this correspondence by further considering that in *DL-Lite*$_\mathcal{R}$, every UCQ q over a TBox can be rewritten as a new UCQ $rew(q)$ over the same alphabet, to be *evaluated* over the ABox considered as a relational database [5] (that is dropping the TBox).

In the literature for data exchange, several conditions that guarantee the finiteness of the chase of TGDs have been considered [17,18]. Here we focus on the original notion of *weakly-acyclic* TGDs [17]. Weak-acyclicity is a syntactic notion that involves the so-called *dependency graph* of the set of TGDs. Informally, a set D of TGDs is weakly-acyclic if there are no cycles in the dependency graph of D involving "existential" relation positions. The key property of *weakly-acyclic* TGDs is that chasing a relational database with them (i.e., applying them in all possible ways) generates a set of facts (a database) that is finite. See [17] for details.

Given a semantic artifact $\mathcal{S} = (T, A_0, R)$ and its inflationary approximate $\mathcal{S}^+ = (T^+, A_0, R^+)$, we define the set $E^+_\emptyset$ of effect specifications that includes an effect $rew(q_i^+) \rightsquigarrow A_i$ for each effect $q_i^+ \rightsquigarrow A_i$ of an action $\rho^+ \in R^+$. Notice that the set $E^+_\emptyset$ can be seen as a set of TGDs. We say that a semantic artifact $\mathcal{S}$ is *weakly-acyclic* if the set $E^+_\emptyset$ seen as a set of TGDs is weakly-acyclic. (Note that the semantic artifact in our example is trivially weakly-acyclic.)

Lemma 3. *Let $\mathcal{S} = (T, A_0, R)$ be a weakly-acyclic semantic artifact and S^+ its inflationary approximate. Then A_{max} computed as above for S^+ is finite.*

Proof (sketch). We have to show that starting from A_0 we get to the least fixpoint A_{max} of the $do(\rho^+, T, \cdot)$ operator in a finite number of steps. To do so, we follow the line of the proof of finiteness of chase of weakly-acyclic TGDs in [17], and show that the number of Skolem terms generated by the effects of actions is bounded by a polynomial in the size of A_0. Differently from [17], we cannot rely on the notion of homomorphism to stop firing actions, but have to use membership of the new set of facts in the previous ones. $\square$

As a direct consequence of Lemma 2 and Lemma 3, for weakly-acyclic semantic artifacts verification is decidable.

Theorem 4. *$\mu\mathcal{L}$ model checking of processes over weakly-acyclic semantic artifacts is decidable.*

Proof (sketch). The result follows by observing that every state generated by the execution of any process Π over a weakly-acyclic semantic artifact $\mathcal{S}$ is a subset of A_{max}, which by Lemma 3 is finite. Hence, we can apply a model checking procedure for μ-calculus on finite-state systems [7]. $\square$

Note that the proof of Theorem 4 is giving us a single exponential upper bound (in the size of A_0) for $\mu\mathcal{L}$ model checking involving weakly-acyclic semantic artifacts.

8 Conclusions

In this paper we have studied verification of processes over semantic artifacts. We obtain an interesting decidability result by relying on the notion of inflationary approximate, which allows for a connection with the theory of chase of TGDs in relational databases. We close by observing that while we fully used the ontology for querying the artifact state, we use it in a limited way when *updating* the state, namely only to guarantee consistency. Ontology update has its own semantic and computational difficulties, see e.g., [19], which our approach sidesteps. However, if one could introduce a suitable notion of inflationary approximate in that case, the approach presented here could be used to devise decidable cases.

References

1. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Bull. on Data Engineering **32**(3) (2009) 3–9
2. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. The MIT Press (1999)
3. Cangialosi, P., De Giacomo, G., De Masellis, R., Rosati, R.: Conjunctive artifact-centric services. In: Proc. of ICSOC 2010. Volume 6470 of LNCS., Springer (2010) 318–333
4. Damaggio, E., Deutsch, A., Vianu, V.: Artifact systems with data dependencies and arithmetic. In: Proc. of ICDT 2011. (2011) 66–77
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3) (2007) 385–429
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: Proc. of IJCAI 2007. (2007) 274–279
7. Emerson, E.A.: Automated temporal reasoning about reactive systems. In: Logics for Concurrency: Structure versus Automata. Volume 1043 of LNCS. Springer (1996) 41–101
8. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proc. of PODS 2005. (2005) 61–75
9. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Proc. of KR'91. (1991)
10. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press (2001)
11. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proc. of STOC'77. (1977) 77–90
12. Milner, R.: An algebraic definition of simulation between programs. In: Proc. of IJCAI'71. (1971) 481–489
13. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1995)
14. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. of ICALP'81. Volume 115 of LNCS., Springer (1981) 73–85
15. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific J. of Mathematics **5**(2) (1955) 285–309
16. Gurevich, Y., Shelah, S.: Fixed-point extensions of first order logic. Annals of Pure and Applied Logics **32** (1986) 265–280
17. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comp. Sci. **336**(1) (2005) 89–124
18. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. PVLDB **2**(1) (2009) 970–981
19. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL-Lite* knowledge bases. In: Proc. of ISWC 2010. Volume 6496 of LNCS., Springer (2010) 112–128

First-Order Expressibility Results for Queries over Inconsistent DL-Lite Knowledge Bases

Meghyn Bienvenu

CNRS & Université Paris Sud, France
meghyn@lri.fr

1 Introduction

In recent years, there has been growing interest in ontology-based data access, in which information in the ontology is used to derive additional answers to queries posed over instance data. The $DL\text{-}Lite$ family of description logics [3, 2]) is considered especially well-suited for such applications due to the fact that query answering can be performed by first incorporating the relevant information from the ontology into the query, and then posing the modified query to the bare data. This property, known as first-order rewritability, means that query answering over $DL\text{-}Lite$ ontologies has very low data complexity, which is considered key to scalability.

An important problem which arises in ontology-based data access is how to handle inconsistencies. This problem is especially relevant in an information integration setting where the data comes from multiple sources and one generally lacks the ability to modify the data so as to remove the inconsistency. In the database community, the related problem of querying databases which violate integrity constraints has been extensively studied (cf. [4] for a survey) under the name of consistent query answering. The standard approach is based on the notion of a repair, which is a database which satisfies the integrity constraints and is as similar as possible to the original database. Consistent answers are defined as those answers which hold in all repairs. A similar strategy can be used for description logics by replacing the integrity constraints with the ontology.

Consistent query answering for the $DL\text{-}Lite$ family of description logics was recently studied in [8, 7]. Unfortunately, the obtained complexity results are rather negative: consistent query answering is co-NP-hard in data complexity, even for instance queries and the simplest dialect $DL\text{-}Lite_{core}$. In the database community, negative results were also encountered, but this spurred a line of research [5, 6, 9] aimed at identifying cases where consistent query answering is feasible, and in particular, can be done using query rewriting. We propose to carry out a similar investigation for $DL\text{-}Lite$ ontologies, with the aim of better understanding the cases in which query rewriting can be profitably used. In this paper, we make some first steps towards this goal. Specifically, we formulate general conditions which can be used to prove that a consistent rewriting does or does not exist for a given $DL\text{-}Lite_{core}$ TBox and instance query.

The paper is organized as follows. After some preliminaries, we introduce in Sections 3 and 4 the problem of consistent query answering and some useful notions and terminology. Our main results are presented in Sections 4, 5, and 6, where we present general conditions which yield co-NP-hardness, first-order inexpressiblity, or first-order expressiblity of consistent instance checking in $DL\text{-}Lite_{core}$. Finally, in Section 7, we show that query rewriting is always possible if we adopt a previously studied alternative semantics. Note that proofs have been omitted for lack of space but can be found in [1].

2 Preliminaries

Syntax. $DL\text{-}Lite_{core}$ knowledge bases (KBs) are built up from a set $\mathsf{N_I}$ of constants, called *individuals*, a set $\mathsf{N_C}$ of unary predicates, called *atomic concepts*, and a set $\mathsf{N_R}$ binary predicates, called *atomic roles*. Complex concept and role expressions are constructed using the following syntax:

$$B \to A \mid \exists R \qquad C \to B \mid \neg B \qquad R \to P \mid P^-$$

where $A \in \mathsf{N_C}$ and $P \in \mathsf{N_R}$. Here B (resp. R) denotes a *basic concept* (resp. *basic role*), and C denotes a *general concept*. A *TBox* is a finite set of *inclusions* of the form $B \sqsubseteq C$ (with B, C as above). An *ABox* is a finite set of *assertions* of the form $B(a)$ ($B \in \mathsf{N_C}$) or $R(a,b)$ ($R \in \mathsf{N_R}$) where $a, b \in \mathsf{N_I}$. A KB consists of a TBox and an ABox.

Notational conventions We use $\mathsf{lhs}(\Gamma)$ (resp. $\mathsf{rhs}(\Gamma)$) to refer to the basic concept appearing on the left (resp. right) side of an inclusion Γ, e.g. $\mathsf{lhs}(\exists P \sqsubseteq \neg D) = \exists P$ and $\mathsf{rhs}(\exists P \sqsubseteq \neg D) = D$. We sometimes use R^- to mean P^- if $R = P \in \mathsf{N_R}$ and P if $R = P^-$, and write $R(a,b)$ to mean $P(a,b)$ if $R = P$ and $R(b,a)$ if $R = P^-$.

Semantics An *interpretation* is $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ maps each $a \in \mathsf{N_I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathsf{N_C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $P \in \mathsf{N_R}$ to $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is straightforwardly extended to general concepts and roles, e.g. $(\exists S)^{\mathcal{I}} = \{c \mid \exists d : (c,d) \in S^{\mathcal{I}}\}$. $\mathcal{I}$ satisfies $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$; it satisfies $A(a)$ (resp. $P(a,b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$). We write $\mathcal{I} \models \alpha$ if $\mathcal{I}$ satisfies inclusion/assertion α. $\mathcal{I}$ is a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$ and assertions in $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable/consistent* if it has a model; otherwise it is *unsatisfiable/inconsistent* ($\mathcal{K} \models \bot$). We say that $\mathcal{K}$ *entails* α, written $\mathcal{K} \models \alpha$, if every model of $\mathcal{K}$ is a model of α. The *closure of* $\mathcal{T}$, written $cl(\mathcal{T})$, consists of all inclusions which are entailed from $\mathcal{T}$. Given an ABox $\mathcal{A}$, we denote by $\mathcal{I}_{\mathcal{A}}$ the interpretation which has as its domain the individuals in $\mathcal{A}$ and which makes true precisely the assertions in $\mathcal{A}$.

Queries A *query* is a formula of first-order logic with equality (FOL), whose atoms are of the form $A(t)$, $P(t,t')$, or $t = t'$ with t, t' *terms*, i.e., variables or individuals. *Conjunctive queries* are queries which do not contain $\forall$, $\neg$, or $=$. *Instance queries* (IQs) are queries consisting of a single atom with no variables (i.e. ABox assertions). A *Boolean query* is a query with no free variables. For a Boolean query q, we write $\mathcal{I} \models q$ when q holds in the interpretation $\mathcal{I}$, and $\mathcal{K} \models q$ when $\mathcal{I} \models q$ for all models $\mathcal{I}$ of $\mathcal{K}$.

3 Consistent query answering

The most commonly used approach to query answering over inconsistent KBs is known as *consistent query answering* and relies on the notion of a repair:

Definition 1. *A* repair *of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is an inclusion-maximal subset $\mathcal{B}$ of $\mathcal{A}$ consistent with $\mathcal{T}$. We use $Rep(\mathcal{K})$ to denote the set of repairs of $\mathcal{K}$.*

Consistent query answering consists in performing standard query answering on each of the repairs and intersecting the answers. For Boolean queries, we have:

Definition 2. *A Boolean query q is said to be* consistently entailed *from $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{K} \models_{cons} q$, if $\mathcal{T}, \mathcal{B} \models q$ for every repair $\mathcal{B} \in Rep(\mathcal{K})$.*

Just as with standard query entailment, we can ask whether consistent query entailment can be tested by rewriting the query and evaluating it over the data.

Definition 3. *A first-order query q' is a* consistent rewriting *of a Boolean query q w.r.t. a TBox $\mathcal{T}$ if for every ABox $\mathcal{A}$, we have $\mathcal{T}, \mathcal{A} \models_{cons} q$ if and only if $\mathcal{I}_{\mathcal{A}} \models q'$.*

We illustrate the notion of consistent rewriting on an example.

Example 1. Consider the query $q = R(a, b)$ and the TBox $\mathcal{T} = \{\exists R \sqsubseteq \neg D, \exists R \sqsubseteq \neg\exists S^-, \exists R^- \sqsubseteq \neg B\}$. We claim $q' = R(a, b) \land \neg D(a) \land \neg \exists x S(x, a) \land \neg B(b)$ is a consistent rewriting of q w.r.t. $\mathcal{T}$. To see why, note that if a repair implies q, then it must contain $R(a, b)$. Moreover, if the ABox $\mathcal{A}$ contains any assertion that contradicts $R(a, b)$ then we can build a repair which does not contain $R(a, b)$. Thus, $R(a, b)$ is consistently entailed just in the case that $R(a, b) \in \mathcal{A}$ and there are no assertions in $\mathcal{A}$ which conflict with $R(a, b)$, which is precisely what q' states.

The method used in Example 1 can be generalized to show that a consistent rewriting exists for all role instance queries[1]. Unfortunately, the same is not true for concept IQs. Indeed, in [7], it was shown that consistent instance checking in DL-Lite$_{core}$ is co-NP-hard in data complexity. We present the reduction in the following example.

Example 2. Consider an instance $\varphi = c_1 \land \ldots \land c_m$ of UNSAT, where each c_i is a propositional clause. Let $v_1, \ldots, v_k$ be the propositional variables appearing in φ. We define the $DL\text{-}Lite_{core}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ as follows:

$$\begin{array}{lcl}
\mathcal{T} & = & \{\exists P^- \sqsubseteq \neg \exists N^-, \exists P \sqsubseteq \neg \exists U^-, \quad \exists N \sqsubseteq \neg \exists U^-, \exists U \sqsubseteq A\} \\
\mathcal{A} & = & \{U(a, c_i) \mid 1 \leq i \leq m\} \cup \{P(c_i, v_j) \mid v_j \in c_i\} \cup \{N(c_i, v_j) \mid \neg v_j \in c_i\}
\end{array}$$

It is not hard to verify that φ is unsatisfiable if and only if $\mathcal{K} \models_{cons} A(a)$. The basic idea is that, because of the inclusion $\exists P^- \sqsubseteq \neg \exists N^-$, each repair corresponds to a valuation of the variables, with v_j assigned true if it has an incoming P-edge in the repair.

The focus in this paper will be on distinguishing between hard and easy instances of the consistent query answering problem. More specifically, we will be interested in the problem of deciding for a given TBox and IQ whether a consistent rewriting exists.

4 Causes and conflicts

In formulating our results, it will be convenient to introduce some terminology for referring to assertions which participate in the entailment of another assertion or its negation.

[1] Obviously this is no longer the case if we consider a logic with role inclusions like $DL\text{-}Lite_{\mathcal{R}}$.

Definition 4. *Let α,β be assertions and Υ an inclusion. We say β* causes (or is a cause of) *α given Υ, written $\beta \xmapsto{\Upsilon} \alpha$, if $\{\Upsilon\}, \{\beta\} \models \alpha$. We say β* conflicts with (or is a conflict for) *α given Υ, written $\beta \overset{\Upsilon}{\bullet\!\!\longrightarrow} \alpha$, if $\Upsilon = B_1 \sqsubseteq \neg B_2$ and $\beta \models B_1(a)$ and $\alpha \models B_2(a)$ for some a. Sometimes we omit Υ if its identity is not relevant.*

The following straightforward proposition characterizes consistent instance checking in terms of causes and conflicts.

Proposition 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL\text{-}Lite_{core}$ KB and α an instance query. Then $\mathcal{K} \not\models_{cons} \alpha$ if and only if there exists a subset $\mathcal{A}' \subseteq \mathcal{A}$ which is consistent with $\mathcal{T}$ and such that for every $\beta \in \mathcal{A}$ which causes α (given some axiom in $cl(\mathcal{T})$), there is $\gamma \in \mathcal{A}'$ which conflicts with β (given some axiom in $cl(\mathcal{T})$).*

In other words, consistent instance checking comes down to deciding existence of a consistent subset of the ABox which contradicts all causes of the instance query.

We now introduce the notion of a cause-conflict chain. The intuition is as follows. Suppose that we have an assertion μ_0 in the ABox which causes the IQ α. Then to show $\mathcal{K} \not\models_{cons} \alpha$, Proposition 1 says we must select some assertion ρ_0 which conflicts with μ_0. But if ρ_0 conflicts with an assertion λ_1 which is a conflict of another cause μ_1, then this forces us to choose a different conflict ρ_1 for μ_1 which is consistent with ρ_0. The presence of ρ_1 may in turn attack a conflict of a third cause μ_3, leading us to select a conflict ρ_3 for μ_3, and so on.

Definition 5. *A* cause-conflict chain *(for TBox $\mathcal{T}$ and IQ α) is a sequence $\mu_0\rho_0\lambda_1\mu_1\rho_1$ $\lambda_2\mu_2\rho_2 \ldots \lambda_n\mu_n\rho_n\lambda_{n+1}\mu_{n+1}$ of distinct assertions, together with a sequence $\Upsilon_0\Gamma_0\Sigma_1$ $\Omega_1\Upsilon_1\Gamma_1\Sigma_2 \ldots \Omega_n\Upsilon_n\Gamma_n\Sigma_{n+1}\Omega_{n+1}\Upsilon_{n+1}$ of inclusions from $cl(\mathcal{T})$, which satisfy:*

- *for every i: $\mu_i \xmapsto{\Upsilon_i} \alpha$, $\mu_i \overset{\Gamma_i}{\bullet\!\!\longrightarrow} \rho_i$, $\rho_i \overset{\Sigma_{i+1}}{\bullet\!\!\longrightarrow} \lambda_{i+1}$, and $\mu_i \overset{\Omega_i}{\bullet\!\!\longrightarrow} \lambda_i$*
- *if $j < i$, then we do not have $\mu_i \bullet\!\!\longrightarrow \rho_j$*
- *$\{\rho_0, \rho_1, \ldots, \rho_n\}$ is consistent with $\mathcal{T}$*

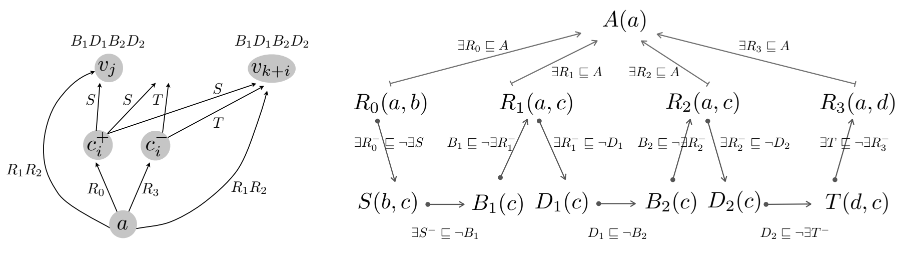
Examples of cause-conflict chains can be found in Figure 1a(b) and 2(b). In the following sections, we will consider particular types of cause-conflict chains and see how they are related to the presence of a consistent rewriting.

5 General co-NP-hardness result

In this section, we formulate a general condition which can be used to show co-NP-hardness of consistent instance checking. We begin by giving a more elaborate reduction from UNSAT, and then we analyze what is needed to make the proof go through.

Example 3. Consider the following TBox $\mathcal{T}$:

$$\{\exists R_0 \sqsubseteq A, \exists R_1 \sqsubseteq A, \exists R_2 \sqsubseteq A, \exists R_3 \sqsubseteq A, \exists R_0 \sqsubseteq \neg \exists S, \exists S^- \sqsubseteq \neg B_1, B_1 \sqsubseteq \neg \exists R_1^-,$$
$$\exists R_1^- \sqsubseteq \neg D_1, D_1 \sqsubseteq \neg B_2, B_2 \sqsubseteq \neg \exists R_2^-, \exists R_2^- \sqsubseteq \neg D_2, D_2 \sqsubseteq \neg \exists T^-, \exists T^- \sqsubseteq \neg \exists R_3^-\}$$



$B_1D_1B_2D_2$
v_j
$B_1D_1B_2D_2$
v_{k+i}
S
S
T
S
T
c_i^+
c_i^-
R_1R_2
R_0
R_3
R_1R_2
a
$A(a)$
$\exists R_0 \sqsubseteq A$
$\exists R_3 \sqsubseteq A$
$\exists R_1 \sqsubseteq A$
$\exists R_2 \sqsubseteq A$
$R_0(a,b)$
$R_1(a,c)$
$R_2(a,c)$
$R_3(a,d)$
$\exists R_0^- \sqsubseteq \neg\exists S$
$B_1 \sqsubseteq \neg\exists R_1^-$
$\exists R_1^- \sqsubseteq \neg D_1$
$B_2 \sqsubseteq \neg\exists R_2^-$
$\exists R_2^- \sqsubseteq \neg D_2$
$\exists T \sqsubseteq \neg\exists R_3^-$
$S(b,c)$
$B_1(c)$
$D_1(c)$
$B_2(c)$
$D_2(c)$
$T(d,c)$
$\exists S^- \sqsubseteq \neg B_1$
$D_1 \sqsubseteq \neg B_2$
$D_2 \sqsubseteq \neg\exists T^-$

(a) ABox (b) Cause-conflict chain

Fig. 1: ABox and type-1 cause-conflict chain for Example 3.

We show using a reduction from UNSAT that deciding whether $\mathcal{T}, \mathcal{A} \models_{cons} A(a)$ is co-NP-hard in data complexity. Given a propositional CNF $\varphi = c_1 \wedge \ldots \wedge c_m$ over $v_1, \ldots, v_k$, we define $\mathcal{A}$ as follows (see Figure 1(a) for a pictorial representation):

$$\begin{aligned}
&\{\, R_0(a, c_i^+), R_3(a, c_i^-) \mid 1 \leq i \leq m \,\} \cup \{R_1(a, v_j), R_2(a, v_j) \mid 1 \leq j \leq k+m\,\} \cup \\
&\{\, S(c_i^+, v_j) \mid v_j \in c_i\} \cup \{T(c_i^-, v_j) \mid \neg v_j \in c_i\} \cup \{\, S(c_i^+, v_{k+i}) \mid 1 \leq i \leq m\} \cup \\
&\{\, T(c_i^-, v_{k+i}) \mid 1 \leq i \leq m\} \cup \{B_1(v_j), B_2(v_j), D_1(v_j), D_2(v_j) \mid 1 \leq j \leq k+m\}
\end{aligned}$$

We show that $\varphi \models \bot$ if and only if $\mathcal{T}, \mathcal{A} \models_{cons} A(a)$. For the first direction, suppose we have a satisfying valuation for φ, and let V be the set of variables which are affected to true. We assume without loss of generality that if a variable v_j appears only positively (resp. negatively) in φ then $v_j \in V$ (resp. $v_j \notin V$). Define the subset $\mathcal{B}$ of $\mathcal{A}$ as follows:

$$\begin{aligned}
&\{\, S(c_i^+, v_j), D_1(v_j), D_2(v_j) \in \mathcal{A} \mid v_j \in V, 1 \leq j \leq k\} \cup \\
&\{\, T(c_i^-, v_j), B_1(v_j), B_2(v_j) \in \mathcal{A} \mid v_j \notin V, 1 \leq j \leq k\} \cup \\
&\{\, T(c_i^-, v_{k+i}), B_1(v_{k+i}), B_2(v_{k+i}) \in \mathcal{A} \mid \exists v_j \in V \text{ with } v_j \in c_i\} \cup \\
&\{\, S(c_i^+, v_{k+i}), D_1(v_{k+i}), D_2(v_{k+i}) \in \mathcal{A} \mid \forall v_j \in V : v_j \notin c_i\}
\end{aligned}$$

It is easy to check that $\mathcal{B}$ is consistent with $\mathcal{T}$ and that $\mathcal{T}, \mathcal{B} \not\models A(a)$. It can also be verified that adding any additional assertions from $\mathcal{A}$ to $\mathcal{B}$ leads to a contradiction. In particular, note that either a clause c_i has some positive variable $v_j \in V$, in which case $S(c_i^+, v_j), T(c_i^-, v_{k+i}) \in \mathcal{B}$, or it contains no such v_j, in which case $S(c_i^+, v_{k+i}), T(c_i^-, v_j) \in \mathcal{B}$. In either case, both $R_0(a, c_i^+)$ and $R_3(a, c_i^-)$ conflict with an assertion in $\mathcal{B}$. Thus, $\mathcal{B}$ is a repair of $\mathcal{A}$ w.r.t. $\mathcal{T}$ which does not entail $A(a)$.

For the other direction, let $\mathcal{B}$ be a repair with $\mathcal{T}, \mathcal{B} \not\models A(a)$. It follows that none of the role assertions in $\mathcal{A}$ involving R_0, R_1, R_2, R_3 appear in $\mathcal{B}$. The absence of R_1- and R_2-assertions and the consistency of $\mathcal{B}$ with $\mathcal{T}$ together imply that for each v_j, we have either B_1 and B_2 or both D_1 and D_2. This means each v_j has either incoming S-edges or incoming T-edges, but not both. We create a valuation in which v_j is affected to true if and only if v_j has an incoming S-edge. Clearly if c_i has a positive literal v_j which is affected to true, then it will be satisfied by this valuation. If instead all of the positive literals in c_i are affected to false, then the absence of $R_0(a, c_i^+)$ can only be explained by the presence in $\mathcal{B}$ of the assertion $S(c_i^+, v_{k+i})$. But this implies in turn the absence of $T(c_i^-, v_{k+i})$ in $\mathcal{B}$. As $R_3(a, c_i^-) \not\in \mathcal{B}$, there must be some assertion in $\mathcal{B}$ of the form $T(c_i^-, v_\ell)$ $(1 \leq \ell \leq k)$. This means v_ℓ will be affected to false by our valuation, and hence the clause will be satisfied. Thus, the formula φ is satisfiable.

To understand how the preceding reduction can be generalized, it is helpful to consider the cause-conflict chain pictured in Figure 1(b). This chain contains the essential structure used in the reduction, with individuals b, c, and d playing the roles of c_i^+, v_j, and c_ℓ^-. We first notice that at the start and end of the chain, there is a switch of individuals, which corresponds to moving from c_i^+ to v_j and then back to c_ℓ^-. Next remark that in order to show consistency of the constructed $\mathcal{B}$, we needed consistency of the sets of "forward" assertions $\{S(b,c), D_1(c), D_2(c)\}$ and "backward" assertions $\{B_1(c), B_2(c), T(d,c)\}$. Also note that in order to use a repair to construct a satisfying valuation, we had to prove that no v_j had both incoming S- and T-edges. This involved showing that the only way to simultaneously contradict all R_i assertions while retaining consistency was to choose all of the forward (D_i) or all of the backward (B_i) assertions. Key to this reasoning was the fact that for each $R_i(a, v_j)$ assertion, we were forced to choose either $B_i(v_j)$ or $D_i(v_j)$. If we could use some $B_\ell(v_j)$ or $D_\ell(v_j)$ with $\ell \neq j$, the line of reasoning fails. Finally we note that none of the conflicts in the chain involves the query individual a. This is important because if we used some assertion $C(a)$ to contradict $R_i(a, v_j)$, then we would also contradict $R_i(a, v_\ell)$ when $\ell \neq j$, making it impossible to independently choose truth values for each variable.

The preceding analysis leads us to define the notion of a position (to be able to talk about switching to a new individual) and the notion of type-1 cause-conflict chains.

Definition 6. *Concepts of the forms A or $\exists P$ (resp. $\exists P^-$) are said to have* position *1 (resp. 2). An inclusion Υ* begins (resp. concludes) *on position p, written* $\mathsf{bpos}(\Upsilon) = p$ *(resp.* $\mathsf{cpos}(\Upsilon) = p$*), if p is the position associated with* $\mathsf{lhs}(\Upsilon)$ *(resp.* $\mathsf{rhs}(\Upsilon)$*).*

Definition 7. *A cause-conflict chain for $\mathcal{T}$ and α defined by the sequence of assertions $\mu_0\rho_0\lambda_1\mu_1 \dots \rho_n\lambda_{n+1}\mu_{n+1}$ and sequence of inclusions $\Upsilon_0\Gamma_0\Sigma_1\Omega_1\Upsilon_1 \dots \Sigma_{n+1}\Omega_{n+1}\Upsilon_{n+1}$ is said to be of* type-1 *if it satisfies the following conditions:*

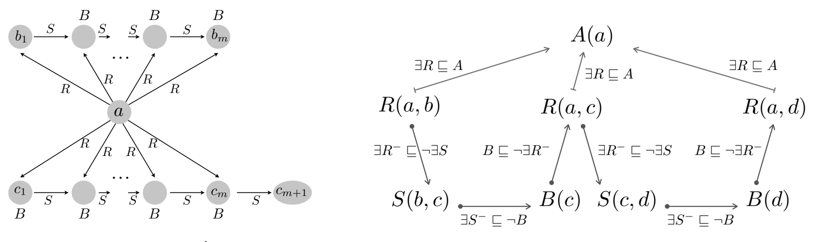
(C1) $\mathsf{bpos}(\Upsilon_i) \neq \mathsf{bpos}(\Gamma_i)$ *and* $\mathsf{bpos}(\Upsilon_i) \neq \mathsf{cpos}(\Omega_i)$ *for all i*
(C2) $\mathsf{cpos}(\Gamma_0) \neq \mathsf{bpos}(\Sigma_1)$ **(C4)** $\{\lambda_1, \dots, \lambda_{n+1}\}$ *is consistent with $\mathcal{T}$*
(C3) $\mathsf{cpos}(\Sigma_{n+1}) \neq \mathsf{bpos}(\Omega_{n+1})$ **(C5)** *if $j > i$, then we do not have $\mu_i \bullet\!\!\longrightarrow \lambda_j$*

Condition C1 of the definition states that the query individual is not used in the conflicts, whereas C2 and C3 make sure there is a switch to a new individual at the start and end of the chain. Condition C4 guarantees consistency of the "backward" conflict assertions, and C5 ensures that when reading the chain from right to left all causes are relevant (i.e. not already contradicted by one of the previous choices).

Example 4. If $B_1 \sqsubseteq \neg B_2$ were added to the TBox from Example 3, then the chain from Figure 1(b) would not be type-1, since $B_1(c)$ and $B_2(c)$ would conflict (violating C4).

The next result shows that the presence of a type-1 cause-conflict chain is sufficient to show co-NP-hardness (and *a fortiori*, the lack of a consistent rewriting). The proof generalizes the reduction from Example 3.

Theorem 1. *If a type-1 cause-conflict chain for $\mathcal{T}$ and α exists, then the problem of deciding whether $\mathcal{T}, \mathcal{A} \models_{cons} \alpha$ is co-NP-hard in data complexity.*



b_1
b_m
a
c_1
c_m
c_{m+1}
$A(a)$
$R(a,b)$
$R(a,c)$
$R(a,d)$
$S(b,c)$
$B(c)$
$S(c,d)$
$B(d)$
$\exists R \sqsubseteq A$
$\exists R^- \sqsubseteq \neg \exists S$
$B \sqsubseteq \neg \exists R^-$
$\exists S^- \sqsubseteq \neg B$

(a) ABox $\mathcal{A}_1$ (b) Cause-conflict chain

Fig. 2: ABox and Type-2 cause-conflict chain used in Example 5.

6 General first-order inexpressibility result

In this section, we use Ehrenfeucht-Fraïssé games to prove nonexistence of a consistent rewriting. As in the previous section, we start with an illustrative example, before formulating the general condition.

Example 5. Consider the following $DL\text{-}Lite_{core}$ TBox $\mathcal{T}$:

$$\mathcal{T} \quad = \quad \{\, \exists R \sqsubseteq A,\, \exists R^- \sqsubseteq \neg\, \exists S,\, \exists R^- \sqsubseteq \neg B, \exists S^- \sqsubseteq \neg B\,\}$$

We show using Ehrenfeucht-Fraïssé games that there is no consistent first-order rewriting of the query $A(a)$ w.r.t. $\mathcal{T}$. Consider some $k \in \mathbb{N}$, and let $m = 2^k + 1$. We construct two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows ($\mathcal{A}_1$ is pictured in Figure 2(a)):

$$\begin{aligned}
\mathcal{A}_1 \quad &= \quad \{\, R(a, b_i), R(a, c_i), B(c_i), S(c_i, c_{i+1}) \,|\, 1 \leq i \leq m\} \cup \\
&\qquad \{\, B(b_i) \,|\, 2 \leq i \leq m\,\} \,\cup\, \{\, S(b_i, b_{i+1}),\, |\, 1 \leq i \leq m-1\,\} \\
\mathcal{A}_2 \quad &= \quad \mathcal{A}_1 \setminus \{B(c_1)\} \cup \{B(b_1)\}
\end{aligned}$$

We show that $\mathcal{T}, \mathcal{A}_1 \models_{cons} A(a)$ and $\mathcal{T}, \mathcal{A}_2 \not\models_{cons} A(a)$. For the first point, suppose for a contradiction that there is a repair $\mathcal{B}$ of $\mathcal{A}_1$ w.r.t. $\mathcal{T}$ such that $\mathcal{T}, \mathcal{B} \not\models A(a)$. Then there can be no assertions in $\mathcal{B}$ of the form $R(a, b_i)$, and hence each such assertion must provoke a contradiction when added to $\mathcal{B}$. In order for $\mathcal{B} \cup \{R(a, b_1)\}$ to be inconsistent with $\mathcal{T}$, we must have $S(b_1, b_2) \in \mathcal{B}$, as $S(b_1, b_2)$ is the only assertion in $\mathcal{A}$ which conflicts with $R(a, b_1)$. But this means that $B(b_2) \notin \mathcal{B}$, and hence that $S(b_2, b_3) \in \mathcal{B}$, or else we could add $R(a, b_2)$ to $\mathcal{B}$ without provoking a contradiction. Continuing in this manner, we find that $S(b_{m-1}, b_m) \in \mathcal{B}$, and so $B(b_m) \notin \mathcal{B}$. But in this case, $\mathcal{B} \cup \{R(a, b_m)\}$ is consistent with $\mathcal{T}$, which contradicts the maximality of $\mathcal{B}$. For the second point, we remark that the set $\mathcal{B} = \{B(b_i), S(c_i, c_{i+1}) \mid 1 \leq i \leq m\}$ is a repair of $\mathcal{A}_2$ w.r.t. $\mathcal{T}$ such that $\mathcal{T}, \mathcal{B} \not\models A(a)$.

We now must show that duplicator has a k-round winning strategy in the Ehrenfeucht-Fraïssé game based on interpretations $\mathcal{I}_{\mathcal{A}_1}$ and $\mathcal{I}_{\mathcal{A}_2}$. The basic idea is as follows (we defer the full argument to [1]). Whenever spoiler selects a point which is "closer" to the side of b_m/c_{m+1} in $\mathcal{I}_{\mathcal{A}_1}$, duplicator responds with the identical point in $\mathcal{I}_{\mathcal{A}_2}$. When spoiler plays "closer" to the b_1/c_1 side, then duplicator plays c_i if b_i was played, and b_i if c_i was played. The important thing is to make sure there is sufficient distance between the indices j where duplicator copies spoiler and those where he chooses differently. This can be done by keeping track of the rightmost point where the choices differ and the leftmost point where they coincide and ensuring that the distance between these points is always at least 2^{k-i}, where i is the the current round of play.

Figure 2(b) presents a cause-conflict chain for the preceding example. Most of the conditions we identified in the previous section continue to hold for this chain. The only exception is that we do not have a switch of individuals at the end of the chain. Instead, we can remark that the initial cause-type is repeated further down the chain and can be contradicted in the same way, and this is what we use to create the long chain structure required in the proof. This leads us to define a second class of cause-conflict chains, in which we replace C3 with a new condition which captures this repetition.

Definition 8. *A cause-conflict chain for $\mathcal{T}$ and α whose sequence of inclusions is $\Upsilon_0\Gamma_0$ $\Sigma_1\Omega_1\Upsilon_1 \ldots \Sigma_{n+1}\Omega_{n+1}\Upsilon_{n+1}$ is said to be* type-2 *if it satisfies C1, C2, C4, C5, and C6:*

(C6) $\Upsilon_0 = \Upsilon_n$ *and* $\Gamma_0 = \Gamma_n$

The following theorem states that type-2 cause-conflict chains witness nonexistence of a consistent rewriting. The proof generalizes the argument outlined in Example 5.

Theorem 2. *If there exists a type-2 cause-conflict chain for $\mathcal{T}$ and α, then there is no consistent first-order rewriting for α w.r.t. $\mathcal{T}$.*

We next establish the relationship between type-1 and type-2 chains.

Theorem 3. *If there exists a type-1 cause-conflict chain for $\mathcal{T}$ and α, then there also exists a type-2 cause-conflict chain. The converse does not hold (assuming P≠NP).*

Proof (Sketch). For the first point, the idea to take a second copy of the type-1 chain, reverse it, and append it to the original. For the second point, we show that consistent instance checking for the TBox and IQ from Example 5 can be done in polynomial time by iteratively applying the following rule: if $R(a,c) \in \mathcal{A}$ and there is no $S(c,d) \in \mathcal{A}$, then delete all incoming S-edges to c. We continue until either we find $R(a,c) \in \mathcal{A}$ such that neither $B(c)$ nor any $S(c,d)$ belongs to $\mathcal{A}$ (in which case $A(a)$ is consistently entailed), or the rule is no longer applicable (and $A(a)$ is not consistently entailed).

7 Rewriting Procedure

In this section, we develop a procedure which is guaranteed to produce a consistent rewriting whenever the TBox $\mathcal{T}$ and query $\alpha = A(a)$ satisfy the following two criteria:

Ordering There exists a total order $<$ on $\mathsf{CauseT}(A)$ such that whenever a cause-conflict chain begins with inclusion $B_1 \sqsubseteq A$, ends with inclusion $B_2 \sqsubseteq A$, and satisfies conditions C1 and C3, we have $B_2 < B_1$.

No loops Every cause-conflict chain for $\mathcal{T}, \alpha$ of length $n{+}1$ which satisfies $\mathsf{cpos}(\Sigma_i) = \mathsf{bpos}(\Omega_i)$ for every $1 \leq i \leq n+1$ is such that $\Upsilon_i \neq \Upsilon_j$ for all $i \neq j < n+1$.

where $\mathsf{CauseT}(A) = \{D \mid D \sqsubseteq A \in cl(\mathcal{T})\}$ is the set of *cause-types* of A. We define the set of *conflict-types* of A analogously: $\mathsf{ConflT}(A) = \{D \mid D \sqsubseteq \neg A \in cl(\mathcal{T})\}$.

Algorithm 1 `Rewrite`

```
Input: TBox $\mathcal{T}$, IQ $A(a)$    Output: a first-order query $\varphi$
Initialize $\varphi$ to $\bot$ and initialize $\mathcal{G}$ to the set of all tuples $(\mathcal{C},\mathcal{D})$ which satisfy:
    (a) $\mathcal{C} = \{C \in \mathsf{CauseT}(A) \mid \exists D \in \mathcal{D} \text{ with } D \in \mathsf{ConflT}(C)\}$
    (b) for all $D \in \mathcal{D}$, there exists $C \in \mathcal{C}$ such that $D \in \mathsf{ConflT}(C)$
    (c) there do not exist $D_1, D_2 \in \mathcal{D}$ with $D_2 \in \mathsf{ConflT}(D_1)$
For every $(\mathcal{C},\mathcal{D}) \in \mathcal{G}$    // choose which cause-types to treat globally
    Let $\mathcal{D} = \{B_1,\dots,B_k,\exists P_1,\dots,\exists P_\ell,\exists P^-_{\ell+1},\dots,\exists P^-_m\}$ $(B_i \in \mathsf{N_C}, P_i \in \mathsf{N_R})$
    $S = \{B_i(a)\}_{i=1}^k \cup \{P_i(a,w_i)\}_{i=1}^\ell \cup \{P_i(w_i,a)\}_{i=\ell+1}^m$ // realize concepts in $\mathcal{D}$ at a
    // compute inequalities needed to ensure consistency (treating variables as individuals)
    $I = \{v_i \neq v_j \mid v_i, v_j \in \{a, w_1,\dots,w_m\} \text{ and } \mathcal{T}, S \cup \{v_i = v_j\} \models \bot\}$
    $U = \mathsf{CauseT}(A) \setminus \mathcal{C}$    // cause-types not yet treated
    $\varphi = \varphi \lor \exists w_1 \dots w_m \bigwedge_{\beta \in S} \beta \land \bigwedge_{\gamma \in I} \gamma \land \bigwedge_{C \in U} (\forall x\, \mathtt{auxRewrite}(\mathcal{T}, A(a), C, x, S))$
Output $\neg\varphi$
```

Our algorithm `Rewrite` creates a big disjunction, where each disjunct corresponds to a choice of a set of cause-types to be conflicted *globally*, i.e. one single assertion involving the query individual is used to conflict all causes of that type. For each disjunct, we first fix the assertions which realize these global conflicts, and then invoke subroutine `auxRewrite` to build one conjunct per untreated cause-type whose purpose is to see whether for each cause of that type there is an assertion which conflicts with it and can safely be added to the repair under construction. These conjuncts have a tree-like structure whose "paths" are cause-conflict chains which satisfy $\mathsf{cpos}(\varSigma_i) = \mathsf{bpos}(\varOmega_i)$ for all i. Property **No Loops** can thus be applied to show that the recursion depth of `auxRewrite` is no more than $|\mathsf{CauseT}(A)| + 1$, ensuring termination. The difficult step in the correctness proof is to show $\mathcal{I}_\mathcal{A} \not\models$ `Rewrite`$(\mathcal{T}, q)$ implies $\mathcal{T}, \mathcal{A} \not\models_{cons} q$. The basic idea is to use the way the negation of the formula is satisfied to direct our construction of a repair which conflicts with every cause of q. **Ordering** is used to decide in which order we should treat the causes. We illustrate this idea on a concrete example:

Example 6. Let $q = A(a)$ and $\mathcal{T}$ be the following TBox:

$$\{\exists R_0 \sqsubseteq A, \exists R_1 \sqsubseteq A, \exists R_2 \sqsubseteq A, \exists R_0^- \sqsubseteq \neg \exists S, \exists S^- \sqsubseteq \neg B_1, B_1 \sqsubseteq \neg \exists R_1^-,$$
$$\exists R_1^- \sqsubseteq \neg D_1, D_1 \sqsubseteq \neg \exists T^-, B_1 \sqsubseteq \neg \exists T^-, \exists T \sqsubseteq \neg \exists R_2^- \}$$

It can be verified that the negation of `Rewrite`$(\mathcal{T}, q)$ consists of a single disjunct:

$$\begin{aligned} &\forall x\, R_0(a,x) \rightarrow \exists y (S(x,y) \land (R_1(a,y) \rightarrow D_1(y))) \\ \land \quad &\forall x\, R_1(a,x) \rightarrow (B_1(x) \lor D_1(x)) \\ \land \quad &\forall x\, R_2(a,x) \rightarrow \exists y (T(x,y) \land \neg R_1(a,y)) \end{aligned}$$

We show that if this formula is satisfied in $\mathcal{I}_\mathcal{A}$, then we can construct a repair $\mathcal{B}$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ which does not entail $A(a)$. First we fix an order on $\mathsf{CauseT}(A)$ satisfying the conditions in **Ordering**: $\exists R_0 < \exists R_2 < \exists R_1$. This means we start by considering causes via $\exists R_0$. If $R_0(a,b) \in \mathcal{A}$, then the first conjunct allows us to find c such that $S(b,c) \in \mathcal{A}$ and $R_1(a,c) \in \mathcal{A}$ implies $D_1(c) \in \mathcal{A}$. We add $S(b,c)$ to $\mathcal{B}$, and also add $D_1(c)$ if $R_1(a,c) \in \mathcal{A}$. We then move on to the next cause-type in the order, $\exists R_2$. If we have $R_2(a,b) \in \mathcal{A}$, then we use the third conjunct to find c such that $T(b,c) \in \mathcal{A}$

Algorithm 2 `auxRewrite`

```
Input: TBox $\mathcal{T}$, IQ $A(a)$, $C \in \mathsf{CauseT}(A)$, variable $x$, $S$ set of atoms
Output: a first-order query $\chi$
If $C \in \mathsf{N_C}$, output $\neg C(a)$
Set $\alpha = R(a,x)$, $\chi = \neg\alpha$, and $B = \exists R^-$ where $C = \exists R$   // R basic role
For each $D \in \mathsf{ConflT}(B)$   // Consider different ways to contradict $\alpha$ on $x$
    Set $\beta = D(x)$ if $D \in \mathsf{N_C}$ and $\beta = T(x,y)$ [$y$ fresh variable] if $D = \exists T$
    If $\beta$ is necessarily inconsistent with $S$ given $\mathcal{T}$, exit the for-loop
    Else, let $\epsilon$ be the inequalities needed to ensure $\{\beta\} \cup S$ is consistent with $\mathcal{T}$
    // Compute untreated causes which are affected by choice of $\beta$
    Initialize $\Delta$ to $\emptyset$
    For all $\exists V \in \mathsf{CauseT}(A)$ such that $\mathcal{T}, S \cup \{\beta\} \cup \{V(a,x)\} \not\models \bot$ and
    $\mathsf{ConflT}(\exists V^-) \cap \mathsf{ConflT}(D) \neq \emptyset$
        Add $(\exists V, x)$ to $\Delta$   // need to find conflict for cause $V(a,x)$
    If $D = \exists T$, then for all $\exists V \in \mathsf{CauseT}(A)$ with $\mathcal{T}, S \cup \{\beta\} \cup \{V(a,y)\} \not\models \bot$
    and $\mathsf{ConflT}(\exists V^-) \cap \mathsf{ConflT}(\exists T^-) \neq \emptyset$
        Add $(\exists V, y)$ to $\Delta$   // need to find conflict for cause $V(a,y)$
    $\chi = \chi \vee (\exists y)(\beta \wedge \epsilon \wedge \bigwedge_{(H,v)\in\Delta} \mathtt{auxRewrite}(\mathcal{T}, A(a), H, v, S \cup \{\beta\}))$
Output $\chi$
```

and $R_1(a,c) \notin \mathcal{A}$, and we add $T(b,c)$ to $\mathcal{B}$. Finally we turn to the final cause-type $\exists R_1$, and let $R_1(a,b) \in \mathcal{A}$. Possibly we have already added $D_1(b)$ when dealing with the first conjunct, in which case we do nothing. Otherwise, because of the second conjunct, we have either $B_1(b) \in \mathcal{A}$ or $D_1(b) \in \mathcal{A}$, which we can add to $\mathcal{B}$. The set $\mathcal{B}$ is still consistent with $\mathcal{T}$ after this step, since if $T(e,b) \in \mathcal{B}$ then we would have $R_1(a,b) \notin \mathcal{A}$, and if $S(e,b) \in \mathcal{B}$, then we would have already added a conflict for $R_1(a,b)$. We have thus found a set $\mathcal{B}$ which is consistent with $\mathcal{T}$ and contradicts every assertion which could cause entailment of $A(a)$. By Proposition 1, we have $\mathcal{T}, \mathcal{A} \not\models_{cons} A(a)$.

Theorem 4. *If a TBox $\mathcal{T}$ and IQ q satisfy conditions **Ordering** and **No Loops**, then* `Rewrite`*($\mathcal{T}$, q) terminates and outputs a consistent rewriting of q w.r.t. $\mathcal{T}$.*

Theorem 4 can be used to derive simpler sufficient conditions, like the following:

Corollary 1. `Rewrite`*($\mathcal{T}$, $A(a)$) terminates with the correct output if there do not exist basic roles R, S with $\mathcal{T} \models \exists R \sqsubseteq A$ and $\mathcal{T} \models \exists R^- \sqsubseteq \neg \exists S$.*

8 Approximating Consistent Query Answering

In order to obtain a more generally applicable positive result, we consider a sound approximation of consistent query answering, which we term *cautious query answering*.

Definition 9. *A query q is* cautiously entailed *by a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{K} \models_{caut} q$, if $\mathcal{T}, \cap_{\mathcal{B} \in Rep(\mathcal{K})} \mathcal{B} \models q$.*

In [7], cautious conjunctive query answering (there called Intersection ABox Repair semantics) was shown to be tractable for $DL\text{-}Lite_{\mathcal{R}}$ KBs. The proposed algorithm first deletes all assertions involved in some conflict, and then queries the resulting ABox. It was left open whether query rewriting techniques could be used instead. We answer this question in the affirmative and thus obtain an improved upper bound of AC_0.

Theorem 5. *Cautious conjunctive query answering is in AC_0 for $DL\text{-}Lite_{core}$.*

Proof (Sketch). Given a $DL\text{-}Lite_{core}$ TBox $\mathcal{T}$ and a CQ q, we first compute (in the standard manner) a UCQ $q' = q_1 \vee ... \vee q_n$ such that for all ABoxes $\mathcal{A}$, we have $\mathcal{T}, \mathcal{A} \models q$ if and only if $\mathcal{I}_{\mathcal{A}} \models q'$. Then to each disjunct we add the negation of each atomic query which could contradict one of the atoms in the disjunct.

Example 7. If $q = \exists y\, B(x) \wedge R(x, y)$ and $\mathcal{T} = \{A \sqsubseteq B, A \sqsubseteq \exists R, B \sqsubseteq \neg D, \exists R^- \sqsubseteq \neg\exists S^-\}$, standard rewriting yields $A(x) \vee \exists y\, B(x) \wedge R(x, y)$. We then add $\neg\exists z S(z, y)$ to the second disjunct and $\neg D(x)$ to both to obtain the cautious rewriting.

Theorem 5 is easily extended to other $DL\text{-}Lite$ logics enjoying FO-rewritability.

9 Conclusion and Future Work

In this paper, we took a closer look at the problem of consistent instance checking in $DL\text{-}Lite$ and identified some general conditions which can be used to prove the absence or existence of a consistent rewriting. While our results were formulated for $DL\text{-}Lite_{core}$, we expect they can be easily lifted to more expressive $DL\text{-}Lite$ dialects.

The main objective for future work is to strengthen our results so as to be able to decide for every TBox and instance query whether a consistent rewriting exists. We conjecture that the absence of a type-2 cause-conflict chain is both a necessary and sufficient condition for existence of a consistent rewriting. Extending our investigation to conjunctive queries would be interesting but quite challenging, as it would likely involve confronting longstanding open problems from the database community, where a full characterization of rewritable cases remains elusive [9].

References

1. `http://www.lri.fr/~meghyn/BienvenuDL11-long.pdf`.
2. Allessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.
3. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
4. Jan Chomicki. Consistent query answering: Five easy pieces. In *Proc. of ICDT*, pages 1–17, 2007.
5. Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. In *Proc. of ICDT*, pages 337–351, 2005.
6. Luca Grieco, Domenico Lembo, Riccardo Rosati, and Marco Ruzzi. Consistent query answering under key and exclusion dependencies: algorithms and experiments. In *Proc. of CIKM*, pages 792–799, 2005.
7. Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of RR (Web Reasoning and Rule Systems)*, pages 103–117, 2010.
8. Domenico Lembo and Marco Ruzzi. Consistent query answering over description logic ontologies. In *Proc. of RR (Web Reasoning and Rule Systems)*, pages 194–208, 2007.
9. Jef Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *Proc. of PODS*, pages 179–190, 2010.

Fuzzy Ontologies over Lattices with T-norms

Stefan Borgwardt and Rafael Peñaloza

Theoretical Computer Science, TU Dresden, Germany
{stefborg,penaloza}@tcs.inf.tu-dresden.de

1 Introduction

In some knowledge domains, a correct handling of vagueness and imprecision is fundamental for adequate knowledge representation and reasoning. For example, when trying to diagnose a disease, medical experts need to confront symptoms described by the patient, which are by definition subjective, and hence vague. Moreover, a single malady may present a diversity of clinical manifestations in different patients, which leads to imprecise (partial) diagnoses.

Fuzzy logic [15] is a prominent approach for dealing with imprecise knowledge. It is based on the notion of fuzzy sets [25], where elements are assigned a membership degree from the real interval $[0,1]$. So-called t-norms are used to define the interpretation of the logical connectives. The notion of membership degrees and the operators used can be generalized to lattices, giving rise to L-fuzzy sets [13] and lattice-based t-norms [26, 12].

During the last two decades, several fuzzy DLs have been defined by enriching classical DLs first with fuzzy set semantics [24, 20, 19] and then t-norms [16, 7, 11]. Attempts have also been made at using L-fuzzy set semantics [21, 17]. However, all these approaches either disregard the terminological knowledge, or allow only for a limited class of TBoxes. In fact, it is still unknown whether standard reasoning in fuzzy DLs with general TBoxes is decidable [5, 3]. To the best of our knowledge, the only approaches capable of dealing with full fuzzy TBoxes are based on a finite total order with the Łukasiewicz t-norm [6, 8] or finite De Morgan lattices with the minimum t-norm [9].

In this paper we introduce the lattice-based fuzzy DL $\mathcal{ALC}_L$, where L is a complete De Morgan lattice equipped with a t-norm operator. We show that satisfiability in this logic is undecidable if L is infinite. Undecidability holds even if L is a countable, residuated total order. On the other hand, if L is finite, then satisfiability becomes decidable and, under some conditions on the lattice and the t-norm, ExpTime-complete, i.e. not harder than satisfiability in crisp $\mathcal{ALC}$.

Our reasoning procedure is in fact general enough to handle any kind of truth-functional semantics, as long as the functions defining the connectives are computable.

2 Lattices

We now give a brief introduction to lattices and t-norms. For a more comprehensive description of these notions, see e.g. [14, 12].

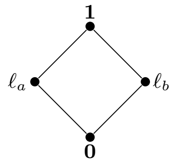


Fig. 1. The De Morgan lattice L_2 with $\sim \ell_a = \ell_a$ and $\sim \ell_b = \ell_b$. This lattice was first considered by Belnap [4] for reasoning with incomplete and inconsistent knowledge.

A *lattice* is an algebraic structure $(L, \vee, \wedge)$ over a *carrier set* L with two binary operations supremum $\vee$ and infimum $\wedge$ that are idempotent, associative, and commutative and satisfy the absorption laws $\ell \vee (\ell \wedge m) = \ell = \ell \wedge (\ell \vee m)$ for all $\ell, m \in L$. The order $\leq$ on L is defined by $\ell \leq m$ iff $\ell \wedge m = \ell$ for all $\ell, m \in L$. A lattice is *distributive* if $\vee$ and $\wedge$ distribute over each other, *finite* if L is finite, and *bounded* if it has a *minimum* and a *maximum* element, denoted as $\mathbf{0}$ and $\mathbf{1}$, respectively. It is *complete* if suprema and infima of arbitrary subsets $T \subseteq L$ exist; these are denoted by $\bigvee_{t \in T} t$ and $\bigwedge_{t \in T} t$, respectively. Notice that every finite lattice is also bounded and complete. Whenever it is clear from the context, we will simply use the carrier set L to represent the lattice $(L, \vee, \wedge)$.

A *De Morgan lattice* is a bounded distributive lattice extended with an involutive and anti-monotonic unary operation $\sim$, called *(De Morgan) negation*, satisfying the De Morgan laws $\sim(\ell \vee m) = \sim \ell \wedge \sim m$ and $\sim(\ell \wedge m) = \sim \ell \vee \sim m$ for all $\ell, m \in L$. Figure 1 shows a simple De Morgan lattice.

In fuzzy logics, conjunctions and disjunctions are interpreted with the help of t-norms and t-conorms. Given a De Morgan lattice L, a *t-norm on* L is an associative and commutative binary operator $\otimes : L \times L \to L$ which has the unit $\mathbf{1}$, and is monotonic in both arguments. Given a t-norm $\otimes$, its associated *t-conorm* $\oplus$ is constructed using the negation as follows: $\ell \oplus m := \sim(\sim \ell \otimes \sim m)$. For example, the infimum operator $\ell \otimes m := \ell \wedge m$ defines a t-norm; its associated t-conorm is then given by $\ell \oplus m := \ell \vee m$.

Another important operator is the residuum, which is used for interpreting implications in the logic. The *residuum* of a t-norm $\otimes$ on a complete lattice L is the binary operator $\Rightarrow$ defined by $\ell \Rightarrow m := \bigvee \{x \mid \ell \otimes x \leq m\}$. If $\ell \otimes (\ell \Rightarrow m) \leq m$ for all $\ell, m \in L$ (that is, if the supremum in the definition of residuum is always a maximum), then $\otimes$ is called *residuated* and L a *residuated lattice*.[1]

In the following we will use two important properties of the residuum: for every $\ell, m \in L$, (i) $\mathbf{1} \Rightarrow \ell = \ell$, and (ii) if $\ell \leq m$, then $\ell \Rightarrow m = \mathbf{1}$. Additionally, if $\otimes$ is residuated, then $\ell \Rightarrow m = \mathbf{1}$ implies that $\ell \leq m$.

In the next section, we describe the multi-valued description logic $\mathcal{ALC}_L$, whose semantics uses the residuum $\Rightarrow$ and the De Morgan negation $\sim$. We emphasize, however, that the reasoning algorithm presented in Section 5 can be used with any choice of operators, as long as these are computable. In particular this means that our algorithm could also deal with other variants of multi-valued semantics, e.g. [9, 21].

[1] Residua are usually only defined for residuated lattices. However, as $\ell \Rightarrow m$ is well-defined for t-norms on complete De Morgan lattices, we remove this restriction.

3 The Fuzzy Logic $\mathcal{ALC}_L$

In the following, we will assume that L is a complete De Morgan lattice and $\otimes$ is a t-norm on L. The multi-valued description logic $\mathcal{ALC}_L$ is a generalization of the crisp DL $\mathcal{ALC}$ that allows the use of the elements of a complete De Morgan lattice as truth values, instead of just the Boolean values *true* and *false*. The syntax of concept descriptions in $\mathcal{ALC}_L$ is the same as in $\mathcal{ALC}$; that is, $\mathcal{ALC}_L$ concept descriptions are built from a set of concept names and role names through the constructors $\sqcap, \sqcup, \neg, \top, \bot, \exists$ and $\forall$.

The semantics of this logic is based on interpretation functions that not simply describe whether an element of the domain belongs to a concept or not, but give a lattice value describing the membership degree of the element to this concept; more formally, the semantics is based on L-fuzzy sets.

Definition 1 (semantics of $\mathcal{ALC}_L$). *An* interpretation *is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty (crisp)* domain *and $\cdot^\mathcal{I}$ is a function that assigns to every concept name A and every role name r functions $A^\mathcal{I} : \Delta^\mathcal{I} \to L$ and $r^\mathcal{I} : \Delta^\mathcal{I} \times \Delta^\mathcal{I} \to L$, respectively. The function $\cdot^\mathcal{I}$ is extended to $\mathcal{ALC}_L$ concept descriptions as follows for every $x \in \Delta^\mathcal{I}$:*

- $\top^\mathcal{I}(x) = \mathbf{1},\ \bot^\mathcal{I}(x) = \mathbf{0}$,
- $(C \sqcap D)^\mathcal{I}(x) = C^\mathcal{I}(x) \otimes D^\mathcal{I}(x),\ \ (C \sqcup D)^\mathcal{I}(x) = C^\mathcal{I}(x) \oplus D^\mathcal{I}(x)$,
- $(\neg C)^\mathcal{I}(x) = \sim C^\mathcal{I}(x)$,
- $(\exists r.C)^\mathcal{I}(x) = \bigvee_{y \in \Delta^\mathcal{I}} r^\mathcal{I}(x,y) \otimes C^\mathcal{I}(y)$,
- $(\forall r.C)^\mathcal{I}(x) = \bigwedge_{y \in \Delta^\mathcal{I}} r^\mathcal{I}(x,y) \Rightarrow C^\mathcal{I}(y)$.

Notice that, unlike crisp $\mathcal{ALC}$, the existential and universal quantifiers are not dual to each other, i.e. in general $\neg\exists r.C$ and $\forall r.\neg C$ have different semantics.

The axioms in a TBox are also associated to a lattice value, allowing for a general notion of subsumption between concepts that is based on the residuum.

Definition 2 (TBox). *A* TBox *is a finite set of* (labeled) general concept inclusions *(GCIs) of the form $\langle C \sqsubseteq D, \ell \rangle$, where C, D are $\mathcal{ALC}_L$ concept descriptions and $\ell \in L$.*

An interpretation $\mathcal{I}$ satisfies *a GCI $\langle C \sqsubseteq D, \ell \rangle$ if $\bigwedge_{x \in \Delta^\mathcal{I}} C^\mathcal{I}(x) \Rightarrow D^\mathcal{I}(x) \geq \ell$. $\mathcal{I}$ is called a* model *of the TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$.*

We emphasize here that $\mathcal{ALC}$ is a special case of $\mathcal{ALC}_L$, where the underlying lattice contains only the elements $\mathbf{0}$ and $\mathbf{1}$, which may be interpreted as *false* and *true*, respectively, and the t-norm and t-conorm are just conjunction and disjunction, respectively. Accordingly, one can think of generalizing the reasoning problems for $\mathcal{ALC}$ to the use of other lattices. We will focus on the problem of deciding satisfiability of a concept. We are further interested in computing the highest degree with which an individual may belong to a concept.

Definition 3 (satisfiability). *Let C, D be $\mathcal{ALC}_L$ concept descriptions, $\mathcal{T}$ a TBox and $\ell \in L$. C is ℓ-*satisfiable *w.r.t. $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $\bigvee_{x \in \Delta^\mathcal{I}} C^\mathcal{I}(x) \geq \ell$. The* best satisfiability degree *for C w.r.t. $\mathcal{T}$ is the largest ℓ such that C is ℓ-satisfiable w.r.t. $\mathcal{T}$.*

Notice that if C is ℓ-satisfiable and ℓ'-satisfiable w.r.t. $\mathcal{T}$, then C is also $\ell \vee \ell'$-satisfiable. Hence, the notion of the best satisfiability degree is well defined.

In some cases, however, this definition of satisfiability turns out to be too weak, since a concept C may be ℓ-satisfiable even if no element of the domain may ever belong to C with a value $\geq \ell$. Consider the following example.

Example 4. We use the lattice L_2 from Figure 1 with t-norm $\ell \otimes \ell' := \ell \wedge \ell'$ and the TBox $\mathcal{T} = \{\langle \top \sqsubseteq (A \sqcap \neg A) \sqcup (B \sqcap \neg B), \mathbf{1} \rangle\}$. The concept A is $\mathbf{1}$-satisfiable w.r.t. $\mathcal{T}$ since the interpretation $\mathcal{I}_0 = (\{x_1, x_2\}, \cdot^{\mathcal{I}_0})$ with

$$A^{\mathcal{I}_0}(x_1) = B^{\mathcal{I}_0}(x_2) = \ell_a \text{ and } B^{\mathcal{I}_0}(x_1) = A^{\mathcal{I}_0}(x_2) = \ell_b$$

is a model of $\mathcal{T}$ and $\ell_a \vee \ell_b = \mathbf{1}$. However, since $\ell \wedge {\sim}\ell \neq \mathbf{1}$ for every $\ell \in L_2$, the axiom can only be satisfied for any $y \in \Delta^{\mathcal{I}}$ if $\{A^{\mathcal{I}}(y), B^{\mathcal{I}}(y)\} = \{\ell_a, \ell_b\}$. Thus, we always have $A^{\mathcal{I}}(y) < \mathbf{1}$.

For this reason, we consider a stronger notion of satisfiability that requires at least one element of the domain to satisfy the concept with the given value. A concept C is *strongly ℓ-satisfiable* w.r.t. a TBox $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ and an $x \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(x) \geq \ell$. Obviously, strong ℓ-satisfiability implies ℓ-satisfiability. As shown in Example 4, the converse does not hold.

Recall that the semantics of the quantifiers require the computation of a supremum or infimum of the membership degrees of a possibly infinite set of elements of the domain. If the lattice is finite, then this is in fact a computation over a finite set of values, but it may be a costly one. If the lattice is infinite, then the problem is more pronounced. For that reason, it is customary in fuzzy description logics to restrict reasoning to witnessed models [16].

Definition 5 (witnessed model). *Let $\eta \in \mathbb{N}$. A model $\mathcal{I}$ of a TBox $\mathcal{T}$ is called* η-witnessed *if for every $x \in \Delta^{\mathcal{I}}$ and every concept description of the form $\exists r.C$ there are η elements $x_1, \ldots, x_\eta \in \Delta^{\mathcal{I}}$ such that*

$$(\exists r.C)^{\mathcal{I}}(x) = \bigvee_{i=1}^{\eta} r^{\mathcal{I}}(x, x_i) \otimes C^{\mathcal{I}}(x_i),$$

and analogously for the universal restrictions $\forall r.C$. In particular, if $\eta = 1$, then the suprema and infima from the semantics of $\exists r.C$ and $\forall r.C$ become maxima and minima, respectively. In this case, we simply say that $\mathcal{I}$ is witnessed.

As we will show, ℓ-satisfiability, even w.r.t. η-witnessed models, is undecidable in general. For finite De Morgan lattices, however, this problem is decidable and belongs to the same complexity class as deciding satisfiability of crisp $\mathcal{ALC}$ concepts, if the lattice operations are easily computable.

4 Undecidability

Consider the lattice L_∞ over the domain $([0,1] \cap \mathbb{Q}) \cup \{-\infty, \infty\}$ with the usual total order, the De Morgan negation ${\sim}\ell = 1 - \ell$ if $\ell \in [0,1]$, ${\sim}\infty = -\infty$, and

$\sim(-\infty) = \infty$, and the t-norm $\otimes$ defined by

$$\ell \otimes m := \begin{cases} \max\{\ell + m - 1, 0\} & \text{if } \ell, m \in [0,1] \text{ and } \ell + m \neq 0, \\ -\infty & \text{if } \ell = m = 0, \text{ and} \\ \min\{\ell, m\} & \text{otherwise.} \end{cases}$$

That is, $\otimes$ is the Łukasiewicz t-norm on the rationals in $(0, 1]$ extended with two extreme elements $-\infty$ and ∞. One can easily confirm that this is in fact a residuated lattice and its t-conorm $\oplus$ is given by

$$\ell \oplus m := \begin{cases} \min\{l + m, 1\} & \text{if } \ell, m \in [0,1] \text{ and } \ell + m \neq 2, \\ \infty & \text{if } \ell = m = 1, \text{ and} \\ \max\{\ell, m\} & \text{otherwise.} \end{cases}$$

We will reduce the well-known undecidable Post Correspondence Problem [18] to decidability of ∞-satisfiability. Notice that for every $T \subseteq L_\infty$, $\bigvee_{t \in T} t = \infty$ iff $\infty \in T$. Thus, a concept is ∞-satisfiable iff it is strongly ∞-satisfiable and it suffices to prove that strong ∞-satisfiability is undecidable.

Definition 6 (PCP). *Let $v_1, \ldots, v_p$ and $w_1, \ldots, w_p$ be two finite lists of words over an alphabet $\Sigma = \{1, \ldots, s\}$. The* Post Correspondence Problem *(PCP) asks whether there is a non-empty sequence $i_1, i_2, \ldots, i_k$, $1 \leq i_j \leq p$ such that $v_{i_1} v_{i_2} \cdots v_{i_k} = w_{i_1} w_{i_2} \cdots w_{i_k}$. Such a sequence, if it exists, is called a* solution *of the problem instance.*

For a word $\nu = i_1 i_2 \cdots i_k \in \{1, \ldots, p\}^*$ we will use v_ν, w_ν to denote the words $v_{i_1} v_{i_2} \cdots v_{i_k}$ and $w_{i_1} w_{i_2} \cdots w_{i_k}$, respectively. Given an instance $\mathcal{P}$ of PCP, we will construct a TBox $\mathcal{T}_\mathcal{P}$ and a concept name S such that S is strongly ∞-satisfiable iff $\mathcal{P}$ has no solution. For doing this, we will encode words w from the alphabet Σ as rational numbers $0.w$ in $[0, 1]$ in base $s + 1$; exceptionally, the empty word will be encoded by the number 0. The two concept names V and W will store the encoding of the concatenated words v_ν and w_ν, respectively.

Given two $\mathcal{ALC}_L$ concept descriptions C, D and a role name r, the expression $\langle C \equiv D \rangle$ abbreviates the two axioms $\langle C \sqsubseteq D, \infty \rangle$,$\langle D \sqsubseteq C, \infty \rangle$ and the expression $\langle C \overset{r}{\rightsquigarrow} D \rangle$ abbreviates the two axioms $\langle C \sqsubseteq \forall r.D, \infty \rangle$, $\langle \neg C \sqsubseteq \forall r.\neg D, \infty \rangle$. For an interpretation $\mathcal{I}$, $\langle C \equiv D \rangle$ expresses that $C^\mathcal{I}(x) = D^\mathcal{I}(x)$ for every $x \in \Delta^\mathcal{I}$, while $\langle C \overset{r}{\rightsquigarrow} D \rangle$ expresses that, for every $x, y \in \Delta^\mathcal{I}$ such that $r^\mathcal{I}(x, y) = \infty$, it holds that $C^\mathcal{I}(x) = D^\mathcal{I}(y)$. We will also use $n \cdot C$ as abbrevation for the n-ary disjunction $C \sqcup \cdots \sqcup C$, which is interpreted at $x \in \Delta^\mathcal{I}$ as the value $\min\{C^\mathcal{I}(x) + \cdots + C^\mathcal{I}(x), 1\} = \min\{n \cdot C^\mathcal{I}(x), 1\}$ whenever $C^\mathcal{I}(x) \in [0, 1]$.

We now define the TBoxes $\mathcal{T}^i_\mathcal{P}$ for $0 \leq i \leq p$ as follows:

$$\begin{aligned} \mathcal{T}^0_\mathcal{P} := \{ & \langle S \sqsubseteq V, 0 \rangle, \langle S \sqsubseteq \neg V, 1 \rangle, \langle S \sqsubseteq W, 0 \rangle, \langle S \sqsubseteq \neg W, 1 \rangle \} \cup \\ & \{ \langle S \sqsubseteq V_i, 0.v_i \rangle, \langle S \sqsubseteq \neg V_i, 1 - 0.v_i \rangle, \\ & \langle S \sqsubseteq W_i, 0.w_i \rangle, \langle S \sqsubseteq \neg W_i, 1 - 0.w_i \rangle \mid 1 \leq i \leq p \}, \end{aligned}$$

$$\begin{aligned}\mathcal{T}_{\mathcal{P}}^{i} := \{&\langle \top \sqsubseteq \exists r_i.\top, \infty\rangle, \langle V \sqcup V_i \overset{r_i}{\rightsquigarrow} V\rangle, \langle W \sqcup W_i \overset{r_i}{\rightsquigarrow} W\rangle\} \cup \\ &\{\langle V_j \equiv (s+1)^{|v_i|} \cdot F_{ij}\rangle, \langle W_j \equiv (s+1)^{|w_i|} \cdot G_{ij}\rangle, \\ &\langle F_{ij} \overset{r_i}{\rightsquigarrow} V_j\rangle, \langle G_{ij} \overset{r_i}{\rightsquigarrow} W_j\rangle \mid 1 \leq j \leq p\}.\end{aligned}$$

Intuitively, $\mathcal{T}_{\mathcal{P}}^{0}$ initializes a search tree for a solution of $\mathcal{P}$, by setting both V and W to the empty word, and describing each pair (v_i, w_i) by the concepts V_i and W_i. Each TBox $\mathcal{T}_{\mathcal{P}}^{i}$ then extends the search tree by concatenating each pair of words v, w produced so far with v_i and w_i, respectively. More formally, consider the interpretation $\mathcal{I}_{\mathcal{P}} = (\Delta^{\mathcal{I}_{\mathcal{P}}}, \cdot^{\mathcal{I}_{\mathcal{P}}})$ where

– $\Delta^{\mathcal{I}_{\mathcal{P}}} = \{1, \ldots, p\}^*$,
– $V^{\mathcal{I}_{\mathcal{P}}}(\nu) = 0.v_\nu, W^{\mathcal{I}_{\mathcal{P}}}(\nu) = 0.w_\nu, V_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = \frac{0.v_i}{(s+1)^{|v_\nu|}}, W_i^{\mathcal{I}_{\mathcal{P}}}(\nu) = \frac{0.w_i}{(s+1)^{|w_\nu|}}$
– $r_i^{\mathcal{I}_{\mathcal{P}}}(\nu, \nu i) = \infty$ and $r_i^{\mathcal{I}_{\mathcal{P}}}(\nu, \nu') = -\infty$ if $\nu' \neq \nu i$, and
– $S^{\mathcal{I}_{\mathcal{P}}}(\varepsilon) = \infty$.

It is easy to see that $\mathcal{I}_{\mathcal{P}}$ is in fact a model of the TBox $\mathcal{T}_0 := \bigcup_{i=0}^{p} \mathcal{T}_{\mathcal{P}}^{i}$. More interesting, however, is that *every* model of this TBox where S is ∞-satisfiable must include $\mathcal{I}_{\mathcal{P}}$, as stated in the following lemma.

Lemma 7. *Let $\mathcal{I}$ be a model of $\mathcal{T}_0$ such that $S^{\mathcal{I}}(x) = \infty$ for some $x \in \Delta^{\mathcal{I}}$. There exists a function $f : \Delta^{\mathcal{I}_{\mathcal{P}}} \to \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}_{\mathcal{P}}}(\nu) = C^{\mathcal{I}}(f(\nu))$ holds for every concept name C occurring in $\mathcal{T}_0$ and $\nu \in \Delta^{\mathcal{I}_{\mathcal{P}}}$.*

Proof (Sketch). The function f is constructed by induction on the length of ν. We can define $f(\varepsilon) := x$ since $S^{\mathcal{I}}(x) = \infty$ and $\mathcal{I}$ is a model of $\mathcal{T}_{\mathcal{P}}^{0}$. Let now ν be such that $f(\nu)$ is already defined. The axioms $\langle \top \sqsubseteq \exists r_i.\top, \infty\rangle$ ensure that, for every $i, 1 \leq i \leq p$ there is a $\gamma \in \Delta^{\mathcal{I}}$ such that $r_i^{\mathcal{I}}(f(\nu), \gamma) = \infty$. The definition $f(\nu i) := \gamma$ satisfies the required property. □

This lemma shows that every model of $\mathcal{T}_0$ must include a search tree for a solution of $\mathcal{P}$. Thus, in order to know whether a solution exists, we need to decide if there is a node of this tree where the concept names V and W are interpreted by the same value. Notice that, for any two values $\ell, m \in [0,1]$, $\ell \neq m$ iff$(\sim\ell \oplus m) \otimes (\ell \oplus \sim m) < 1$. Moreover, $\ell < 1$ i ff$\ell \oplus \ell \leq 1$ or, equivalently, $\sim\ell \otimes \sim\ell \geq 0$. Thus, as $\mathcal{I}_{\mathcal{P}}$ always interprets the concept names V and W in the interval $[0,1]$, it is a model of the TBox

$$\mathcal{T}' := \{\langle E \equiv (\neg A \sqcup B) \sqcap (A \sqcup \neg B)\rangle\} \cup \{ \langle \top \sqsubseteq \forall r_i.\neg(E \sqcup E), 0\rangle \mid 1 \leq i \leq p\}$$

iff $A^{\mathcal{I}_{\mathcal{P}}}(\nu) \neq B^{\mathcal{I}_{\mathcal{P}}}(\nu)$ holds for every $\nu \in \{1, \ldots, p\}^+$.

Theorem 8. *The instance $\mathcal{P}$ of the PCP has a solution iff S is not ∞-satisfiable w.r.t. $\mathcal{T}_{\mathcal{P}} := \mathcal{T}_0 \cup \mathcal{T}'$.*

Notice that the interpretation $\mathcal{I}_{\mathcal{P}}$ is witnessed, which means that undecidability holds even if we restrict reasoning to η-witnessed models, for any $\eta \in \mathbb{N}$.

Corollary 9. *(Strong) satisfiability is undecidable, even if the lattice is a countable, residuated total order and reasoning is restricted to η-witnessed models, with $\eta \in \mathbb{N}$.*

5 Deciding Strong Satisfiability

In the previous section, we have shown that satisfiability is undecidable in general. We now show that if we consider only *finite* De Morgan lattices L, then satisfiability in $\mathcal{ALC}_L$ can be effectively decided. As the following lemmata show, in this case we can restrict to strong ℓ-satisfiability w.r.t. η-witnessed models.

Lemma 10. *The best satisfiability degree for C w.r.t. $\mathcal{T}$ is the supremum of all ℓ such that C is strongly ℓ-satisfiable.*

Proof (Sketch). If C is strongly ℓ-satisfiable and strongly ℓ'-satisfiable, there are two models $\mathcal{I}, \mathcal{I}'$ of $\mathcal{T}$ and $x \in \Delta, x' \in \Delta'$ with $C^{\mathcal{I}}(x) \geq \ell$ and $C^{\mathcal{I}'}(x') \geq \ell'$. The disjoint union of $\mathcal{I}$ and $\mathcal{I}'$ gives a model $\mathcal{J}$ where $\bigvee_{y \in \Delta^{\mathcal{J}}} C^{\mathcal{J}}(y) \geq \ell \vee \ell'$. $\square$

We can then find out whether C is ℓ-satisfiable by comparing ℓ to the best satisfiability degree of C. We will thus focus on finding all the lattice elements that witness the strong ℓ-satisfiability of a given concept.

Lemma 11. *If L has width $\eta \in \mathbb{N}$, i.e. the cardinality of the largest antichain of L is η, then $\mathcal{ALC}_L$ has the η-witnessed model property.*

To simplify the description, we consider $\eta = 1$ only. The algorithm and the proofs of correctness can be easily adapted for any other $\eta \in \mathbb{N}$.

Our approach reduces strong ℓ-satisfiability to the emptiness problem of an automaton on infinite trees. Before giving the details of this reduction, we present a brief introduction to these automata. The automata work over the infinite k-ary tree K^* for $K := \{1, \ldots, k\}$ with $k \in \mathbb{N}$. The positions of the *nodes* in this tree are represented through words in K^*: the empty word ε represents the root node, and ui represents the i-th successor of the node u.

Definition 12 (looping automaton). *A* looping automaton (LA) *is a tuple $\mathcal{A} = (Q, I, \Delta)$ consisting of a finite set Q of* states*, a set $I \subseteq Q$ of* initial states*, and a* transition relation *$\Delta \subseteq Q \times Q^k$. A* run *of $\mathcal{A}$ is a mapping $r : K^* \to Q$ assigning states to each node of K^* such that (i) $r(\varepsilon) \in I$ and (ii) for every $u \in K^*$ we have $(r(u), r(u1), \ldots, r(uk)) \in \Delta$. The* emptiness problem *for LA is to decide whether a given LA has a run.*

The emptiness problem for LA can be solved in polynomial time [23]. It is worth to point out that this procedure not only decides emptiness, but actually computes *all* the states that can be used as initial states to accept a non-empty language. We will later exploit this for computing the best satisfiability degree.

The following automata-based algorithm uses the fact that a concept is strongly ℓ-satisfiable iff it has a well-structured tree model, called a *Hintikka tree.* Intuitively, Hintikka trees are abstract representations of tree models that explicitly express the membership value of all "relevant" concept descriptions. The automaton we construct will have exactly these Hintikka trees as its runs. Strong ℓ-satisfiability is hence reduced to an emptiness test of this automaton.

We denote as $\mathsf{sub}(C,\mathcal{T})$ the set of all subconcepts of C and of the concept descriptions D and E for all $\langle D \sqsubseteq E,\ell\rangle \in \mathcal{T}$. The states of the automaton will be so-called Hintikka sets. These are L-fuzzy sets over the domain $\mathsf{sub}(C,\mathcal{T}) \cup \{\rho\}$, where ρ is an arbitrary new element.

Definition 13 (Hintikka set). *A function $H : \mathsf{sub}(C,\mathcal{T}) \cup \{\rho\} \to L$ is called a* (fuzzy) Hintikka set *for $C,\mathcal{T}$ if the following four conditions are satisfied:*

(i) $H(D \sqcap E) = H(D) \otimes H(E)$ *for every* $D \sqcap E \in \mathsf{sub}(C,\mathcal{T})$,
(ii) $H(D \sqcup E) = H(D) \oplus H(E)$ *for every* $D \sqcup E \in \mathsf{sub}(C,\mathcal{T})$,
(iii) $H(\neg D) = {\sim} H(D)$ *for every* $\neg D \in \mathsf{sub}(C,\mathcal{T})$*, and*
(iv) $H(D) \Rightarrow H(E) \geq \ell$ *for every GCI* $\langle D \sqsubseteq E,\ell\rangle$ *in* $\mathcal{T}$.

The arity k of our automaton is determined by the number of existential and universal restrictions, i.e. concept descriptions of the form $\exists r.D$ or $\forall r.D$, contained in $\mathsf{sub}(C,\mathcal{T})$. Intuitively, each successor will act as the witness for one of these restrictions. The additional domain element ρ will be used to express the degree with which the role relation to the parent node holds. Since we need to know which successor in the tree corresponds to which restriction, we fix an arbitrary bijection $\varphi : \{E \mid E \in \mathsf{sub}(C,\mathcal{T}) \text{ is of the form } \exists r.D \text{ or } \forall r.D\} \to K$. The following conditions define the transitions of our automaton.

Definition 14 (Hintikka condition). *The tuple $(H_0, H_1, \ldots, H_k)$ of Hintikka sets for $C,\mathcal{T}$ satisfies the* Hintikka condition *if:*

(i) $H_0(\exists r.D) = H_{\varphi(\exists r.D)}(\rho) \otimes H_{\varphi(\exists r.D)}(D)$ *for every existential restriction* $\exists r.D \in \mathsf{sub}(C,\mathcal{T})$*, and additionally* $H_0(\exists r.D) \geq H_{\varphi(F)}(\rho) \otimes H_{\varphi(F)}(D)$ *for every restriction* $F \in \mathsf{sub}(C,\mathcal{T})$ *of the form* $\exists r.E$ *or* $\forall r.E$,
(ii) $H_0(\forall r.D) = H_{\varphi(\forall r.D)}(\rho) \Rightarrow H_{\varphi(\forall r.D)}(D)$ *for every universal restriction* $\forall r.D \in \mathsf{sub}(C,\mathcal{T})$*, and additionally* $H_0(\forall r.D) \leq H_{\varphi(F)}(\rho) \Rightarrow H_{\varphi(F)}(D)$ *for every restriction* $F \in \mathsf{sub}(C,\mathcal{T})$ *of the form* $\exists r.E$ *or* $\forall r.E$.

A *Hintikka tree* for $C,\mathcal{T}$ is an infinite k-ary tree $\mathbf{T}$ labeled with Hintikka sets where, for every node $u \in K^*$, the tuple $(\mathbf{T}(u), \mathbf{T}(u1), \ldots, \mathbf{T}(uk))$ satisfies the Hintikka condition. The definition of Hintikka sets ensures that all axioms are satisfied at any node of the Hintikka tree, while the Hintikka condition makes sure that the tree is in fact a witnessed model.

The proof of the following theorem uses arguments similar to those in [2]. The main difference is that one also has to find witnesses for the universal restrictions.

Theorem 15. *Let C be a concept description and $\mathcal{T}$ a TBox. Then C is strongly ℓ-satisfiable w.r.t. $\mathcal{T}$ (in a witnessed model) iff there is a Hintikka tree $\mathbf{T}$ for $C,\mathcal{T}$ such that $\mathbf{T}(\varepsilon)(C) \geq \ell$.*

Proof (Sketch). A Hintikka tree can be seen as a witnessed model with domain K^* and interpretation function given by the Hintikka sets. The conditions satisfied by the Hintikka sets and the Hintikka condition ensure that this interpretation is well-defined. Thus, if there is a Hintikka tree $\mathbf{T}$ for $C,\mathcal{T}$ with $\mathbf{T}(\varepsilon)(C) \geq \ell$, then C is strongly ℓ-satisfiable w.r.t. $\mathcal{T}$.

On the other hand, every witnessed model $\mathcal{I}$ with a domain element $x \in \Delta^{\mathcal{I}}$ for which $C^{\mathcal{I}}(x) \geq \ell$ holds can be *unraveled* into a Hintikka tree $\mathbf{T}$ for $C, \mathcal{T}$ as follows. We start by labeling the root node by the Hintikka set that records the membership values of x for each concept from $\mathsf{sub}(C, \mathcal{T})$. We then create successors of the root by considering every element of $\mathsf{sub}(C, \mathcal{T})$ of the form $\exists r.D$ or $\forall r.D$ and finding the witness $y \in \Delta^{\mathcal{I}}$ for this restriction. We create a new node for y which is an r-successor of the root node with degree $r^{\mathcal{I}}(x, y)$. By continuing this process, we construct a Hintikka tree $\mathbf{T}$ for $C, \mathcal{T}$ for which $\mathbf{T}(\varepsilon)(C) \geq \ell$ holds. □

Thus, strong ℓ-satisfiability w.r.t. witnessed models is equivalent to the non-emptiness of the following automaton.

Definition 16 (Hintikka automaton). *Let C be an $\mathcal{ALC}_L$ concept description, $\mathcal{T}$ a TBox, and $\ell \in L$. The* Hintikka automaton *for $C, \mathcal{T}, \ell$ is the LA $\mathcal{A}_{C,\mathcal{T},\ell} = (Q, I, \Delta)$ where Q is the set of all Hintikka sets for $C, \mathcal{T}$, I contains all Hintikka sets H with $H(C) \geq \ell$, and Δ is the set of all $(k+1)$-tuples of Hintikka sets that satisfy the Hintikka condition.*

The runs of $\mathcal{A}_{C,\mathcal{T},\ell}$ are exactly the Hintikka trees $\mathbf{T}$ having $\mathbf{T}(\varepsilon)(C) \geq \ell$. Thus, C is strongly ℓ-satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{C,\mathcal{T},\ell}$ is not empty.

The size of the automaton $\mathcal{A}_{C,\mathcal{T},\ell}$ is exponential in $C, \mathcal{T}$ and polynomial in L. Hence, the emptiness test for this automaton uses time exponential in $C, \mathcal{T}$ and polynomial in the complexity of the lattice operations on L. Notice however that in general the encoding $\mathrm{enc}(L)$ of a lattice L may be much smaller than the whole lattice L. For this reason we need to consider the complexity of the lattice operations w.r.t. this encoding.

Theorem 17. *If $|L|$ is at most exponential in $|\mathrm{enc}(L)|$ and the lattice operations are in a complexity class* C *w.r.t. the size of* $\mathrm{enc}(L)$,[2] *then strong ℓ-satisfiability (w.r.t. witnessed models) is in* $\textsc{ExpTime}^{\mathrm{C}}$.

Furthermore, the emptiness test of $\mathcal{A}_{C,\mathcal{T},\ell}$ can be used to compute the set of *all* Hintikka sets that may appear at the root of a Hintikka tree. From this set we can extract the set of all values ℓ such that $\mathbf{T}(\epsilon)(C) \geq \ell$ for some Hintikka tree $\mathbf{T}$. From the presented results it follows that the best satisfiability degree can also be computed in $\textsc{ExpTime}^{\mathrm{C}}$.

Corollary 18. *If L is fixed or of size polynomial in $|\mathrm{enc}(L)|$ and $\sim$, $\otimes$ can be computed in time polynomial in $|L|$, then (strong) ℓ-satisfiability (w.r.t. witnessed models) is* ExpTime*-complete.*

Proof. ExpTime-hardness follows from ExpTime-hardness of concept satisfiability in crisp $\mathcal{ALC}$ [1]. By assumption, all lattice operations can be computed in at most polynomial time by several nested iterations over L. Applying Theorem 17 yields inclusion in $\textsc{ExpTime}^{\textsc{PTime}} = \textsc{ExpTime}$. □

[2] More formally, deciding $\ell \leq m$, $\ell \otimes m = n$, etc. for given $\ell, m, n \in L$ is in C.

Notice that the definitions of Hintikka sets and Hintikka trees are independent of the operators used. One could have chosen the residual negation $\ominus \ell := \ell \Rightarrow \mathbf{0}$ to interpret the constructor $\neg$, or the Kleene-Dienes implication $\ell \Rightarrow m := {\sim}\ell \vee m$ instead of the residuum. The only restrictions are that the semantics must be truth functional, i.e. the value of a formula must depend only on the values of its direct subformulas, and the underlying operators must be computable.

As a last remark, we want to point out that the algorithm can be modified for reasoning w.r.t. η-witnessed models with $\eta > 1$. One needs only extend the arity of the Hintikka trees to account for η witnesses for each quantified formula in $\mathsf{sub}(C, \mathcal{T})$. The emptiness test of the automaton, and hence also satisfiability w.r.t. η-witnessed models, is exponential in η.

6 Conclusions

We have introduced the fuzzy DL $\mathcal{ALC}_L$ whose semantics is based on arbitrary complete De Morgan lattices and t-norms. To the best of our knowledge, all previously existing approaches for fuzzy $\mathcal{ALC}$, either based on total orders or on lattices, are special cases of $\mathcal{ALC}_L$.

We showed that reasoning in this logic is undecidable, even if restricted to a very simple infinite lattice and t-norm. This result suggests, but does not prove, that reasoning with the Łukasiewicz t-norm over the interval $[0, 1]$ may, contrary to previous claims [22], be undecidable.

For the special case of finite lattices, we showed decidability by presenting an automata-based decision procedure that runs in exponential time, assuming a polynomial-time oracle for computing the lattice and t-norm operations. An advantage of our decision procedure is that it can easily be adapted to deal with different kinds of truth-functional semantics, and hence is useful for different applications. Given the promising first steps towards an automata-based implementation of $\mathcal{ALC}$ reasoning shown in [10], we believe that our algorithm not only yields an interesting theoretical result, but may be useful for a future implementation. We intend to further study this possibility by developing adequate optimizations and analyzing low-complexity instances of lattice operators.

There are three issues that we will pursue in future work. The first is to explore the limits of undecidability: are there classes of infinite lattices and t-norms in which reasoning is decidable? As said before, it is still unknown whether reasoning in fuzzy $\mathcal{ALC}$ with continuous t-norms over $[0, 1]$ is decidable.

The second issue is to explore the expressivity of DLs. We believe that our approach can easily be adapted to fuzzy $\mathcal{SI}$. Additionally, if we restrict to acyclic TBoxes, we may be able to obtain a PSpace upper bound as in [2].

Finally, we want to develop an algorithm for deciding ℓ-subsumption. Notice that the residuum cannot, in general, be expressed using the t-norm, t-conorm and negation. Thus, the usual idea of reducing subsumption to satisfiability by constructing an equivalent concept cannot be applied.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2nd edition, 2007.
2. F. Baader, J. Hladik, and R. Peñaloza. Automata can show PSPACE results for description logics. *Inform. Comput.*, 206(9-10):1045–1056, 2008.
3. F. Baader and R. Peñaloza. Are fuzzy description logics with general concept inclusion axioms decidable? In *Proc. FuzzIEEE'11*, 2011. To appear.
4. N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logic*, pages 7–37. Reidel Publishing Company, Boston, 1977.
5. F. Bobillo, F. Bou, and U. Straccia. On the failure of the finite model property in some fuzzy description logics. *Fuzzy Set. Syst.*, 172(1):1–12, 2011.
6. F. Bobillo and U. Straccia. Towards a crisp representation of fuzzy description logics under Łukasiewicz semantics. In *Proc. ISMIS'08*, volume 4994 of *LNCS*, pages 309–318. Springer, 2008.
7. F. Bobillo and U. Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Set. Syst.*, 160(23):3382–3402, 2009.
8. F. Bobillo and U. Straccia. Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$. *Inf. Sci.*, 181(4):758–778, 2011.
9. S. Borgwardt and R. Peñaloza. Description logics over lattices with multi-valued ontologies. In *Proc. IJCAI'11*, 2011. To appear.
10. D. Calvanese, D. Carbotta, and M. Ortiz. A practical automata-based technique for reasoning in expressive description logics. In *Proc. IJCAI'11*, 2011. To appear.
11. M. Cerami, F. Esteva, and F. Bou. Decidability of a description logic over infinite-valued product logic. In *Proc. KR 2010*, pages 203–213. AAAI Press, 2010.
12. G. De Cooman and E. E. Kerre. Order norms on bounded partially ordered sets. *J. Fuzzy Math*, 2:281–310, 1993.
13. J. A. Goguen. L-fuzzy sets. *J. Math. Anal. Appl.*, 18(1):145–174, 1967.
14. G. Grätzer. *General Lattice Theory, Second Edition.* Birkhäuser Verlag, 2003.
15. P. Hájek. *Metamathematics of Fuzzy Logic (Trends in Logic).* Springer, 2001.
16. P. Hájek. Making fuzzy description logic more general. *Fuzzy Set. Syst.*, 154(1):1–15, 2005.
17. Y. Jiang, Y. Tang, J. Wang, P. Deng, and S. Tang. Expressive fuzzy description logics over lattices. *Knowl.-Based Syst.*, 23(2):150–161, 2010.
18. E. Post. A variant of a recursively unsolvable problem. *Bulletin of the AMS*, 52:264–268, 1946.
19. G. Stoilos, U. Straccia, G. Stamou, and J. Pan. General concept inclusions in fuzzy description logics. In *Proc. ECAI'06*, pages 457–461. IOS Press, 2006.
20. U. Straccia. Reasoning within fuzzy description logics. *JAIR*, 14:137–166, 2001.
21. U. Straccia. Description logics over lattices. *Int. J. Unc. Fuzz.*, 14(1):1–16, 2006.
22. U. Straccia and F. Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing*, 14(3):247–259, 2007.
23. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.*, 32(2):183–221, 1986.
24. J. Yen. Generalizing term subsumption languages to fuzzy logic. In *Proc. IJCAI'91*, pages 472–477, 1991.
25. L. A. Zadeh. Fuzzy sets. *Inform. Control*, 8(3):338–353, 1965.
26. M. Zherui and W. Wangming. Logical operators on complete lattices. *Inform. Sciences*, 55(1-3):77–97, 1991.

The Complexity of Conjunctive Query Abduction in *DL-Lite*[⋆]

Diego Calvanese[1], Magdalena Ortiz[2], MantasŠimkus [2], and Giorgio Stefanoni[12]

[1] KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
calvanese@inf.unibz.it, giorgio.stefanoni@gmail.com
[2] Institute of Information Systems
Vienna University of Technology, Austria
ortiz@kr.tuwien.ac.at, simkus@dbai.tuwien.ac.at

Abstract. In order to meet usability requirements, most logic-based applications provide explanation facilities for reasoning services. This holds also for DLs, where research focused on the explanation of both TBox reasoning and, more recently, query answering. Besides explaining the presence of a tuple in a query answer, it is important to explain also why a given tuple is missing. We address this latter problem for (conjunctive) query answering over DL-Lite ontologies, by adopting abductive reasoning, that is, we look for additions to the ABox that force a given tuple to be in the result. As reasoning tasks, we consider existence and recognition of an explanation, and relevance and necessity of a certain assertion for an explanation. We characterize the computational complexity of these problems for subset minimal and cardinality minimal solutions.

1 Introduction

Query answering over ontologies formulated in Description Logics (DLs) has received considerable attention both in research and industry. Given an ontology, users typically pose queries over the conceptual schema and get answers that take into account the constraints specified at the conceptual level. Many efforts have concentrated on lightweight description logics. For instance, *DL-Lite*$_\mathcal{A}$ has been tailored for query answering over large data sets [7]. For this reason, expressive power is traded in favor of a better computational behavior in terms of data-complexity. In fact, conjunctive query answering in *DL-Lite*$_\mathcal{A}$ enjoys FOL-rewritability, i.e., it can be reduced to the problem of evaluating a suitably constructed FOL query over a database instance.

In order to meet usability requirements set by domain users, most logic-based applications provide explanation algorithms for reasoning services. This holds also for DLs, where research focused on the explanation of both TBox reasoning [12,6,14,3] and, more recently, query answering [5]. In addition, the latter paper advocates the importance of tackling the problem of explaining the absence of a tuple in the answers to a query over an ontology. This problem stems from the database community, where it has been solved in the context of databases extended with provenance information [9]. We address this problem by considering explanations for the absence of a

[⋆] This work was partially supported by the Austrian Science Fund (FWF) grant P20840.

tuple in the context of query answering over *DL-Lite*$_\mathcal{A}$ ontologies. We adopt *abductive reasoning* [10,11], that is, we consider which additions need to be made to the ABox to force the given tuple to be in the result. More precisely, given a TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and a query q, an *explanation* for a given tuple $\boldsymbol{t}$ is a new ABox $\mathcal{U}$ such that the answer to q over $\langle\mathcal{T}, \mathcal{A} \cup \mathcal{U}\rangle$ contains $\boldsymbol{t}$. An important aspect in explanations is to provide the user with solutions that are simple to understand and free of redundancy, hence as small as possible. To address this requirement, we study various restrictions on solutions, in particular, we focus on subset minimal and cardinality minimal ones. We consider standard decision problems associated to logic-based abduction: *(i) existence* of an explanation, *(ii) recognition* of a given ABox as being an explanation, and *(iii) relevance* and *(iv) necessity* of an ABox assertion, i.e., whether it occurs in some or all explanations. After motivating such problems and formalizing them, we provide algorithms to solve them and a precise characterization of their computational complexity for *DL-Lite*$_\mathcal{A}$. The complexity results for the various reasoning tasks are summarized in Table 1.

2 Preliminaries

***DL-Lite*$_\mathcal{A}$.** *DL-Lite*$_\mathcal{A}$ is a member of the *DL-Lite* family of DLs [7], which have been designed for dealing efficiently with large amounts of extensional information. In *DL-Lite*$_\mathcal{A}$, concept expressions C, denoting sets of objects, and role expressions R, denoting binary relations between objects, are formed as follows:

$$C \longrightarrow A \mid \exists R, \qquad\qquad R \longrightarrow P \mid P^-.$$

where A denotes an atomic concept and P an atomic role[3]. In a *DL-Lite*$_\mathcal{A}$ ontology $\mathcal{O} = \langle\mathcal{T}, \mathcal{A}\rangle$, the TBox $\mathcal{T}$ consists of axioms of the form

$$\begin{array}{lll} C_1 \sqsubseteq C_2, & R_1 \sqsubseteq R_2, & \\ C_1 \sqsubseteq \neg C_2, & R_1 \sqsubseteq \neg R_2, & (\mathsf{funct}\ R), \end{array}$$

and the ABox $\mathcal{A}$ consists of assertions of the form $A(c)$ and $R(c, c')$, where c, c' are constants (or, *individuals*) from a countably infinite set $\mathbf{C}$. An interpretation is a pair $\mathcal{I} = \langle\Delta^\mathcal{I}, \cdot^\mathcal{I}\rangle$, where $\Delta^\mathcal{I}$ is a non-empty *domain*, and the *interpretation function* $\cdot^\mathcal{I}$ is defined as usual. We adopt here the *unique name assumption* (UNA), i.e., $c_1^\mathcal{I} \neq c_2^\mathcal{I}$ for all $c_1, c_2 \in \mathbf{C}$ with $c_1 \neq c_2$. We refer to [7] for more details.

Conjunctive Queries. Let $\mathbf{V}$ be a countably infinite set of variables. Expressions $A(t)$ and $P(t, t')$ are called *atoms*, where $t, t' \in \mathbf{V} \cup \mathbf{C}$. A *conjunctive query (CQ)* q is an expression $q(x_1, \ldots, x_n) \leftarrow a_1, \ldots, a_m$, where each a_i, $1 \leq i \leq m$, is an atom. Let $\mathbf{V}(q)$ denote the set of variables occurring in q, $\mathbf{C}(q)$ the set of constants in q, and let $at(q) = \bigcup_{1\leq i\leq m}\{a_i\}$. A *match* for q in an interpretation $\mathcal{I}$ is a mapping $\pi : \mathbf{V}(q) \cup \mathbf{C}(q) \to \Delta^\mathcal{I}$ such that π is the identity on constants, $\pi(t) \in A^\mathcal{I}$ for each

[3] We ignore here the distinction between data values and objects, since it is immaterial for our results. As a consequence, we do not consider value domains and attributes, which are present in *DL-Lite*$_\mathcal{A}$.

$A(t) \in at(q)$, and $\langle \pi(t), \pi(t') \rangle \in P^{\mathcal{I}}$ for each $P(t, t') \in at(q)$. The tuple $\langle x_1, \ldots, x_n \rangle$ is the tuple of *answer variables* of q. The *answer* to q over $\mathcal{I}$, denoted $\mathsf{ans}(q, \mathcal{I})$, is the set of all n-tuples $\langle d_1, \ldots, d_n \rangle \in \mathbf{C}^n$ such that $\langle d_1^{\mathcal{I}}, \ldots, d_n^{\mathcal{I}} \rangle = \langle \pi(x_1), \ldots, \pi(x_n) \rangle$ for some match π for q in $\mathcal{I}$. A *union of conjunctive queries* (UCQ) is a set of CQs with the same answer variable tuple. For a UCQ q, we let $\mathsf{ans}(q, \mathcal{I}) = \bigcup_{q' \in q} \mathsf{ans}(q', \mathcal{I})$. The *certain answer* to a CQ or a UCQ q over $\mathcal{O}$ is defined as $\mathsf{cert}(q, \mathcal{O}) = \{\boldsymbol{c} \in \mathbf{C}^n \mid \boldsymbol{c} \in \mathsf{ans}(q, \mathcal{I})$ for each model $\mathcal{I}$ of $\mathcal{O}\}$.

3 Explaining Negative Query Answers

We now define the problem considered in this paper:

Definition 1. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology, q a UCQ, and* **c** *a tuple of constants. We call $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ a* query abduction problem *(QAP). A* solution *to $\mathcal{P}$ (or an* explanation for *$\mathcal{P}$) is any ABox $\mathcal{U}$ such that the ontology $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A} \cup \mathcal{U} \rangle$ is consistent and $\boldsymbol{c} \in \mathsf{cert}(q, \mathcal{O}')$. The set of all explanations for $\mathcal{P}$ is denoted $\mathsf{expl}(\mathcal{P})$.*

If $\boldsymbol{c} \notin \mathsf{cert}(q, \mathcal{O})$, then we call $\boldsymbol{c}$ a *negative answer* to q over $\mathcal{O}$. Note that a query over the ontology can have a negative answer only if the ontology is satisfiable. Differently, if the ontology is unsatisfiable then the QAP $\mathcal{P}$ does not have any solution. In the following, we will examine various restrictions to $\mathsf{expl}(\mathcal{P})$ to reduce redundancy in explanations. This is achieved by the introduction of a preference relation among explanations. This relation is reflexive and transitive, i.e., we have a pre-order among solutions.

Definition 2. *Assume a QAP $\mathcal{P}$. Let $\preceq$ denote a pre-order on the set $\mathsf{expl}(\mathcal{P})$ of solutions. We write $\mathcal{U} \prec \mathcal{U}'$ if $\mathcal{U} \preceq \mathcal{U}'$ and $\mathcal{U}' \npreceq \mathcal{U}$. The preferred explanations $\mathsf{expl}_{\preceq}(\mathcal{P})$ of a QAP $\mathcal{P}$ under the pre-order $\preceq$ are defined as follows: $\mathsf{expl}_{\preceq}(\mathcal{P}) = \{\mathcal{U} \in \mathsf{expl}(\mathcal{P}) \mid$ there is no $\mathcal{U}' \in \mathsf{expl}(\mathcal{P})$ s.t. $\mathcal{U}' \prec \mathcal{U}\}$, i.e., $\mathsf{expl}_{\preceq}(\mathcal{P})$ contains all the $\preceq$-*explanations *that are* minimal *under $\preceq$.*

Two preference orders are considered here: the *subset-minimality order*, denoted by $\subseteq$, and the *minimum explanation size order*, denoted by $\leq$. The latter order is defined by $\mathcal{U} \leq \mathcal{U}'$ iff $|\mathcal{U}| \leq |\mathcal{U}'|$. Observe that $\mathsf{expl}_{\leq}(\mathcal{P}) \subseteq \mathsf{expl}_{\subseteq}(\mathcal{P})$.

We define now four decision problems related to (minimal) explanations, which are parametric w.r.t. the chosen preference order $\preceq$. Given a QAP $\mathcal{P}$:

– $\preceq$-EXISTENCE: Does there exist a $\preceq$-explanation for $\mathcal{P}$?
– $\preceq$-RECOGNITION: Is a set $\mathcal{U}$ of ABox assertions a $\preceq$-explanation for $\mathcal{P}$?
– $\preceq$-RELEVANCE: Does an assertion α occur in some $\preceq$-explanation for $\mathcal{P}$?
– $\preceq$-NECESSITY: Does an assertion α occur in all $\preceq$-explanations for $\mathcal{P}$?

In the following, whenever no preference is applied (i.e., when $\preceq$ is the identity) we omit to write $\preceq$ in front of the problem's names. We provide an example, in which we highlight the consequences of choosing among the various orderings.

Table 1: Summary of main complexity results (completeness)

$\preceq$	$\preceq$-EXISTENCE	$\preceq$-RECOGNITION	$\preceq$-RELEVANCE	$\preceq$-NECESSITY
none	PTIME (4.1)	NP	PTIME (4.3)	PTIME (4.2)
$\leq$	PTIME	DP	$\mathsf{P}^{\mathsf{NP}}_{\parallel}$	$\mathsf{P}^{\mathsf{NP}}_{\parallel}$ (4.2)
$\subseteq$	PTIME	DP	Σ^{P}_2 (4.3)	PTIME (4.2)

Example 1. Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology describing a university domain, where $\mathcal{T}$ is

$$\begin{array}{rrr} \mathit{PostGrad} \sqsubseteq \mathit{Student}, & \mathit{Tutor} \sqsubseteq \mathit{Professor}, & \mathit{Advanced} \sqsubseteq \mathit{Course}, \\ \mathit{UnderGrad} \sqsubseteq \mathit{Student}, & \exists \mathit{hasTutor} \sqsubseteq \mathit{PartTime}, & \exists \mathit{teaches} \sqsubseteq \mathit{Professor}, \\ \mathit{UnderGrad} \sqsubseteq \neg \mathit{PostGrad}, & \exists \mathit{hasTutor}^- \sqsubseteq \mathit{Tutor}, & \exists \mathit{teaches}^- \sqsubseteq \mathit{Course}, \\ \mathit{PartTime} \sqsubseteq \mathit{UnderGrad}. & & \end{array}$$

That is, there are two different kinds of students, $\mathit{PostGrad}$ and $\mathit{UnderGrad}$. Moreover, $\mathit{PartTime}$ students are tutored by Tutors, who are particular professors. Additionally, the university offers some $\mathit{Advanced}$ courses. Let the ABox $\mathcal{A}$ consist of the assertions $\mathit{teaches}(\mathit{rob}, \mathit{SWT})$, $\mathit{hasTutor}(\mathit{peter}, \mathit{rob})$. Now, assume that a user is interested in finding all those who both teach an advanced course and tutor a student. Then, she would write the query

$$q(x) \leftarrow \mathit{teaches}(x, y), \mathit{Advanced}(y), \mathit{hasTutor}(z, x).$$

Moreover, she may expect rob to be part of the result, i.e., $\mathit{rob} \in \mathsf{cert}(q, \mathcal{O})$, but this is not the case. Intuitively, rob satisfies all the constraints imposed by the query, except that the SWT course is not known to be $\mathit{Advanced}$. One can easily see that $\{\mathit{teaches}(\mathit{rob}, \mathit{TOC}), \mathit{Advanced}(\mathit{TOC}), \mathit{hasTutor}(\mathit{john}, \mathit{rob})\}$ is an explanation, $\{\mathit{teaches}(\mathit{rob}, \mathit{ALG}), \mathit{Advanced}(\mathit{ALG})\}$ is a $\subseteq$-minimal explanation, while $\{\mathit{Advanced}(\mathit{SWT})\}$ is a $\leq$-minimal explanation. ■

In the next section, the complexity of the four main problems is studied in the light of the different preference relations.

4 Complexity of Explanations

Table 1 provides an overview of our complexity results. Recall that the class Σ^{P}_2 is a member of the Polynomial Hierarchy [13]; it is the class of all decision problems solvable in non-deterministic polynomial time using an NP oracle. Moreover, the class $\mathsf{P}^{\mathsf{NP}}_{\parallel}$ contains all the decision problems that can be solved in polynomial time with an NP oracle, where all oracle calls must be first prepared and then issued in parallel. The class DP contains all problems that, considered as languages, can be characterized as the intersection of a language in NP and a language in CONP [13]. Note that: PTIME $\subseteq$ NP $\subseteq$ DP $\subseteq$ $\mathsf{P}^{\mathsf{NP}}_{\parallel}$ $\subseteq$ Σ^{P}_2 is believed to be a strict hierarchy of inclusions and here we make such an assumption.

Our results can be explained as follows. We show in the next section that EXISTENCE can be reduced to the PTIME-complete satisfiability problem for *DL-Lite*$_\mathcal{A}$ without the UNA [1], which justifies our PTIME upper bound. This result can then be used to characterize the complexity of RELEVANCE, NECESSITY, and $\subseteq$-NECESSITY. $\leq$-RELEVANCE and $\leq$-NECESSITY are harder. The reason being that in order to solve these problems one has to compute first the minimal size of a solution and, then, inspect all the solutions of that size. Additionally, there is another increase in complexity when dealing with $\subseteq$-RELEVANCE. The intuition is that there is an exponential number of candidate solutions to examine and for each of them one has to check that none of its subsets is itself a solution, which requires a coNP computation. Due to space limitations, the results on $\preceq$-RECOGNITION are not detailed in this paper (see [8] for more details). The intuition for the NP bound for RECOGNITION is that one needs simply to check consistency and perform query evaluation to solve the problem. In case a preference order is in place, one has to check minimality as well, which is a coNP check for $\subseteq$- and $\leq$-minimal explanations that leads to completeness for DP.

4.1 Complexity of $\preceq$-EXISTENCE

For $\preceq$-EXISTENCE note first that the existence of an explanation for $\mathcal{P}$ implies the existence of an explanation under the $\subseteq$ and $\leq$ orderings. Thus, we only consider EXISTENCE. Our first task is to show that we can restrict ourselves to explanations built from the original signature of the input QAP plus a small number of fresh constants.

Proposition 1. *If $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ has a solution, then $\mathcal{P}$ has a solution $\mathcal{U}'$ with concepts, roles, and attributes only from $\mathcal{O}$ and at most $m = max_{q_i \in q}|at(q_i)|$ fresh ABox individuals.*

Proof. Assume an arbitrary solution $\mathcal{U}$ to $\mathcal{P}$. Given the consistency of $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A} \cup \mathcal{U} \rangle$, it follows that there exists a model $\mathcal{I}$ of $\mathcal{O}'$ under the UNA. W.l.o.g. we assume that $\Delta^\mathcal{I} = \mathbf{C}$ with $c^\mathcal{I} = c$ for each $c \in \mathbf{C}$. Additionally, the interpretation $\mathcal{I}$ admits a match π for some $q_i(\boldsymbol{x}) \in q$, such that $\pi(\boldsymbol{x}) = \boldsymbol{c}$. Let $\mathcal{U}' = \{A(o) \mid A(t) \in at(q_i)$ and $\pi(t) = o\} \cup \{R(o, o') \mid R(t, t') \in at(q_i)$ and $\pi(t) = o$ and $\pi(t') = o'\}$. Observe that $\mathcal{U}'$ has no more individuals than q_i has variables. It remains to see that $\mathcal{U}'$ is a solution. Clearly the original match π witnesses also $\boldsymbol{c} \in \mathsf{ans}(q_i, \mathcal{A} \cup \mathcal{U}')$. It remains to see that $\mathcal{O}'' = \langle \mathcal{T}, \mathcal{A} \cup \mathcal{U}' \rangle$ is consistent. But this follows from the fact that $\mathcal{I}$ is a model of $\mathcal{O}'$ and that the atoms in $\mathcal{U}'$ hold in $\mathcal{I}$ □

The above restriction allows us to consider *canonical explanations*, i.e., explanations resulting from suitable instantiations of the bodies of CQs $q_i \in q$. Keeping in mind that CQs, seen as FOL formulae, are always satisfiable, an explanation does not exist only if the structure of the query is not compliant with the constraints expressed in the ontology. That is, for all the interpretations $\mathcal{J}$ of q with $\mathsf{ans}(q, \mathcal{J}) \neq \emptyset$, there is no model $\mathcal{I}$ of $\mathcal{O}$, such that $\mathcal{I} \cup \mathcal{J} \models \mathcal{O}$. To check whether a UCQ is compliant with the ontological constraints, a naïve method is to iteratively go through all the CQs in q and instantiate them in the ABox. If for none of the CQs we obtain a consistent ontology, then the query violates some of the constraints imposed at the conceptual level.

Proposition 2. *For DL-Lite$_{\mathcal{A}}$ ontologies,* EXISTENCE *is* PTIME-*complete.*

Proof. (MEMBERSHIP) Note that $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$, with q a UCQ, has a solution iff $\mathcal{P}_{q'} = \langle \mathcal{O}, q', \boldsymbol{c} \rangle$ has a solution for some $q' \in q$. Hence, it suffices to show the upper bound for CQs. To this end, we provide a logspace reduction from EXISTENCE to consistency in *DL-Lite*$_{\mathcal{A}}$ without UNA, which in turn is PTIME-complete [1]. Assume a QAP $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$, where q is a CQ and $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$. We argue that $\mathcal{P}$ has a solution iff $\mathcal{O}' = \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{U}_q \cup \mathcal{A}' \rangle$ is consistent, where $\mathcal{O}'$ is an ontology obtained from $\mathcal{O}$ and $q(\boldsymbol{c})$ as follows. The ABox $\mathcal{U}_q$ is obtained from $at(q(\boldsymbol{c}))$ by replacing each variable x with a fresh individual name a_x. The ABox $\mathcal{A}'$ consists of assertions $A_o(o)$ for all constants o occurring in $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{U}_q$, where each A_o is a fresh concept name. The TBox $\mathcal{T}'$ consists of axioms $A_o \sqsubseteq \neg A_{o'}$ for all pairs $o \neq o'$ of constants occurring in $\mathcal{A}$ and $q(\boldsymbol{c})$. We now show that $\mathcal{P}$ has a solution iff $\mathcal{O}'$ is consistent.

Assume that $\mathcal{P}$ has a solution $\mathcal{U}$. Then, due to consistency of $\mathcal{O}'' = (\mathcal{T}, \mathcal{A} \cup \mathcal{U})$, there is a model $\mathcal{I}$ of $\mathcal{O}''$ under the UNA. Additionally, $\mathcal{I}$ admits a match π for $q(\boldsymbol{c})$. Let $\mathcal{I}'$ be the extension of $\mathcal{I}$ that additionally interprets *(i)* constants in $\mathcal{U}_q$ as $a_x^{\mathcal{I}'} = \pi(x)$ for all variables x in q, and *(ii)* $A_o^{\mathcal{I}'} = \{o^{\mathcal{I}}\}$ for all freshly introduced A_o. It remains to show that $\mathcal{I}'$ is a model of $\mathcal{O}'$. Observe that since $\mathcal{I}$ is under the UNA, we have that $A_o^{\mathcal{I}'} \cap A_{o'}^{\mathcal{I}'} = \emptyset$ for all constant pairs $o \neq o'$. Thus $\mathcal{I}'$ satisfies all the disjointness axioms $A_o \sqsubseteq \neg A_{o'}$ in $\mathcal{T}'$. The assertions in $\mathcal{A}'$ are satisfied due to *(ii)*, while the assertions in $\mathcal{U}_q$ due to *(i)* above.

The other direction of the proof is obvious and we omit it here.

(HARDNESS) Let us reduce consistency in *DL-Lite*$_{\mathcal{A}}$ without UNA to EXISTENCE. Given $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, we create QAP $\mathcal{P} = \langle \mathcal{O}', q(), \langle\rangle \rangle$ simply by encoding the ABox $\mathcal{A}$ in the CQ q by replacing each constant $a \in \mathcal{A}$ by a distinct variable name x_a in q, while the ontology $\mathcal{O}'$ consists only of the TBox $\mathcal{T}$. □

4.2 Complexity of ($\preceq$-)NECESSITY

The existence of an explanation is most of the times not very informative to the user. In fact, given a negative answer to a query, it is important to delineate the fundamental reasons leading to the absence of the expected tuple. That is, users would like to know which assertions occur in all the solutions to a QAP $\mathcal{P}$.

Proposition 3. *For DL-Lite$_{\mathcal{A}}$ ontologies, the* NECESSITY *and* $\subseteq$-NECESSITY *problems are* PTIME-*complete.*

Proof. (MEMBERSHIP) We assume a QAP $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ with $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, an assertion $\varphi(\boldsymbol{t})$ and consider NECESSITY first. This problem can be reduced to non-EXISTENCE, which was shown to be in PTIME in the previous section. We build $\mathcal{O}' = \langle \mathcal{T}', \mathcal{A}' \rangle$ such that $\varphi(\boldsymbol{t})$ occurs in all explanations for $\mathcal{P}$ iff $\mathcal{P}' = \langle \mathcal{O}', q, \boldsymbol{c} \rangle$ has no explanation. We define $\mathcal{O}'$ by setting $\mathcal{A}' = \mathcal{A} \cup \{\bar{\varphi}(\boldsymbol{t})\}$ and $\mathcal{T}' = \mathcal{T} \cup \{\bar{\varphi} \sqsubseteq \neg\varphi\}$, where $\bar{\varphi}$ is a fresh predicate name. It is easy to see that if $\mathcal{P}'$ has no explanation, then either $\mathcal{P}$ has no explanation as well, or, all the explanations for $\mathcal{P}$ must contain $\varphi(\boldsymbol{t})$. For $\subseteq$-NECESSITY, observe that $\varphi(\boldsymbol{t})$ occurs in all explanation for $\mathcal{P}$ iff $\varphi(\boldsymbol{t})$ occurs in all $\subseteq$-minimal explanations for $\mathcal{P}$. Thus $\subseteq$-NECESSITY can be decided in polynomial time using our algorithm for NECESSITY.

(HARDNESS) The lower-bound can be proved through a logspace reduction from EXISTENCE to non-NECESSITY, that is, deciding whether there exists a solution to a QAP that does not contain the given assertion. Let $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ be a QAP with q a CQ, we build $\langle \mathcal{P}, \alpha \rangle$ such that $\mathcal{P}$ has a solution iff $\langle \mathcal{P}, \alpha \rangle$ is a negative instance to NECESSITY. Let $\alpha = A(o)$, for some fresh concept A and constant o not occurring in $\mathcal{P}$. Now, assume $\mathcal{P}$ has a solution $\mathcal{U}$. By Proposition 1, we know that there exists a solution $\mathcal{U}'$ to $\mathcal{P}$ containing only concept and role names from $\mathcal{O}$. Hence, $A(o) \notin \mathcal{U}'$, since concept name A is not in the ontology. Therefore, one can conclude that $A(o)$ is not a necessary assertion to $\mathcal{P}$.

The other direction is straightforward. □

Let us now show the complexity of necessity under the $\leq$ preference order.

Theorem 1. *For DL-Lite$_\mathcal{A}$ ontologies,* $\leq$-NECESSITY *is* $\mathsf{P}_{\parallel}^{\mathsf{NP}}$*-complete.*

Proof. (MEMBERSHIP only, HARDNESS in [8]) Let's assume a QAP $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ and an assertion α. By the use of canonical explanations, we know that the size m of the largest solution to $\mathcal{P}$ corresponds to the size of the largest CQ in q. Observe that $(\mathcal{P}, \alpha)$ is a negative instance of $\leq$-NECESSITY iff there is an $0 \leq i \leq m$ such that *(a)* $\mathcal{P}$ has an explanation $\mathcal{U}$ with $|\mathcal{U}| = i$ and $\alpha \notin \mathcal{U}$, and *(b)* $\mathcal{U}$ is $\leq$-minimal. Thus, we use an auxiliary problem SIZE-OUT, which is to decide given a tuple $\langle \mathcal{P}', \alpha', n' \rangle$, where $\mathcal{P}'$ is a QAP, α' is an assertion, and n' is an integer, whether there exists an explanation $\mathcal{U}'$ for $\mathcal{P}'$ such that $|\mathcal{U}'| = n'$ and $\alpha' \notin \mathcal{U}'$. Furthermore, the problem NO-SMALLER is to decide given a tuple $\langle \mathcal{P}', n' \rangle$ of a QAP and an integer whether there is no explanation $\mathcal{U}'$ for $\mathcal{P}'$ such that $|\mathcal{U}'| < n'$. Observe that SIZE-OUT is in NP, while NO-SMALLER is in coNP. Take the tuple $S = \langle A_0, B_0, \ldots, A_m, B_m \rangle$, where *(a)* $A_i = (\mathcal{P}, \alpha, i)$, for all $0 \leq i \leq m$, and *(b)* $B_i = (\mathcal{P}, i)$, for all $0 \leq i \leq m$. Due to the above observation, α occurs in all $\leq$-minimal explanations $\mathcal{U}$ for $\mathcal{P}$ iff for all $0 \leq i \leq m$, one of the following holds: *(i)* A_i is a negative instance of SIZE-OUT, or *(ii)* B_i is a negative instance of NO-SMALLER. Note that S can be built in polynomial time in the size of the input, while the positivity of the instances in S can be decided by making $2m$ parallel calls to an NP oracle. Thus we obtain membership in $\mathsf{P}_{\parallel}^{\mathsf{NP}}$. □

4.3 Complexity of $\preceq$-RELEVANCE

A domain user faced with a negative answer to a query may ask herself whether, the absence of a certain ABox assertion α in the ontology is related with the lack of the tuple in the results. That is, she would like to know whether α occurs in some explanation to QAP $\mathcal{P}$.

Proposition 4. *For DL-Lite$_\mathcal{A}$ ontologies,* RELEVANCE *is* PTIME*-complete.*

Proof. (MEMBERSHIP) We assume a QAP $\mathcal{P} = \langle \mathcal{O}, q, \boldsymbol{c} \rangle$ with $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an assertion $\phi(\boldsymbol{t})$. We now provide a reduction from RELEVANCE to EXISTENCE. We construct $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}' = \mathcal{A} \cup \phi(\boldsymbol{t})$. Then, $\mathcal{P}$ has an explanation $\mathcal{U}$ with $\phi(\boldsymbol{t}) \in \mathcal{U}$ iff there exists an explanation to $\mathcal{P}' = (\mathcal{O}', q, \boldsymbol{c})$. This is because, any explanation to $\mathcal{P}'$ can be extended by adding $\phi(\boldsymbol{t})$. It is simple to see that any such explanation is an explanation to $\mathcal{P}$, as well. Finally, if $\mathcal{P}'$ does not admit explanations, then $\phi(\boldsymbol{t})$ is a source of inconsistency in $\mathcal{P}$.

(HARDNESS) Hardness can again be proved with a reduction from EXISTENCE in a similar way as done in Section 4.2. □

Let us now tackle the problem of $\subseteq$-RELEVANCE, for which we recall the FOL-rewritability of query answering in *DL-Lite*$_{\mathcal{A}}$. Given an ABox $\mathcal{A}$, let $DB(\mathcal{A})$ be the following interpretation: *(i)* $\Delta^{DB(\mathcal{A})}$ is the set of constants occurring in $\mathcal{A}$, *(ii)* $A^{DB(\mathcal{A})} = \{o \in \Delta^{DB(\mathcal{A})} \mid A(o) \in \mathcal{A}\}$, for each atomic concept A, and *(iii)* $P^{DB(\mathcal{A})} = \{\langle o, o' \rangle \in \Delta^{DB(\mathcal{A})} \mid P(o, o') \in \mathcal{A}\}$, for each atomic role P.

Proposition 5 ([7]). *Let PerfectRef*$(q, \mathcal{T})$ *be the* perfect reformulation *of q w.r.t.* $\mathcal{T}$*, which is a UCQ obtained by applying the rewrite rules given in [7]. Then,* $\mathsf{cert}(q, \mathcal{O}) = \bigcup_{r \in PerfectRef(q,\mathcal{T})} \mathsf{ans}(r, DB(\mathcal{A}))$.

In other words, the certain answers to a UCQ can be computed by rewriting the query into a FOL query to be evaluated over the ABox.

Theorem 2. *For DL-Lite*$_{\mathcal{A}}$ *ontologies,* $\subseteq$-RELEVANCE *is* Σ_2^{P}*-complete.*

Proof. (MEMBERSHIP) The membership in Σ_2^{P} is clear from Algorithm 1, which works as follows. An explanation $\mathcal{U}$ containing $\phi(\boldsymbol{t})$ is non-deterministically computed by guessing an instantiation of a subquery in *PerfectRef*$(q(\boldsymbol{c}), \mathcal{T})$, where $Anon$ is a set of fresh ABox individuals (see Proposition 1). Let HAS-SUBEXPL solve the problem of deciding whether a solution $\mathcal{U}$ has a subset which is itself an explanation. The problem can be easily proved to be in NP. Then, the algorithm checks the complement of HAS-SUBEXPL in order to assure that none of the subsets of $\mathcal{U}$ is itself an explanation, from which it follows that $\phi(\boldsymbol{t})$ is $\subseteq$-relevant. Checking the complement of HAS-SUBEXPL requires the power of a CONP machine. For this reason, the algorithm is solvable in non-deterministic polynomial time by a TM with an NP oracle.

(HARDNESS) We prove it by a reduction from the Σ_2^{P}-complete problem co-CERT3COL [15] (see also [4]). An instance of co-CERT3COL is given by a graph $G = (V, E)$ with vertices $V = \{0, \ldots, n-1\}$ such that every edge is labeled with a disjunction of two literals over the Boolean variables $\{p_{(i,j)} \mid i, j < n\}$. G is a positive instance if there is a truth value assignment t to the Boolean variables such that the graph $t(G)$

Algorithm 1

```
INPUT: QAP $\mathcal{P} = \langle q, \mathcal{O}, \boldsymbol{c} \rangle$ and ABox assertion $\phi(\boldsymbol{t})$
OUTPUT: yes iff $\phi(\boldsymbol{t})$ is relevant to $\mathcal{P}$
 1: Guess $q_i \in \{q_1, \ldots, q_n\} = q$
 2: Guess the derivation of one rewriting $r(\boldsymbol{c})$ in PerfectRef$(q_i(\boldsymbol{c}), \mathcal{T})$
 3: Guess a set of atoms $U \subseteq at(r)$
 4: Guess a mapping $\pi$ from $\mathbf{V}(q)$ to constants in $DB(\mathcal{A})$ and $Anon$
 5: Check that $(\mathcal{T}, \mathcal{A} \cup \mathcal{U}) \not\models \bot$, where $\mathcal{U}$ is the instantiation of $U$ through $\pi$.
 6: Check that $\phi(\boldsymbol{t}) \in \mathcal{U}$ and $\pi$ is a match for $r(\boldsymbol{c})$ over $DB(\mathcal{A} \cup \mathcal{U})$
 7: Check that HAS-SUBEXPL $(\mathcal{P}, \mathcal{U}) = no$.
```

obtained from G by including only those edges whose label evaluates to true under t is not 3-colorable. Assume an instance G of co-CERT3COL. We show how to build in polynomial time a QAP $\mathcal{P}_G = \langle (\mathcal{T}_G, \mathcal{A}_G), q_G, \boldsymbol{c}_G \rangle$ and an ABox assertion α_G such that: G is a positive instance of co-CERT3COL iff α_G is $\subseteq$-relevant for $\mathcal{P}_G$. We use an empty TBox and a Boolean query, thus $\mathcal{T}_G = \emptyset$ and $\boldsymbol{c}_G = \langle\rangle$. The query q_G is a UCQ $q_G = q_{e_1} \cup \cdots \cup q_{e_k} \cup q'$, where $\{e_1, \ldots, e_k\} = E$, each q_{e_i} is an atomic query $q_{e_i}() \leftarrow W_{e_i}(x, y)$, and q' is defined as follows. Assume $\mathcal{B} = \{t, f\}$ to be the set of truth values. The query q' has the following atoms for each edge $e = (i, j)$ in E: *(a)* $B(x_i)$, $R_e(x_i, y_e)$, $R_e(y_e, x_j)$, $B(x_j)$, and *(b)* $P(y_e, z_{p_i})$, $A_{p_i}(z_{p_i})$, $W_{p_i}(z_{p_i}, z'_{p_i})$, where $p_i \in \{p_1, p_2\}$ and p_1, p_2 are the first and the second proposition in the labeling of e, respectively. The query q' simply incorporates G together with the disjunctions on the edges. Observe that if two edges have the same proposition in their label, then this will be reflected in q' by some shared variables z_{p_i}.

To build $\mathcal{A}_G$ we use individuals c_p and $c_{\neg p}$ for the truth value of proposition p. Intuitively, the truth value of p will be determined by either $A_p(c_p)$ or $A_p(c_{\neg p})$ being in the update. Assume a tuple $\boldsymbol{t} = \langle e, v_1, v_2, a, b \rangle$, where $e \in E$, $\{v_1, v_2\} \subseteq \mathcal{B}$, and a, b are individuals. Let p_1, p_2 be the first and the second propositions of e. For $i \in \{1, 2\}$ and $v_i = \mathbf{t}$, let $l_i = p_i$ if p_i is positive and $l_i = \neg p_i$ otherwise. Similarly, for $i \in \{1, 2\}$ and $v_i = \mathbf{f}$, let $l_i = \neg p_i$ if p_i is positive and $l_i = p_i$ otherwise. Then, the ABox $\mathcal{A}(\boldsymbol{t})$ consists of the assertions $R_e(a, d_T)$, $R_e(d_T, b)$, $P(d_T, c_{l_1})$ and $P(d_T, c_{l_2})$ depending on the boolean values in input.

The ABox $\mathcal{A}_G$ is the union of the following ABoxes:

(A1) $\mathcal{A}(\langle e, v, v', a_i, a_j \rangle)$ for all $e \in E$, $v, v' \in \mathcal{B}$, $0 \le i, j \le 2$, and $i \neq j$;
(A2) $\mathcal{A}(\langle e, f, f, a_i, a_i \rangle)$ for all $e \in E$, and $0 \le i \le 2$;
(A3) $\mathcal{A}(\langle e, v, v', b, b \rangle)$ for all $e \in E$, $v, v' \in \mathcal{B}$;
(A4) The ABox $\{B(a_0), B(a_1), B(a_2)\}$;
(A5) The assertions $W_p(c_p, c_{\neg p})$ and $W_p(c_{\neg p}, c_p)$ for all propositions.

Let $\alpha_G = B(b)$. It is not too difficult to see that G is a positive instance of co-CERT3COL iff there exists an $\subseteq$-explanation $\mathcal{U}$ to $\mathcal{P}$ such that $\alpha_G \in \mathcal{U}$. Basically, definitions (A1)-(A3) encode a triangular structure T in which edges in G that evaluate to false according to a given truth assignment can be mapped on any edge of T, reflexive edges included. If an edge of G evaluates to true, then it must be mapped to one of the non-reflexive edges. This ensures that if G can be mapped to T under truth assignment t, then $t(G)$ is 3-colorable. Instead, definitions (A4)-(A5) define a cyclic structure C into which any graph G can be embedded. It has to be noted that the node b is not asserted to be a member of B, hence q_G cannot be mapped there directly with any truth assignment. We see this more formally next:

"$\Rightarrow$" Suppose there is a truth assignment t such that $t(G)$ is not 3-colorable. Let $\mathcal{U} = \{B(b)\} \cup \mathcal{U}_1$, where $\mathcal{U}_1 = \{A_p(c_p) \mid t(p) = t\} \cup \{A_p(c_{\neg p}) \mid t(p) = f\}$. It remains to argue that $\mathcal{U}$ is a $\subseteq$-explanation to $\mathcal{P}$. It is not hard to see that $\mathcal{U}$ is an explanation. Indeed q_G matches already in the ABox obtained by point (A3) (hint: since $B(b) \in \mathcal{U}$, we match q_G by mapping all variables of q_G to (interpretation of) b). Suppose there is a smaller update $\mathcal{U}' \subset \mathcal{U}$. Observe that $\mathcal{U}_1 \subseteq \mathcal{U}'$. This is because for all propositions p, the symbol A_p does not occur in $\mathcal{A}_G$ but does occur in q_G. Then, $\mathcal{U} \setminus \{B(b)\}$ must be an update. If this is the case, then q_G can be matched in $\mathcal{A}_G$ without the ABox from (A3), i.e. in the triangle part. Then $t(G)$ is 3-colorable, which contradicts the assumption.

"$\Leftarrow$" Let $\mathcal{U}$ be a $\subseteq$-minimal explanation containing $B(b)$. Due to the presence of $q_{e_1} \cup \ldots \cup q_{e_k}$ in q_G and the assumption, the role W_p cannot occur in $\mathcal{U}$ for any proposition p. Since $\mathcal{U}$ is an explanation, by the definition of q' and (A5) we have that $A_p(c_p) \in \mathcal{U}$ or $A_p(c_{\neg p}) \in \mathcal{U}$ for all propositions p. Since for any proposition p we have that A_p occurs in q_G with one and only variable z_p, we know that exactly one of $A_p(c_p) \in \mathcal{U}$ and $A_p(c_{\neg p}) \in \mathcal{U}$ holds. Due to the atoms $W_p(z_p, z'_p)$ in q_G, we also have that individuals of the form c_p and $c_{\neg p}$ are the only ones that can get an A_p label. Consider the assignment t defined as follows: $t(e) = \mathsf{t}$ if $A_p(c_p) \in \mathcal{U}$, and $t(e) = \mathsf{f}$ if $A_p(c_{\neg p}) \in \mathcal{U}$. It is not difficult to argue that $t(G)$ is not 3-colorable and thus G is a positive instance of co-CERT3COL. Indeed, if $t(G)$ was 3-colorable, Q should be mappable into the triangle part obtained in (A1)-(A3). Then $\mathcal{U} \setminus \{B(b)\}$ would be a smaller update, which would mean a contradiction. □

Note that the above lower bound holds already for empty TBoxes.

Proposition 6. *For DL-Lite$_\mathcal{A}$ ontologies, $\leq$-RELEVANCE is $\mathsf{P}_{\|}^{\mathsf{NP}}$-complete.*

Proof. (MEMBERSHIP only, HARDNESS in [8]) $\leq$-RELEVANCE can be tackled in a way similar to $\leq$-NECESSITY. In fact, the algorithm described in Theorem 1 can be modified in order to solve this problem. Let SIZE-IN solve the following problem: given a tuple $\langle \mathcal{P}, \alpha, n \rangle$, where $\mathcal{P}$ is a QAP, α an assertion, and n an integer, decide whether there exists an explanation $\mathcal{U}$, with $|\mathcal{U}| = n$ and $\alpha \in \mathcal{U}$. Then, we change the positivity condition of the $\leq$-NECESSITY algorithm as follows: *α occurs in some $\leq$-minimal explanations $\mathcal{U}$ for $\mathcal{P}$ iff for some i, $0 \leq i \leq m$, it holds that: (i) A_i is a positive instance of* SIZE-IN*, and (ii) B_i is a positive instance of* NO-SMALLER. It is easy to see that SIZE-IN is solvable in NP, hence the whole problem is again in $\mathsf{P}_{\|}^{\mathsf{NP}}$. □

5 Conclusions

In this paper we have provided the formalization of a new problem, namely the explanation of negative answers to user queries over ontologies. A tuple is said to be a negative answer, if the user expects it to be part of $\mathsf{cert}(q, \mathcal{O})$ but the tuple is actually not. In our framework, an explanation consists of an ABox that when added to the ontology leads the negative answer to be returned in the results of the query. We define various problems that help us in characterizing the complexity of finding explanations, such as the existence of explanations and relevance/necessity of assertions. We further consider a minimality criterion to be applied over explanations, such as subset-minimal and minimum-size preference orders. Within this framework, we provide a characterization of the computational complexity of the various problems for the DL *DL-Lite$_\mathcal{A}$*.

Future work includes studying the application of this framework to other lightweight description logics, starting with the $\mathcal{EL}$-family. We would also like to investigate the problem in the case where the ontology signature and the explanation signature may be different. That is, the signature over which explanations can be constructed is restricted only to a subset of the ontology signature [2].

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
2. F. Baader, M. Bienvenu, C. Lutz, and F. Wolter. Query and predicate emptiness in description logics. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, 2010.
3. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *J. of Logic and Computation*, 20(1):5–34, 2010.
4. P. A. Bonatti, C. Lutz, and F. Wolter. The complexity of circumscription in description logics. *J. of Artificial Intelligence Research*, 35:717–773, 2009.
5. A. Borgida, D. Calvanese, and M. Rodríguez-Muro. Explanation in the *DL-Lite* family of description logics. In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008)*, volume 5332 of *Lecture Notes in Computer Science*, pages 1440–1457. Springer, 2008.
6. A. Borgida, E. Franconi, and I. Horrocks. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)*, 2000.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. In S. Tessaris and E. Franconi, editors, *Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.
8. D. Calvanese, M. Ortiz, M.Š imkus, and G. Stefanoni. The complexity of conjunctive query abduction in *DL-Lite*. Technical Report INFSYS RR 1843-11-01, Institute of Information Systems, Vienna University of Technology, 2011. Available at: `http://www.kr.tuwien.ac.at/research/reports/`.
9. A. Chapman and H. V. Jagadish. Why not? In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 523–534, 2009.
10. T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. of the ACM*, 42(1):3–42, 1995.
11. S. Klarman, U. Endriss, and S. Schlobach. ABox abduction in the description logic $\mathcal{ALC}$. *J. of Automated Reasoning*, 46(1):43–80, 2011.
12. D. L. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 816–821, 1995.
13. C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Publ. Co., 1994.
14. R. Penaloza and B. Sertkaya. Complexity of axiom pinpointing in the *DL-Lite* family. In *Proc. of the 23rd Int. Workshop on Description Logic (DL 2010)*, volume 573 of *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, 2010.
15. I. A. Stewart. Complete problems involving boolean labelled structures and projection transactions. *J. of Logic and Computation*, 1(6):861–882, 1991.

Mapping Data to Higher-Order Description Logic Knowledge Bases

Floriana Di Pinto, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma
`lastname@dis.uniroma1.it`

Abstract. In this paper we introduce the notion of *mapping-based knowledge base* (MKB), to formalize those ontology-based data access (OBDA) scenarios where both the extensional and the intensional level of the ontology are determined by suitable mapping assertions involving the data sources. We study reasoning over MKBs in the context of $Hi(\textit{DL-Lite}_{\mathcal{R}})$, a higher-order version of the DL *DL-Lite*$_{\mathcal{R}}$. We show that answering queries posed to MKBs expressed in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ can be done efficiently through FOL rewriting: hence, query answering can be delegated to a DBMS, as in the case of traditional OBDA systems.

1 Introduction

Ontology-based data access (OBDA) [2] is a recent application of Description Logics (DLs) that is gaining momentum. The idea behind OBDA is to use a DL ontology as a means to access a set of data sources, so as to mask the user from all application-dependent aspects of data, and to extract useful information from the sources based on a conceptual representation of the domain, expressed as a TBox in a suitable DL. In current approaches to OBDA, the intensional level of the ontology (the TBox) is fixed once for all at design time, and the mapping assertions specify how the data at the sources correspond to instances of the concepts, roles, and attibutes in the TBox. More precisely, the various mapping assertions determine a sort of virtual ABox, in which the individual objects are built out from data, and the instance assertions are specified through the relationships between the sources and the elements of the ontology.

Several OBDA projects have been carried out in the last years [9], and OBDA systems have been designed to support OBDA applications [1]. The experience gained in this work has shown that there are important aspects that are missing in current OBDA technology. In this paper, we concentrate on three such aspects.

The first aspect is related to the need of making the intensional level of the ontology more dynamic. Indeed, in real applications, the information about which are the concepts and roles that are relevant in the domain of interest is often stored in the data sources. Consider, for example, the database $\mathcal{D}$ of a motor industry shown in figure 1, storing data about different types of cars (table `T-CarTypes`), and various cars of such types (table `T-Cars`) manufactured by the firm. The key observation is that the database $\mathcal{D}$ stores information not only about the instances of concepts, but also about the concepts themselves, and their relationships. For example, table `T-CarTypes` tells us that there are four concepts in our ontology that are subconcepts of the concept `Car`, and, implicitely, tells us that they are mutually disjoint. Table `T-Cars`, on the other

T-CarTypes

Code	TypeName
T1	Coupé
T2	SUV
T3	Sedan
T4	Estate

T-Cars

NumberPlate	CarType	EngineSize	BreakPower	Color	TopSpeed
AB111	T1	2000	200	Silver	260
AF333	T2	3000	300	Black	200
BR444	T2	4000	400	Grey	220
AC222	T4	2000	125	Dark Blue	180
BN555	T3	1000	75	Light Blue	180
BP666	T1	3000	600	Red	240

Fig. 1. The database of a motor industry

hand, provides information about the instances of the various concepts, as well as other properties about such instances.

The second aspect is related to the need of metamodeling constructs in the language used to specify the ontology [4, 6]. Metamodeling allows one to treat concepts and properties as first-order citizens, and to see them as individuals that may constitute the instances of other concepts, called meta-concepts. In our example, it is convenient to introduce in the ontology the concept `Car-Type`, whose instances are exactly the subconcepts of cars stored in table `T-CarTypes`. In this way, we allow users to dynamically acquire knowledge about relevant car types through simple queries asking for the instances of the meta-concept `Car-Type`.

The third aspect deals with the need of designing tractable algorithms for query answering in OBDA systems. In [8], it is argued that, since the data sources used in OBDA systems are likely to be very large, such systems should be based on DLs that are tractable in data complexity. In particular, [8] advocates the use of the *DL-Lite* family, that allows for First-Order Logic (FOL) rewritability of (unions of) conjunctive queries. We remind the reader that in a DL enjoing FOL rewritability, query answering can be divided in two steps. In the first step, called rewriting, using the TBox only, the query q is transformed into a new FOL query q', and in the second step q' is evaluated over the ABox. The correctness of the whole method relies on the fact the answers to q' over the ABox coincide with the certain answers to q over the whole ontology. The challenge is now to design tractable query answering algorithms even in cases where the mappings relate data at the sources both to the extensional and the intensional level of the ontology, and meta-concepts and meta-roles are used in the queries. In this paper, we address the above aspects, and present the following contributions.

(*i*) We introduce the notion of *mapping-based knowledge base* (MKB) (Section 3), to formalize the situation where both the extensional and the intensional level of the ontology are determined by suitable mapping assertions involving the data sources.

(*ii*) We describe the higher-order DL $Hi(\textit{DL-Lite}_{\mathcal{R}})$ (Section 2), based on the approach presented in [5]. In that paper, it is shown how, starting from a traditional DL $\mathcal{L}$, one can define its higher-order version, called $Hi(\mathcal{L})$. Here, we apply this idea, and present $Hi(\textit{DL-Lite}_{\mathcal{R}})$, which is the higher-order version of *DL-Lite*$_{\mathcal{R}}$ [3].

(*iii*) We show that answering queries posed to MKBs expressed in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ can be done efficiently through FOL rewriting (Section 4). More specifically, we describe an algorithm that, given a query q over a MKB, rewrites q into a FOL query that is evaluated taking into account only the mapping assertions $\mathcal{M}_{\mathcal{A}}$ of the MKB involving the extensional level of the ontology. Hence query answering can be delegated to a DBMS, as in the case of traditional OBDA systems.

2 Higher-order *DL-Lite*$_{\mathcal{R}}$

In this section, we describe the higher-order DL $Hi(\textit{DL-Lite}_{\mathcal{R}})$, based on the approach presented in [5]. Every traditional DL $\mathcal{L}$ is characterized by a set $OP(\mathcal{L})$ of *operators*, used to form concept and role expressions, and a set of $MP(\mathcal{L})$ of *meta-predicates*, used to form assertions. Each operator and each meta-predicate have an associated arity. If symbol S has arity n, then we write S/n to denote such a symbol and its arity. For *DL-Lite*$_{\mathcal{R}}$, we have

– $OP(\textit{DL-Lite}_{\mathcal{R}}) = \{Inv/1,\ Exists/1\}$;
– $MP(\textit{DL-Lite}_{\mathcal{R}}) = \{Inst_C/2, Inst_R/3, Isa_C/2, Isa_R/2, Disj_C/2, Disj_R/2\}$.

We assume that the reader is familiar with *DL-Lite*$_{\mathcal{R}}$. Therefore, the intuitive meaning of all the above symbols should be clear. The formal specification of their semantics will be given shortly.

Syntax. We assume the existence of two disjoint, countably infinite alphabets: $\mathcal{S}$, the set of *names*, and $\mathcal{V}$, the set of *variables*. Intutively, the names in $\mathcal{S}$ are the symbols denoting the atomic elements of a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ knowledge base. The building blocks of such a knowledge base are assertions, which in turn are based on terms and atoms.

We inductively define the set of *terms*, denoted by $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ as follows:
– if $E \in \mathcal{S}$ then $E \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$;
– if $V \in \mathcal{V}$ then $V \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$;
– if $C/n \in OP(\textit{DL-Lite}_{\mathcal{R}})$ and $t_1,\ldots,t_n \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$ then $C(t_1,\ldots,t_n) \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$.

Ground terms, i.e., terms without variables, are called *expressions*, and the set of expressions is denoted by $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S})$.

A *DL-Lite*$_{\mathcal{R}}$*-atom*, or simply *atom*, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ is a statement of the form $M(E_1,\ldots,E_n)$ where $M \in MP(\textit{DL-Lite}_{\mathcal{R}})$, n is the arity of M, and for every $1 \leq i \leq n$, $E_i \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$. If X is a subset of $\mathcal{V}$, a is a *DL-Lite*$_{\mathcal{R}}$-atom, and all variables appearing in a belongs to X, then a is called an X-atom in *DL-Lite*$_{\mathcal{R}}$.

Ground *DL-Lite*$_{\mathcal{R}}$-atoms, i.e., *DL-Lite*$_{\mathcal{R}}$-atoms without variables, are called *DL-Lite*$_{\mathcal{R}}$*-assertions*, or simply *assertions*. Thus, an assertion is simply an application of a meta-predicate to a set of expressions. Intuitively, an assertion is an axiom that predicates over a set of individuals, concepts or roles.

A $Hi(\textit{DL-Lite}_{\mathcal{R}})$ *knowledge base (KB)* over $\mathcal{S}$ is a set of *DL-Lite*$_{\mathcal{R}}$-assertions over $\mathcal{S}$. To agree with the usual terminology of DLs, we use the term TBox to denote a set of Isa_C, Isa_R, $Disj_C$ and $Disj_R$ assertions, and the term ABox to denote a set of $Inst_C$ and $Inst_R$ assertions.

Semantics. Our definition of semantics for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ is based on the notion of interpretation structure. An *interpretation structure* is a triple $\Sigma = \langle \Delta, \mathcal{I}_c, \mathcal{I}_r \rangle$ where: (*i*) Δ is a non-empty (possibly countably infinite) set; (*ii*) $\mathcal{I}_c$ is a function that maps each $d \in \Delta$ into a subset of Δ; and (*iii*) $\mathcal{I}_r$ is a function that maps each $d \in \Delta$ into a subset of $\Delta \times \Delta$. In other words, Σ treats every element of Δ simultaneously as: (*i*) an individual; (*ii*) a unary relation, i.e., a concept, through $\mathcal{I}_c$; and (*iii*) a binary relation, i.e., a role, through $\mathcal{I}_r$.

An *interpretation* for $\mathcal{S}$ (simply called an interpretation, when $\mathcal{S}$ is clear from the context) over the interpretation structure Σ is a pair $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$, where

– $\Sigma = \langle \Delta, \mathcal{I}_c, \mathcal{I}_r \rangle$ is an interpretation structure, and
– $\mathcal{I}_o$ is a function that maps:
  1. each element of $\mathcal{S}$ to a single object in Δ; and
  2. each element $C/n \in OP(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ to an n-ary function $C^{\mathcal{I}_o} : \Delta^n \to \Delta$ that satisfies the conditions characterizing the operator C/n. In particular, the conditions for the operators in $OP(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ are as follows:
     (a) for each $d_1 \in \Delta$, if $d = \mathit{Inv}^{\mathcal{I}_o}(d_1)$ then $d^{\mathcal{I}_r} = (d_1^{\mathcal{I}_r})^{-1}$;
     (b) for each $d_1 \in \Delta$, if $d = \mathit{Exists}(d_1)$ then $d^{\mathcal{I}_c} = \{e \mid \exists e_1 \text{ s.t. } \langle e, e_1 \rangle \in d_1^{\mathcal{I}_r}\}$.

We extend $\mathcal{I}_o$ to ground terms in $\tau_{\mathit{DL\text{-}Lite}_{\mathcal{R}}}(\mathcal{S})$ inductively as follows: if $C/n \in OP(\mathit{DL\text{-}Lite}_{\mathcal{R}})$, then $(C(t_1, \ldots, t_n))^{\mathcal{I}_o} = C^{\mathcal{I}_o}(E_1^{\mathcal{I}_o}, \ldots, E_n^{\mathcal{I}_o})$.

We now turn our attention to the interpretation of terms in $\mathit{Hi}(\mathit{DL\text{-}Lite}_{\mathcal{R}})$. To interpret non-ground terms, we need assignments over interpretations, where an *assignment* μ over $\langle \Sigma, \mathcal{I}_o \rangle$ is a function $\mu : \mathcal{V} \to \Delta$. Given an interpretation $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$ and an assignment μ over $\mathcal{I}$, the interpretation of terms is specified by the function $(\cdot)^{\mathcal{I}_o,\mu} : \tau_{\mathit{DL\text{-}Lite}_{\mathcal{R}}}(\mathcal{S}, \mathcal{V}) \to \Delta$ defined as follows:

– if $t \in \mathcal{S}$ then $t^{\mathcal{I}_o,\mu} = t^{\mathcal{I}_o}$;
– if $t \in \mathcal{V}$ then $t^{\mathcal{I}_o,\mu} = \mu(t)$;
– if t is of the form $C(t_1, \ldots, t_n)$, then $t^{\mathcal{I}_o,\mu} = C^{\mathcal{I}_o}(t_1^{\mathcal{I}_o,\mu}, \ldots, t_n^{\mathcal{I}_o,\mu})$.

Finally, we define the semantics of atoms, by defining the notion of satisfaction of an atom with respect to an interpretation $\mathcal{I}$ and an assignment μ over $\mathcal{I}$ as follows:

– $\mathcal{I}, \mu \models \mathit{Inst}_C(E_1, E_2)$ if $E_1^{\mathcal{I}_o,\mu} \in (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
– $\mathcal{I}, \mu \models \mathit{Inst}_R(E_1, E_2, E_3)$ if $\langle E_1^{\mathcal{I}_o,\mu}, E_2^{\mathcal{I}_o,\mu} \rangle \in (E_3^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
– $\mathcal{I}, \mu \models \mathit{Isa}_C(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \subseteq (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
– $\mathcal{I}, \mu \models \mathit{Isa}_R(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \subseteq (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
– $\mathcal{I}, \mu \models \mathit{Disj}_C(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \cap (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} = \emptyset$;
– $\mathcal{I}, \mu \models \mathit{Disj}_R(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \cap (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} = \emptyset$.

A $\mathit{Hi}(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ is satisfied by $\mathcal{I}$ if all the assertions in $\mathcal{H}$ are satisfied by $\mathcal{I}$[1]. As usual, the interpretations $\mathcal{I}$ satisfying $\mathcal{H}$ are called the *models* of $\mathcal{H}$. A $\mathit{Hi}(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ is *satisfiable* if it has at least one model.

3 Mapping-based knowledge bases

As we said in the previous section, a $\mathit{Hi}(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ KB is simply a set of assertions. One might think of such a set of assertions as explicitly stated by the designer of the KB. This is a reasonable assumption only in those cases where the ontology is managed by an ad-hoc system, and is built from scratch for the specific application. However, in many applications, it is of interest to derive the KB directly from a set of data sources, so that the assertions of the KB are defined by specific mappings to such data sources. The resulting notion will be called *mapping-based knowledge base*.

In the following, we assume that the data sources are expressed in terms of the relational data model. In other words, all the technical development presented in the rest

[1] We do not need to mention assignments here, since all assertions in $\mathcal{H}$ are ground.

of this section assumes that the set of sources to be linked to the knowledge base is one relational database. Note that this is a realistic assumption, since many data federation tools are now available that are able to wrap a set of heterogeneous sources and present them as a single relational database.

When mapping relational data sources to a knowledge base over $\mathcal{S}$, one should take into account that sources store "data values", and such data values should not be confused with the elements in $\mathcal{S}$. To face this impedance mismatch problem, [8] proposes to structure the mapping assertions in such a way that the elements of the knowledge base are denoted by terms built out from data values stored at the sources using special function symbols. Although we could in principle follow the same idea here, for the sake of simplicity, in this paper we assume that the relational data sources store directly elements of $\mathcal{S}$. Note, however, that all the results presented in the next sections easily extend to the case where mappings build complex terms for denoting the elements of the knowledge base.

We are now ready to provide the definition of mapping-based knowledge base.

Definition 1. *A* $Hi(\textit{DL-Lite}_{\mathcal{R}})$ mapping-based knowledge base (MKB) *is a pair* $\mathcal{K} = \langle DB, \mathcal{M}\rangle$ *such that:* (i) *DB is a relational database;* (ii) *$\mathcal{M}$ is a* mapping*, i.e. a set of* mapping assertions*, each one of the form $\Phi(\boldsymbol{x}) \rightsquigarrow \psi$, where Φ is an arbitrary FOL query over DB of arity $n > 0$ with free variables $\boldsymbol{x} = \langle x_1, \ldots, x_n\rangle$, and ψ is an X-atom in DL-Lite$_{\mathcal{R}}$, with $X = \{x_1, \ldots, x_n\}$.*

In the following, if $\mathcal{K} = \langle DB, \mathcal{M}\rangle$ is a MKB, then we denote by $\mathcal{M}_A$ the set of mapping assertions from $\mathcal{M}$ whose head predicate is either $Inst_C$ or $Inst_R$. Furthermore, we denote by $\mathcal{M}_T$ the set $\mathcal{M} \setminus \mathcal{M}_A$, i.e., the set of mapping assertions from $\mathcal{M}$ whose head predicate belongs to the set $\{Isa_C, Isa_R, Disj_C, Disj_R\}$. We call a mapping $\mathcal{M}$ an *instance-mapping* if $\mathcal{M} = \mathcal{M}_A$, i.e., if the metapredicates $Inst_C$ and $Inst_R$ are the only ones to appear in the right-hand side of the mapping assertions in $\mathcal{M}$.

In order to define the semantics of a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB $\mathcal{K} = \langle DB, \mathcal{M}\rangle$, we need to define when an interpretation *satisfies an assertion in $\mathcal{M}$ with respect to a database DB*. To this end, we make use of the notion of ground instance of an atom, and the notion of answer to a query over DB. Let ψ be an X-atom with $X = \{x_1, \ldots, x_n\}$, and let $\boldsymbol{v}$ be a tuple of arity n with values from DB. Then the ground instance $\psi[\boldsymbol{x}/\boldsymbol{v}]$ of ψ is the formula obtained by substituting every occurrence of x_i with v_i (for $i \in \{1, .., n\}$) in ψ. Also, if DB is a relational database, and q is a query over DB, we write $ans(\Phi, DB)$ to denote the set of answers to q over DB.

We now specify when an interpretation satisfies a mapping assertion with respect to a database. We say that an interpretation $\mathcal{I}$ satisfies the mapping assertion $\Phi(\boldsymbol{x}) \rightsquigarrow \psi$ with respect to the database DB, if for every tuple of values $\boldsymbol{v} \in ans(\Phi, DB)$, the ground atom $\psi[\boldsymbol{x}/\boldsymbol{v}]$ is satified by $\mathcal{I}$. We say that $\mathcal{I}$ is a model of $\mathcal{K} = \langle DB, \mathcal{M}\rangle$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$ with respect to DB.

The following example shows how $Hi(\textit{DL-Lite}_{\mathcal{R}})$ *mapping-based knowledge bases* can be used to model real world situations in a suitable manner.

Example 1. Consider the database $\mathcal{D}$ shown in the introduction. We define a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB $\mathcal{K}_1 = \langle \mathcal{D}, \mathcal{M}\rangle$, where the mapping $\mathcal{M}$ is defined as follows:

– M1: $\{y \mid \texttt{T-CarTypes}(x, y)\} \rightsquigarrow Isa_C(y, \texttt{Car})$
– M2: $\{(x, z) \mid \texttt{T-Cars}(x, y, t, u, v, q) \wedge \texttt{T-CarTypes}(y, z)\} \rightsquigarrow Inst_C(x, z)$
– M3: $\{(x, y) \mid \texttt{T-CarTypes}(z_1, x) \wedge \texttt{T-CarTypes}(z_2, y) \wedge x \neq y\} \rightsquigarrow Disj_C(x, y)$

Intuitively, M1 states that every type of car (whose name appears in the second column of table T-CarTypes, i.e. Coupé, SUV, Sedan, etc.) is a Car. M2, instead, indicates how to correctly retrieve the instances of different types of cars (e.g. car with plate number AB111 has to be retrieved as an instance of the concept Coupé, cars with plate numbers AF333 and BR444 as instances of the concept SUV, and so on). Finally, the intended meaning of assertion M3 is that different types of cars are pairwise disjoint (e.g. a Coupé is not a SUV, a SUV is not a Sedan, and so on). Obviously, mapping assertions are always written by people who know the semantics of the information stored in the database. Notice that the mapping assertions in *M* are able to model the car-types hierarchy without knowing a priori (i.e. at design-time) all the different kinds of cars that are produced by the motor industry.

Suppose now that the motor industry decides to introduce new types of cars to its car fleet, and in particular it decides to produce Campers and Caravans as well, thus extending the hierarchy. As one can imagine, these new kinds of cars share some common characteristics with the previous car types, even though not all of them. Therefore, it might be a reasonable choice for the database designers to introduce a new relational table in $\mathcal{D}$:

T-NewCars

NumberPlate	CarType	Height	Weight	EngineSize	BreakHorsePower
CM777	Camper	2,50 mt	680 Kg	4000 cc	200 bhp
CM888	Camper	2,20 mt	550 Kg	3000 cc	150 bhp
CV333	Caravan	2,30 mt	620 Kg	3000 cc	200 bhp
CV222	Caravan	2,50 mt	580 Kg	4000 cc	250 bhp

The new situation can be modeled in our framework by simply adding to $\mathcal{M}$ the following mapping assertions:
– M4: $\{y \mid \texttt{T-NewCars}(x, y, t, u, v, q)\} \rightsquigarrow \mathit{Isa}_C(y, \texttt{Car})$
– M5: $\{(x, z) \mid \texttt{T-NewCars}(x, z, t, u, v, q)\} \rightsquigarrow \mathit{Inst}_C(x, z)$
– M6: $\{(x, y) \mid \texttt{T-NewCars}(z_1, x) \wedge \texttt{T-NewCars}(z_2, y) \wedge x \neq y\} \rightsquigarrow \mathit{Disj}_C(x, y)$
where mapping M4 states that the new kinds of cars (Camper and Caravan) are Cars, the second assertion indicates how to correctly retrieve their instances, (e.g. car with plate number CM777 as an instance of Camper), and mapping M6 states that the new types of cars are pairwise disjoint (i.e. a Camper is not a Caravan).

Notice that if (instead of creating a new table) the new kinds of cars had been simply introduced into the initial table T-CarTypes (thus without modifying $\mathcal{D}$ in any way), the new concepts (Camper and Caravan) would been automatically detected at run-time by mappings M1-M3, whitout requiring any further mapping definition. □

Next, we introduce the notion of query, which in turn relies on the notion of "query atom". Intuitively, a query atom is a special kind of atom, constituted by a meta-predicate applied to a set of arguments, where each argument is either an expression or a variable. More precisely, we define the set of *q-terms* to be $\tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}) \cup \mathcal{V}$. We define a *query atom* as an atom constituted by the application of a meta-predicate in $MP(\textit{DL-Lite}_\mathcal{R})$ to a set of q-terms, and we call a query atom *ground* if no variable occurs in it. A query atom whose meta-predicate is Inst_C or Inst_R is called an *instance-query atom*. A *higher-order conjunctive query (HCQ)* of arity n is an expression of the form $q(x_1, \ldots, x_n) \leftarrow a_1, \ldots, a_m$ where q, called the query predicate, is a symbol not in $\mathcal{S} \cup \mathcal{V}$, every x_i belongs to $\mathcal{V}$, every a_i is a (possibly non-ground) query atom, and all variables $x_1, \ldots, x_n$ occur in some a_j. The variables $x_1, \ldots, x_n$ are called the *free variables* (or distinguished variables) of the query, while the other variables occurring in $a_1, \ldots, a_m$ are called *existential variables*. A HCQ constituted by instance atoms only is called an *instance HCQ* (IHCQ). A *higher-order union of conjunctive queries (HUCQ)* of arity n is a set of HCQs of arity n with the same query predicate. A HUCQ constituted by instance HCQs only is called an *instance HUCQ* (IHUCQ). A HCQ/HUCQ is called *Boolean* if it has no free variables.

Let $\mathcal{I}$ be an interpretation and μ an assignment over $\mathcal{I}$. A Boolean HCQ q of the form $q \leftarrow a_1, \ldots, a_n$ is *satisfied* in $\mathcal{I}, \mu$ if every query atom a_i is satisfied in $\mathcal{I}, \mu$. A Boolean HUCQ Q is *satisfied* in $\mathcal{I}, \mu$ if there exists a Boolean HCQ $q \in Q$ that is satisfied in $\mathcal{I}, \mu$. A Boolean HUCQ Q is satisfied in $\mathcal{I}$, written $\mathcal{I} \models Q$, if there exists an assignment μ over $\mathcal{I}$ such that Q is satisifed in $\mathcal{I}, \mu$. Given a Boolean HUCQ Q and a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{K}$, we say that Q is *logically implied* by $\mathcal{K}$ (denoted by $\mathcal{K} \models Q$) if for each model $\mathcal{I}$ of $\mathcal{K}$ there exists an assignment μ such that Q is satisfied by $\mathcal{I}, \mu$.

Given a non-Boolean HUCQ q of the form $q(t_1, \ldots, t_n) \leftarrow a_1, \ldots, a_m$, a grounding substitution of q is a substitution θ such that $t_1\theta, \ldots, t_n\theta$ are ground terms. We call $t_1\theta, \ldots, t_n\theta$ a grounding tuple. The set of *certain answers* to q in $\mathcal{K}$ is the set of grounding tuples $t_1\theta, \ldots, t_n\theta$ that make the Boolean query $q_\theta \leftarrow a_1\theta, \ldots, a_n\theta$ logically implied by $\mathcal{K}$. Notice that, in general, the set of certain answers may be infinite even if the KB is finite. Therefore, it is of interest to define suitable notions of safeness, which guarantee that the set of answers is bounded. This issue, however, is beyond the scope of the present paper. Indeed, in this paper, we focus on Boolean queries only, so as to address the computation of certain answers as a decision problem.

Example 2. Let us refer to the MKB $\mathcal{K}_1 = \langle \mathcal{D}, \mathcal{M} \rangle$ of example 1. Interesting queries that can be posed to $\mathcal{K}_1$ include: (*i*) Return all the instances of *Car* manufactured by the motor industry, each one with its own type: $q(x, y) \leftarrow \mathit{Inst}_C(x, y), \mathit{Inst}_C(y, \mathit{Car})$; (*ii*) Return all the concepts which car with plate number *'AB111'* belongs to: $q(x) \leftarrow \mathit{Inst}_C('AB111', x)$.

4 Query answering

In this section, we study the problem of answering IHUCQs over $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKBs. Our query answering technique is based on query rewriting, so we will first deal with the problem of computing a perfect reformulation of a IHUCQ over a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB. Then, we will present a query answering algorithm for MKBs based on the above perfect reformulation technique. In the following, we assume that the MKB is consistent. This does not constitute a limitation, since it is possible to show that checking consistency of a MKB can also be done through query answering, by means of techniques analogous to the ones defined for *DL-Lite*.

We start with some auxiliary definitions. Given an assertion $\alpha = \mathit{Inst}_C(e_1, e_2)$, we say that e_2 occurs as a concept argument in α. Given an assertion $\alpha = \mathit{Inst}_R(e_1, e_2, e_3)$, we say that e_3 occurs as a role argument in α. Given an assertion $\alpha = \mathit{Isa}_C(e_1, e_2)$, we say that e_1 and e_2 occur as concept arguments in α. Given an assertion $\alpha = \mathit{Isa}_R(e_1, e_2)$, we say that e_1 and e_2 occur as role arguments in α. Given an assertion $\alpha = \mathit{Disj}_C(e_1, e_2)$, we say that e_1 and e_2 occur as concept arguments in α. Given an assertion $\alpha = \mathit{Disj}_R(e_1, e_2)$, we say that e_1 and e_2 occur as role arguments in α.

A DL atom is an atom of the form $N(t)$ or $N(t_1, t_2)$, where N is a name and t, t_1, t_2 are either variables or names. An *extended CQ (ECQ)* is a conjunction of DL atoms, Inst_C atoms and Inst_R atoms. An extended UCQ (EUCQ) is a union of ECQs. Given an atom α, $\mathit{Pred}(\alpha)$ denotes the term appearing in predicate position in α (such a term may be either a variable or an expression). Given a TBox $\mathcal{T}$, we denote by $\mathit{Concepts}(\mathcal{T})$ the set $\{e, \mathit{Exists}(e), \mathit{Exists}(\mathit{Inv}(e)) \mid e \in \mathcal{E} \text{ and } e \text{ occurs as a concept argument in } \mathcal{T}\}$ and denote by $\mathit{Roles}(\mathcal{T})$ the set $\{e, \mathit{Inv}(e) \mid e \in \mathcal{E} \text{ and } e \text{ occurs as a role argument in } \mathcal{T}\}$. Given a mapping $\mathcal{M}$ and a database DB, we denote by $\mathit{Retrieve}(\mathcal{M}, DB)$ the $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ defined as:

$$\mathcal{H} = \{\psi(\boldsymbol{t}) \mid \Phi(\boldsymbol{x}) \rightsquigarrow \psi \in \mathcal{M} \text{ and } DB \models \Phi(\boldsymbol{t})\}$$

Given an instance-mapping $\mathcal{M}$ and an ABox $\mathcal{A}$, we say that $\mathcal{A}$ is *retrievable through* $\mathcal{M}$ if there exists a database DB such that $\mathcal{A} = \mathit{Retrieve}(\mathcal{M}, DB)$.

Query rewriting. We start by providing an intuitive explanation of our rewriting technique. The basic idea is to reduce the perfect reformulation of an IHUCQ over a $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ TBox to the perfect reformulation of a standard UCQ over a $\textit{DL-Lite}_{\mathcal{R}}$ TBox, which can be done e.g. by the algorithm $\mathit{PerfectRef}$ presented in [3]. To do so, we have to first transform a IHUCQ into a standard UCQ, actually an EUCQ. This is done through a first partial grounding of the query (through the function PMG) and then through the functions $\mathit{Normalize}$ and τ presented below. Once computed the perfect reformulation of the EUCQ, we then have to transform the EUCQ back into a IHUCQ, through the functions $\mathit{Denormalize}$ and τ^{-} presented below.

Given two IHCQs q, q' and a TBox $\mathcal{T}$, we say that q' is a *partial metagrounding of* q *with respect to* $\mathcal{T}$ if $q' = \sigma(q)$ where σ is a partial substitution of the metavariables of q with the expressions occurring in $\mathcal{T}$ such that, for each metavariable x of q, either $\sigma(x) = x$ or: (i) if x occurs in a concept position in q, then $\sigma(x) \in \mathit{Concepts}(\mathcal{T})$; (ii) if x occurs in a role position in q, then $\sigma(x) \in \mathit{Roles}(\mathcal{T})$. Given an IHCQ q and a TBox $\mathcal{T}$, we denote by $PMG(q, \mathcal{T})$ the set of all partial metagroundings of q with respect to $\mathcal{T}$, i.e., the following IHUCQ Q:

$$Q = \{q' \mid q' \text{ is a partial metagrounding of } q \text{ with respect to } \mathcal{T}\}$$

Moreover, given a IHUCQ Q and a TBox $\mathcal{T}$, we define $PMG(Q, \mathcal{T})$ as the IHUCQ $\bigcup_{q \in Q} PMG(q, \mathcal{T})$.

Given an instance atom α, we define $\mathit{Normalize}(\alpha)$ as follows:

- if $\alpha = \mathit{Inst}_C(e_1, e_2)$ and e_2 has the form $\mathit{Exists}(e')$ where e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_1, \_, e')$;
- if $\alpha = \mathit{Inst}_C(e_1, e_2)$ and e_2 has the form $\mathit{Exists}(\mathit{Inv}(e'))$ where e' is any expression, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(\_, e_1, e')$;
- if $\alpha = \mathit{Inst}_R(e_1, e_2, e_3)$ and e_3 is of the form $\mathit{Inv}^k(e')$ where $k \geq 1$ and k is an even number and e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_1, e_2, e')$;
- if $\alpha = \mathit{Inst}_R(e_1, e_2, e_3)$ and e_3 is of the form $\mathit{Inv}^k(e')$ where $k \geq 1$ and k is an odd number and e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_2, e_1, e')$.

Given an IHCQ $q \leftarrow \alpha_1, \ldots, \alpha_n$, $\mathit{Normalize}(q)$ returns the IHCQ $q \leftarrow \mathit{Normalize}(\alpha_1), \ldots, \mathit{Normalize}(\alpha_n)$. Finally, given an IHUCQ Q, we define $\mathit{Normalize}(Q)$ as $\bigcup_{q \in Q} \mathit{Normalize}(q)$.

Given an IHCQ q and an instance-mapping $\mathcal{M}$, $\mathit{Denormalize}(q, \mathcal{M})$ is the IHUCQ Q defined inductively as follows:

- $q \in Q$;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(t_1, \_, t_2)$, and either $\mathit{Exists}(t_2)$ occurs in $\mathcal{M}$ or $\mathit{Exists}(x)$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_C(t_1, \mathit{Exists}(t_2))$ belongs to Q;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(\_, t_1, t_2)$, and either $\mathit{Exists}(\mathit{Inv}(t_2))$ occurs in $\mathcal{M}$ or $\mathit{Exists}(\mathit{Inv}(x))$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_C(t_1, \mathit{Exists}(\mathit{Inv}(t_2)))$ belongs to Q;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(t_1, t_2, t_3)$ and either $\mathit{Inv}(t_2)$ occurs in $\mathcal{M}$ or $\mathit{Inv}(x)$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_R(t_2, t_1, \mathit{Inv}(t_3)))$ belongs to Q.

Finally, given an IHUCQ Q and a mapping $\mathcal{M}$, we define $\mathit{Denormalize}(Q, \mathcal{M})$ as $\bigcup_{q \in Q} \mathit{Denormalize}(q, \mathcal{M})$.

Given an IHUCQ Q and a TBox $\mathcal{T}$, we denote by $\mathit{PerfectRef}(Q, \mathcal{T})$ the EUCQ returned by the query rewriting algorithm for *DL-Lite*$_\mathcal{R}$ shown in [3].[2]

We now define the functions τ and τ^- which translate IHUCQs into EUCQs and vice versa. Given an IHCQ q and a TBox $\mathcal{T}$, $\tau(q, \mathcal{T})$ is the ECQ obtained from q as follows: (*i*) for each atom of q of the form $\mathit{Inst}_C(t_1, t_2)$, if $t_2 \in \mathit{Concepts}(\mathcal{T})$ then replace the atom with the atom $t_2(t_1)$; (*ii*) for each atom of q of the form $\mathit{Inst}_R(t_1, t_2, t_3)$, if $t_3 \in \mathit{Roles}(\mathcal{T})$ then replace the atom with the atom $t_3(t_1, t_2)$. Then, given an IHUCQ Q, we define $\tau(Q, \mathcal{T}) = \{\tau(q, \mathcal{T}) \mid q \in Q\}$.

Given a ECQ q and a TBox $\mathcal{T}$, $\tau^-(q, \mathcal{T})$ is the IHCQ obtained from q as follows: (*i*) for each atom of q of the form $t_2(t_1)$, replace the atom with the atom $\mathit{Inst}_C(t_1, t_2)$; (*ii*) for each atom of q of the form $t_3(t_1, t_2)$, replace the atom with the atom $\mathit{Inst}_R(t_1, t_2, t_3)$. Then, given an IHUCQ Q, we define $\tau^-(Q, \mathcal{T}) = \{\tau^-(q, \mathcal{T}) \mid q \in Q\}$.

We are now ready to formally define our rewriting algorithm, which takes as input a IHUCQ, a TBox and an instance-mapping, and returns a new IHUCQ.

```
ALGORITHM $\mathit{RewriteIUCQ}(Q, \mathcal{T}, \mathcal{M})$
INPUT: Boolean IHUCQ $Q$, DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$, instance-mapping $\mathcal{M}$
OUTPUT: Boolean IHUCQ $Q'$
    $Q_0 = \mathit{PMG}(Q, \mathcal{T})$;
    $Q_1 = \mathit{Normalize}(Q_0)$;
    $Q_2 = \tau(Q_1, \mathcal{T})$;
    $Q_3 = \mathit{PerfectRef}(Q_2, \mathcal{T})$;
    $Q_4 = \tau^-(Q_3, \mathcal{T})$;
    $Q' = \mathit{Denormalize}(Q_4, \mathcal{M})$;
    return $Q'$;
```

[2] We are actually considering a slight generalization of the algorithm, which allows for the presence of a ternary relation (Inst_R) in the query.

The IHUCQ returned by $\mathit{RewriteIUCQ}(Q, \mathcal{T}, \mathcal{M})$ constitutes a perfect reformulation of the query Q with respect to the TBox $\mathcal{T}$ and the mapping $\mathcal{M}$, as formally stated by the following theorem.

Theorem 1. *Let $\mathcal{T}$ be a TBox, let $\mathcal{M}$ be an instance-mapping and let Q be a IHUCQ. Then, for every ABox $\mathcal{A}$ that is a retrievable through $\mathcal{M}$, $\mathcal{T} \cup \mathcal{A} \models Q$ iff $\mathcal{A} \models \mathit{RewriteIUCQ}(Q, \mathcal{T}, \mathcal{M})$.*

Query answering. Based on the above query rewriting technique, we now present an algorithm for query answering over MKBs. Our idea is to first compute a *DL-Lite*$_\mathcal{R}$ TBox by evaluating the mapping assertions involving the predicates Isa_C, Isa_R, Disj_C, Disj_R over the database of the MKB; then, such a TBox is used to compute the perfect reformulation of the input IHUCQ.

To complete query answering, we now have to consider the mapping of the predicates Inst_C and Inst_R, and to reformulate the query thus obtained replacing the above predicates with the corresponding FOL queries of the mapping assertions. In this way we obtain a FOL query expressed on the database. This second rewriting step, usually called *unfolding*, can be performed by the algorithm UnfoldDB presented in [8].[3]

In the following, given a mapping $\mathcal{M}$ and a database DB, we denote by $DB_{\mathcal{M}_A}$ the database constituted by every relation R of DB such that R occurs in $\mathcal{M}_A$. Analogously, we denote by $DB_{\mathcal{M}_T}$ the database constituted by every relation R of DB such that R occurs in $\mathcal{M}_T$. We are now ready to present our query answering algorithm.

```
ALGORITHM $\mathit{Answer}(Q, \mathcal{K})$
INPUT: Boolean IHUCQ $Q$, $\mathit{Hi}(\mathit{DL\text{-}Lite}_\mathcal{R})$ MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$
OUTPUT: true if $\mathcal{K} \models Q$, false otherwise
    $\mathcal{T} = \mathit{Retrieve}(\mathcal{M}_T, DB_{\mathcal{M}_T})$;
    $Q' = \mathit{RewriteIUCQ}(Q, \mathcal{T}, \mathcal{M}_A)$;
    $Q'' = \mathsf{UnfoldDB}(Q', \mathcal{M}_A)$;
    **if** $DB_{\mathcal{M}_A} \models Q''$
    **then return** true
    **else return** false
```

The algorithm starts by retrieving the TBox from the DB through the mapping $\mathcal{M}_T$. Then, it computes the perfect reformulation of the query with respect to the retrieved TBox, and next computes the unfolding of such a query with respect to the mapping $\mathcal{M}_A$. Finally, it evaluates the query over the database.

The following property can be proved by slightly extending the proof of correctness of the algorithm UnfoldDB shown in [8].

Lemma 1. *Let $\mathcal{M}$ be an instance-mapping and let Q be a IHUCQ. Then, for every database DB, $\langle \mathcal{M}, DB \rangle \models Q$ iff $DB_{\mathcal{M}_A} \models \mathsf{UnfoldDB}(Q, \mathcal{M})$.*

[3] Here we assume that the algorithm UnfoldDB takes as input a EUCQ and an instance-mapping. This corresponds to actually considering a straightforward extension of the algorithm presented in [8] in order to deal with the presence of the ternary predicate Inst_R.

The above lemma allows us to prove correctness of the algorithm $Answer$.

Theorem 2. *Let $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ be a Hi(DL-Lite$_{\mathcal{R}}$) MKB, let Q be a IHUCQ. Then, $\mathcal{K} \models Q$ iff $Answer(Q, \mathcal{K})$ returns true.*

Finally, from the algorithm $Answer$ we are able to derive the following complexity results for query answering over Hi(*DL-Lite*$_{\mathcal{R}}$) MKBs.

Theorem 3. *Let $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ be a Hi(DL-Lite$_{\mathcal{R}}$) MKB, and let Q be a IHUCQ. Deciding whether $\mathcal{K} \models Q$ is in AC^0 with respect to the size of $DB_{\mathcal{M}_A}$, is in PTIME with respect to the size of $\mathcal{K}$, and is NP-complete with respect to the size of $\mathcal{K} \cup Q$.*

5 Conclusions

In this paper we have investigated the possibility of generating a knowledge base on the fly, while computing instance queries, from data stored in data sources through asserted mappings. A key point to obtain such a degree of flexibility is relying on higher-order description logics which blur the distinction between classes/roles at the intensional level and individuals at the extensional level. This paper is only scratching the surfaces of the immense possibilities that this approach opens. For example, we may allow the coexistence of multiple TBoxes within the same data sources, and allow the user to select which TBox to load when querying the system, possibly depending on the query, much in the spirit of [7]. The user can in principle even compose on the fly the TBox to use when answering a query. Obviously notions such as authorization views acquire an intriguing flavor in this setting (hiding intensional as well as extensional knowledge), as well as consistency, since we may even allow for contradicting assertions to coexist as long as they are not used together when performing query answering.

References

1. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The Mastro system for ontology-based data access. *Semantic Web Journal*, 2(1):43–53, 2011.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Ontology-based database access. In *Proc. of SEBD 2007*, pages 324–331, 2007.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
4. W. Chen, M. Kifer, and D. S. Warren. HILOG: A foundation for higher-order logic programming. *J. of Logic Programming*, 15(3):187–230, 1993.
5. G. De Giacomo, M. Lenzerini, and R. Rosati. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*, 2011.
6. J. Z. Pan and I. Horrocks. OWL FA: a metamodeling extension of OWL DL. In *Proc. of WWW 2006*, pages 1065–1066, 2006.
7. J. Parsons and Y. Wand. Emancipating instances from the tyranny of classes in information modeling. *ACM Trans. on Database Systems*, 25(2):228–268, 2000.
8. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
9. D. F. Savo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, V. Romagnoli, M. Ruzzi, and G. Stella. MASTRO at work: Experiences on ontology-based data access. In *Proc. of DL 2010*, volume 573 of *CEUR,* `ceur-ws.org`, pages 20–31, 2010.

Correcting Access Restrictions to a Consequence More Flexibly

Eldora[1]*, Martin Knechtel[2], and Rafael Peñaloza[1]

[1] Theoretical Computer Science, TU Dresden, Germany
eldora.eldora@mailbox.tu-dresden.de, penaloza@tcs.inf.tu-dresden.de
[2] SAP Research, Germany
martin.knechtel@sap.com

Abstract. Recent research has shown that labeling ontologies can be useful for restricting the access to some of the axioms and their implicit consequences. However, the labeling of the axioms is an error-prone and highly sensible task. In previous work we have shown how to correct the access restrictions if the security administrator knows the precise access level that a consequence must receive, and axioms are relabeled to that same access level. In this paper, we look at a more general situation in which access rights can be granted or denied to some specific users, without having to fully specify the precise access level. We also allow a more flexible labeling function, where the new access level of the relabeled axioms may differ from the level of the restriction. We provide black-box algorithms for computing suggestions of axioms to be relabeled.

1 Introduction

Description Logics (DL) [1] have been successfully used to represent knowledge of various application domains. One of the main advantages of using a logic-based knowledge representation language is the possibility of reasoning within the system; that is, deriving implicit consequences from the explicitly stated knowledge in the ontology.

In some application domains it is desirable to restrict users to access only portions of the ontology. For instance, in a security scenario [5], users with a low security clearance should not be able to access classified information. Other motivations for restricting access to users are the reduction of information overload, or filtering w.r.t. a level of specialization. Rather than maintaining different sub-ontologies for each definable user level, we have previously proposed [2] to label each axiom with information on which users can access it. Reasoning then generalizes to the task of finding an adequate label for each implicit consequence of the ontology. This label, called a boundary, can be computed through black-box [2] as well as glass-box [9] techniques.

However, the task of labeling axioms according to their access level is error-prone and highly sensitive to noise. Indeed, a set of seemingly innocuous axioms

* This work was developed while the author worked for SAP Research Dresden.

may allow a user to derive some unwanted consequence. Dually, a too restrictive access level may hide a consequence from relevant users. This problem becomes more pronounced if neither the security administrator nor the knowledge engineer is an expert in logic. We thus want to develop a system that can automatically suggest changes in the labeling function that correct the access to a given consequence.

In previous work [8, 7] we have developed and implemented efficient algorithms for correcting access restrictions to implicit consequences if (i) the knowledge engineer knows the exact access level the consequence must receive (called the goal label) and (ii) axioms are always relabeled to the goal label. In this paper we relax these both conditions. On the one hand, we allow the knowledge engineer to specify a bound on the desired access level, rather than an exact value. This is useful, for instance, to express that a set of users must all have access to the consequence, but it is irrelevant which other users (if any) can also derive it. On the other hand, the knowledge engineer is also able to specify a so-called target label to which the axioms are relabeled. Contrary to the previous approach, the target label needs not be equal to the goal label.

We develop black-box algorithms for finding the minimal sets of axioms that need to be relabeled to the target label in order for the access of the consequence to satisfy the restriction imposed. Additionally, we show that our methods can be improved if one is only interested in finding one such set of minimal cardinality. All our methods are based on results and ideas from axiom-pinpointing [11, 3], but optimized by considering the labels of the axioms used.

2 Preliminaries

To keep our presentation and results as general as possible, we impose only minimal restrictions to our ontology language. We just assume that an *ontology* is a finite set, whose elements are called *axioms*. An ontology language specifies which sets of axioms are admitted as ontologies, with the only restriction that every subset of an ontology is itself an ontology. If $\mathcal{O}' \subseteq \mathcal{O}$ and $\mathcal{O}$ is an ontology, then $\mathcal{O}'$ is called a *sub-ontology* of $\mathcal{O}$. A *monotone consequence relation* $\models$ is a binary relation between ontologies $\mathcal{O}$ and *consequences* c such that if $\mathcal{O} \models c$, then for every ontology $\mathcal{O}' \supseteq \mathcal{O}$ it holds that $\mathcal{O}' \models c$. If $\mathcal{O} \models c$, we say that c *follows from* $\mathcal{O}$ or that $\mathcal{O}$ *entails* c. Consider, for instance, a description logic $\mathcal{L}$. Then, an ontology is a finite set of general concept inclusion axioms (GCIs) of the form $C \sqsubseteq D$, with C, D $\mathcal{L}$-concept descriptions and assertion axioms of the form $C(b)$, with C an $\mathcal{L}$-concept description and b an individual name. An example of a consequence is the subsumption relation $A \sqsubseteq B$ between concept names A, B.

If $\mathcal{O} \models c$, we may be interested in finding the axioms responsible for this fact. A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a *MinA* for $\mathcal{O}$,c if $\mathcal{S} \models c$ and for every $\mathcal{S}' \subset \mathcal{S}, \mathcal{S}' \not\models c$. The dual notion of a MinA is that of a diagnosis. A *diagnosis* for $\mathcal{O}$,c is a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus \mathcal{S} \not\models c$ and $\mathcal{O} \setminus \mathcal{S}' \models c$ for all $\mathcal{S}' \subset \mathcal{S}$.

For a lattice $(L, \leq)$ and a set $K \subseteq L$, we denote as $\bigoplus_{\ell \in K} \ell$ and $\bigotimes_{\ell \in K} \ell$ the *join* (least upper bound) and *meet* (greatest lower bound) of K, respectively. We consider that ontologies are *labeled* with elements of the lattice. More formally, for an ontology $\mathcal{O}$ there is a labeling function lab that assigns a *label* $\mathsf{lab}(a) \in L$ to every element a of $\mathcal{O}$. We will often use the notation $L_{\mathsf{lab}} := \{\mathsf{lab}(a) \mid a \in \mathcal{O}\}$.

For a user labeled with the access level $\ell \in L$, we denote as $\mathcal{O}_{\geq \ell}$ the sub-ontology $\mathcal{O}_{\geq \ell} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) \geq \ell\}$ visible for him. The sub-ontologies $\mathcal{O}_{\leq \ell}, \mathcal{O}_{= \ell}, \mathcal{O}_{\neq \ell}, \mathcal{O}_{\ngeq \ell}$, and $\mathcal{O}_{\nleq \ell}$ are defined analogously. Conversely, for a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, we define

$$\lambda_{\mathcal{S}} := \bigotimes_{a \in \mathcal{S}} \mathsf{lab}(a) \text{ and } \mu_{\mathcal{S}} := \bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a).$$

An element $\ell \in L$ is *join prime relative to* L_{lab} if for every $K_1, \ldots, K_n \subseteq L_{\mathsf{lab}}$, it holds that $\ell \leq \bigoplus_{i=1}^{n} \lambda_{K_i}$ implies that there is $i, 1 \leq i \leq n$ such that $\ell \leq \lambda_{K_i}$. For instance, in the lattice from Figure 1, ℓ_1 and ℓ_4 are the only elements that are not join prime relative to $L_{\mathsf{lab}} = \{\ell_1, \ldots, \ell_5\}$, since $\ell_1 \leq \ell_2 \oplus \ell_4$ but neither $\ell_1 \leq \ell_2$ nor $\ell_1 \leq \ell_4$ and similarly $\ell_4 \leq \ell_5 \oplus \ell_3$ but neither $\ell_4 \leq \ell_5$ nor $\ell_4 \leq \ell_3$. Join prime elements relative to L_{lab} are called *user labels*. The set of all user labels is denoted as U. When dealing with labeled ontologies, the reasoning problem of interest consists on the computation of a boundary for a consequence c. Intuitively, the boundary divides the user labels ℓ of U according to whether $\mathcal{O}_{\geq \ell}$ entails c or not.

Definition 1 (Boundary). *Let $\mathcal{O}$ be an ontology,* lab *a labeling function and c a consequence. An element $\nu \in L$ is called a* boundary *for $\mathcal{O}$,c,*lab *if for every join prime element relative to* L_{lab} *ℓ it holds that $\ell \leq \nu$ iff $\mathcal{O}_{\geq \ell} \models c$.*

Given a user label ℓ_u, we will say that the user *sees* a consequence c if $\ell_u \leq \nu$ for some boundary ν. The following lemma relating MinAs and boundaries was shown in [2].

Lemma 2. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all MinAs for $\mathcal{O}$,c, then $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$,c.*

A dual result relating the boundary with the set of diagnoses, also holds.

Lemma 3. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all diagnoses for $\mathcal{O}$,c, then $\bigotimes_{i=1}^{n} \mu_{\mathcal{S}_i}$ is a boundary for $\mathcal{O}$,c.*

Example 4. Let $(L_d, \leq_d)$ be the lattice shown in Figure 1, and $\mathcal{O}$ a labeled ontology from a marketplace in the Semantic Web with the following axioms

a_1 : *EUecoService* $\sqcap$ *HighperformanceService*(*ecoCalculatorV1*)
a_2 : *HighperformanceService*
  $\sqsubseteq$ *ServiceWithLowCustomerNr* $\sqcap$ *LowProfitService*
a_3 : *ServiceWithLowCustomerNr* $\sqsubseteq$ *ServiceWithComingPriceIncrease*
a_4 : *EUecoService* $\sqsubseteq$ *ServiceWithLowCustomerNr* $\sqcap$ *LowProfitService*
a_5 : *LowProfitService* $\sqsubseteq$ *ServiceWithComingPriceIncrease*

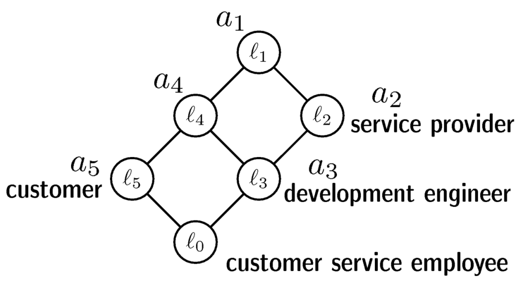


Fig. 1. Lattice $(L_d, \leq_d)$ with 4 user labels and an assignment of 5 axioms to labels

where the function lab assigns to each axiom a_i the label ℓ_i as shown in Figure 1. This ontology entails *ServiceWithComingPriceIncrease*(*ecoCalculatorV1*). The MinAs for $\mathcal{O}$,c are $\{a_1, a_2, a_3\}$, $\{a_1, a_2, a_5\}$, $\{a_1, a_3, a_4\}$, $\{a_1, a_4, a_5\}$, and its diagnoses are $\{a_1\}$, $\{a_2, a_4\}$, $\{a_3, a_5\}$. Using Lemma 3, we can compute the boundary as $\mu_{\{a_1\}} \otimes \mu_{\{a_2,a_4\}} \otimes \mu_{\{a_3,a_5\}} = \ell_1 \otimes \ell_1 \otimes \ell_4 = \ell_4$. The join prime elements relative to L_{lab}, which define valid user labels, are $\ell_0, \ell_2, \ell_3, \ell_5$. These labels represent the user roles as illustrated. Thus, the consequence c is only visible for the user roles ℓ_0, ℓ_5 and ℓ_3, i.e. for customer service employees, customers, and development engineers.

3 Modifying the Boundary

An efficient implementation of a black-box algorithm for computing the boundary of DL consequences already exists [2]. However, a desirable addition to this method is the capability of automatically relabeling some of the axioms to correct the access level of some implicit consequence. Indeed, labeling axioms w.r.t. their access restrictions is highly error-prone, and very small changes in the labeling function may produce consequences to become visible to unauthorized users, or inaccessible to the relevant users.

We have previously shown [8, 7] how to detect a set of axioms of minimal cardinality that needs to be relabeled for obtaining a given boundary. However, in that setting the knowledge engineer must specify the exact boundary that the consequence must receive, and all axioms are relabeled to that value. We now relax these restrictions, by allowing more general constraints on the new boundary, and a more flexible relabeling function.

Definition 5 (Boundary Constraint, Change Set). *A* boundary constraint *is a tuple $\beta = (c, \propto \ell_g, \ell_t)$, where c is a consequence, $\propto \ell_g$, with $\propto \in \{\leq, \geq, \not\geq, \not\leq\}$, $\ell_g \in L$ is a* condition *and ℓ_t is the target label with $\ell_t \propto \ell_g$.*

Let $\mathcal{O}$ be an ontology, $\mathcal{S} \subseteq \mathcal{O}$, lab a labeling function, and $\ell \in L$. We define the modified labeling function $\mathsf{lab}_{\mathcal{S},\ell}$ as

$$\mathsf{lab}_{\mathcal{S},\ell}(a) = \begin{cases} \ell & \text{if } a \in \mathcal{S}, \\ \mathsf{lab}(a) & \text{otherwise.} \end{cases}$$

A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a change set (CS) *for the boundary constraint $\beta = (c, \propto \ell_g, \ell_t)$ if the boundary ν for $\mathcal{O}, c, \mathsf{lab}_{\mathcal{S},\ell_t}$ satisfies $\nu \propto \ell_g$.*

For the rest of this paper, we assume, w.l.o.g. that the boundary for $\mathcal{O}, c, \mathsf{lab}$ does not satisfy the condition $\propto \ell_g$ since otherwise, the empty set is already a CS and nothing needs to be changed in the labeling function.

Notice that, since $\ell_t \propto \ell_g$, the whole ontology $\mathcal{O}$ is always a change set. However, using the whole ontology as a change set would set ℓ_t as the boundary of every consequence of $\mathcal{O}$. In general, we want to make the least possible changes when correcting the boundary of a given consequence. For that reason, we will focus on finding all those change sets that are minimal w.r.t. set inclusion. These sets are useful if the knowledge engineer wants to obtain several suggestions of correction, and then choose the adequate one by some external criterion. However, due to the huge number (possibly exponentially many [4]) of change sets that may exist, one may also look for the "best" change set, and use it automatically in the correction. Hence, we also study how to find a *smallest* change set; that is, one with the least cardinality.

We divide this section in two parts. First we look at the case where the boundary restriction is of the form $\leq$ or $\geq$. We show that previously known techniques can be used also in this setting. We then look at the negative restrictions, which require new methods to be developed.

3.1 Positive Conditions

We now focus on the case where the condition of the boundary constraint is of the form $\geq \ell_g$. Due to the duality of MinAs and diagnoses, the case for $\leq \ell_g$ can be treated in an analogous way (see e.g. [8]).

Let $\beta = (c, \geq \ell_g, \ell_t)$ be a boundary constraint and $\ell_t \geq \ell_g$. Recall (Lemma 2) that the boundary can be computed as the supremum of all $\lambda_{\mathcal{S}_i}$, where $\mathcal{S}_i$ is a MinA for $\mathcal{O}, c$. Thus, if we relabel all the axioms in a MinA $\mathcal{S}$ to ℓ_t, then the boundary for $\mathcal{O}, c, \mathsf{lab}_{\mathcal{S},\ell_t}$ is $\geq \ell_t \geq \ell_g$; that is, every MinA is a change set. Yet, this change set may not be minimal. In fact, we only need that the infimum of the labels of all the axioms in this MinA is $\geq \ell_g$. This can be achieved by only relabeling the axioms in $\mathcal{S}$ that are not already $\geq \ell_g$.

Example 6. Continuing Example 4, recall that we have computed the label ℓ_4 as the boundary of the consequence c. Suppose now that we want to change this boundary to be $\geq \ell_2$, using ℓ_2 also as the relabeling target. As described above, every MinA is also a change set for this consequence. If we consider the MinA $\mathcal{S} = \{a_1, a_2, a_3\}$, then under the new labeling $\mathsf{lab}_{\mathcal{S},\ell_2}$ we obtain the new boundary

$$\lambda_{\{a_1,a_2,a_3\}} \oplus \lambda_{\{a_1,a_2,a_5\}} \oplus \lambda_{\{a_1,a_3,a_4\}} \oplus \lambda_{\{a_1,a_4,a_5\}} = \ell_2 \oplus \ell_0 \oplus \ell_3 \oplus \ell_0 = \ell_2.$$

However, it is easy to see through a simple computation, that the set $\{a_3\}$ is also a change set, which is strictly included in the previous MinA. This set is obtained from the MinA by removing all axioms whose label is greater or equal ℓ_2, namely a_1 and a_2.

Intuitively, we simply consider every axiom $a \in \mathcal{O}$ with $\mathsf{lab}(a) \geq \ell_g$ as *fixed* in the sense that its label cannot be changed, as changing it will be superfluous for any CS. We thus consider a generalization of MinAs, called IAS.

Definition 7 (IAS). *A* minimal inserted axiom set (IAS) *for ℓ is a subset $I \subseteq \mathcal{O}$ such that $\mathcal{O}_{\geq \ell} \cup I \models c$ and $\mathcal{O}_{\geq \ell} \cup I' \not\models c$ for all $I' \subset I$.*

The known algorithms for computing all MinAs [6, 12] through a hitting set tree (HST) method [10] can easily be adapted for also computing IAS [8]. More interestingly, the set of all minimal change sets corresponds to the set of all IAS.

Theorem 8. *Let $\mathcal{O}$ be an ontology, $\beta = (c, \geq \ell_g, \ell_t)$ a boundary constraint and $\mathcal{S} \subseteq \mathcal{O}$. $\mathcal{S}$ is a minimal CS for β iff $\mathcal{S}$ is an IAS for ℓ_g.*

In [8, 7] it is shown how to compute the set of all IAS for a consequence c. Moreover, the algorithms presented there have been also optimized for finding the smallest IAS, through the inclusion of a cardinality restriction. Basically, the construction of an IAS stops once that this has reached the cardinality of the smallest IAS found so far. It was shown that using these (partial) IAS can drastically reduce the search space, while preserving correctness of the method. Due to Theorem 8, all the algorithms for computing IAS and IAS of minimal cardinality can be used for finding the minimal change sets and a change set of minimal cardinality, for positive boundary constraints.

3.2 Negative Conditions

We now consider the case in which the boundary constraint has a condition of the form $\not\geq \ell_g$. As in the previous section, the case for $\not\leq \ell_g$ can be solved dually by simply interchanging MinAs and diagnoses.

Given an ontology $\mathcal{O}$, a labeling function lab and a consequence c, if the boundary for $\mathcal{O}, c, \mathsf{lab}$ is greater or equal to ℓ_g, then we know that for every diagnosis $\mathcal{S}$ for $\mathcal{O}, c$ it holds that $\mu_{\mathcal{S}} \geq \ell_g$ (see Lemma 3). Hence, if we relabel all the axioms in any diagnosis $\mathcal{S}$ to $\ell_t \not\geq \ell_g$, it follows that the boundary is then changed to a new value $\not\geq \ell_g$; that is, $\mathcal{S}$ is a CS. However, just as in the previous section, this CS may not be minimal. One idea to try to find a minimal CS is to follow the same intuition as in the previous section, and fix all axioms whose labels already satisfy the condition $\not\geq \ell_g$. Unfortunately, this idea is not correct, as shown by the following example.

Example 9. Returning to Example 4, suppose now that we want to change the boundary from ℓ_4 to some value $\not\geq \ell_4$, using ℓ_5 as a target label. Recall that $\{a_2, a_4\}$ and $\{a_3, a_5\}$ are diagnoses for the consequence. If we consider the axioms having a label $\not\geq \ell_4$ as fixed, then none of these diagnoses produces a change set. In the first one, the axiom a_2 would be fixed, but then, under the relabeling $\mathsf{lab}_{\{a_4\},\ell_5}$ we will obtain the boundary

$$\mu_{\{a_1\}} \otimes \mu_{\{a_2,a_4\}} \otimes \mu_{\{a_3,a_5\}} = \ell_1 \otimes (\ell_2 \oplus \ell_5) \otimes (\ell_3 \oplus \ell_5) = \ell_1 \otimes \ell_1 \otimes \ell_4 = \ell_4,$$

Algorithm 1 Compute one minimal CS contained in a diagnosis

```
Procedure compute-one-CS($\mathcal{S}, \beta$)
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint;
Output: $\mathcal{T} \subseteq \mathcal{S}$ minimal CS for $\beta$ contained in $\mathcal{S}$
1: if $\ell_t \geq \ell_g$ then
2:     return $\emptyset$
3: $\mathcal{T} := \mathcal{S}$
4: $\ell := \ell_t$
5: for every $a \in \mathcal{S}$ do
6:     if $\ell \oplus \mathsf{lab}(a) \not\geq \ell_g$ then
7:         $\mathcal{T} := \mathcal{T} \setminus \{a\}$
8:         $\ell := \ell \oplus \mathsf{lab}(a)$
9: return $\mathcal{T}$
```

which does not satisfy the restriction $\not\geq \ell_4$; hence, $\{a_4\}$ is not a change set.

In the case of the second diagnosis, the problem is even greater, since both axioms will be considered as fixed. Thus, the approach would deduce that no axiom needs to be relabeled to obtain a boundary $\not\geq \ell_4$, which is obviously not true.

Despite this, it is still possible to use diagnoses as a basis for computing the minimal CS. Suppose that we have a diagnosis $\mathcal{S}$ containing an axiom a_0 such that $\ell_t \oplus \mathsf{lab}(a_0) \not\geq \ell_g$. Then, $\mathcal{S}' = \mathcal{S} \setminus \{a_0\}$ is also a CS, since

$$\bigoplus_{a \in \mathcal{S}} \mathsf{lab}_{\mathcal{S}', \ell_t}(a) = \ell_t \oplus \mathsf{lab}(a_0) \not\geq \ell_g.$$

Obviously, this result holds not only for a single axiom a_0 but for any subset $\mathcal{T}$ of $\mathcal{S}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$.

Lemma 10. *Let $\mathcal{S}$ be a diagnosis for $\mathcal{O}, c$ and $\beta = (c, \not\geq \ell_g, \ell_t)$ a boundary constraint. If $\mathcal{T}$ is a subset of $\mathcal{S}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$, then $\mathcal{S} \setminus \mathcal{T}$ is a CS for β.*

Proof. For every axiom $a \in \mathcal{S} \setminus \mathcal{T}$, $\mathsf{lab}_{\mathcal{S} \setminus \mathcal{T}, \ell_t}(a) = \ell_t$. Additionally, we know that $\bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a) \geq \ell_g$, and hence $\mathcal{T} \neq \mathcal{S}$. Thus, under the new labeling, we have that

$$\bigoplus_{a \in \mathcal{S}} \mathsf{lab}_{\mathcal{S} \setminus \mathcal{T}, \ell_t}(a) = \ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$$

Since $\mathcal{S}$ is a diagnosis, Lemma 3 implies that the new boundary satisfies the condition, and hence $\mathcal{S} \setminus \mathcal{T}$ is a CS. □

A simple consequence of this lemma is that, given a maximal subset $\mathcal{T}$ of $\mathcal{S}$ satisfying $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$, $\mathcal{S} \setminus \mathcal{T}$ is a minimal change set for β *contained* in $\mathcal{S}$. Algorithm 1 describes how to compute one such minimal change set from a diagnosis. This, however, might not be a "globally" minimal change set; that is, there might still exist other change sets strictly contained in it, as shown in the following example.

Example 11. Consider the lattice in Figure 1, an ontology $\mathcal{O}$ having four axioms $\{a_1, a_2, a_3, a_4\}$, and a consequence c such that the diagnoses for $\mathcal{O}, c$ are the sets $\{a_1, a_2, a_3\}$ and $\{a_1, a_4\}$. Assume that the labeling function lab is given by the mapping $\mathsf{lab}(a_1) = \ell_4, \mathsf{lab}(a_2) = \ell_5, \mathsf{lab}(a_3) = \mathsf{lab}(a_4) = \ell_2$. It is easy to see that the boundary for this consequence is ℓ_1. If we apply Algorithm 1 to the diagnosis $\{a_1, a_2, a_3\}$ and the boundary constraint $\beta = (c, \not\geq \ell_1, \ell_3)$, where at Line 5, we first choose a_3, then ℓ is changed to ℓ_2 at Line 8, and hence the test $\ell \oplus \mathsf{lab}(a) \not\geq \ell_1$ fails for axioms a_1 and a_2. Thus, the algorithm returns the change set $\{a_1, a_2\}$. However, $\{a_1\}$ is also a change set, since if a_1 is relabeled to ℓ_3, then $\mu_{\{a_1,a_4\}} = \ell_2$,and thus the boundary is $\not\geq \ell_1$.

Although Algorithm 1 does not always output a globally minimal change set, one can still use it for computing all the minimal change sets for β. The idea is based on the following lemma, which is a simple consequence of the definition of diagnoses and change sets.

Lemma 12. *Let $\mathcal{S}$ be a minimal change set for $(c, \not\geq \ell_g, \ell_t)$. Then, there exists a set $\mathcal{T}$ such that (i) $\ell_t \oplus \bigoplus_{a \in \mathcal{T}} \mathsf{lab}(a) \not\geq \ell_g$ and (ii) $\mathcal{S} \cup \mathcal{T}$ is a diagnosis for $\mathcal{O}, c$.*

For instance, in Example 11 we found the minimal change set $\{a_1\}$. The set $\mathcal{T} = \{a_4\}$ satisfies the two conditions stated in Lemma 12.

To compute all minimal change sets, one then needs to compute all diagnoses, and from each of these diagnoses compute all the minimal change sets that are contained in it. This is possible through a nesting of two hitting set tree (HST) algorithms: the external one produces all different diagnoses for $\mathcal{O}, c$, while the internal generates, for any given diagnosis, all the maximal subsets of axioms that can be removed to obtain a CS. Algorithm 2 shows how this internal HST algorithm works.

The idea behind all HST-like algorithms is the following. One first computes a set of axioms $\mathcal{T}$ satisfying some property; in the case of Algorithm 2, the set is a minimal CS for β contained in $\mathcal{S}$. This set is then used to label the root of the tree. The algorithm then branches as follows. For each axiom a in $\mathcal{T}$, a new branch is created and a is removed from the search space. A new set $\mathcal{T}'$ satisfying the property is then computed, and used to label the successor node. The removal of the axiom $a \in \mathcal{T}$ from the search space ensures that $\mathcal{T} \not\subseteq \mathcal{T}'$. This process is then iterated until the property is not satisfied by the search space; that is, Algorithm 1 returns the empty set. This process stops after at most exponentially many iterations, on the size of $\mathcal{S}$, and the labels of the tree contain all the minimal sets of axioms satisfying the property; in our case, all minimal change sets contained in the diagnosis.

There are two common optimizations for HST algorithms, which are also used in Algorithm 2. The first one is called *early path termination.* The idea behind this optimization is that if one can distinguish parts of the tree that will yield no new minimal sets of axioms, then one can stop exploring those branches. The two conditions for early path termination described in Line 1 of $\mathsf{expand\text{-}hst}$ test for a path where the search space is contained in a search space already explored in a

Algorithm 2 HST to compute all minimal CS contained in a diagnosis

```
Procedure compute-all-CS($\mathcal{S}, \beta$)
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint;
Output: $\mathbf{C}$: all minimal CS for $\beta$ contained in $\mathcal{S}$
 1: Global $\mathbf{C}, \mathbf{H} := \emptyset$
 2: $\mathcal{T}$ :=compute-one-CS($\mathcal{S}, \beta$)
 3: $\mathbf{C} := \{\mathcal{T}\}$
 4: for each $a \in \mathcal{T}$ do
 5:     expand-hst($\mathcal{S}, (c, \not\geq \ell_g, \ell_t \oplus \mathsf{lab}(a)), \{a\}$)
 6: return C
Procedure expand-hst($\mathcal{S}, \beta, H$)
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint; $H$: list of axioms
Side effects: modifications to $\mathbf{C}$,$\mathbf{H}$
 1: if exists some $H' \in \mathbf{H}$ such that $H' \subseteq H$ or
       $H'$ contains a prefix path $P$ with $P = H$ then
 2:     return                                            (early path termination)
 3: $\mathcal{T}' := \emptyset$
 4: if exists some $\mathcal{T} \in \mathbf{C}$ such that $\ell_t \oplus \bigoplus_{a \in \mathcal{S} \setminus \mathcal{T}} \mathsf{lab}(a) \propto \ell_g$ then
 5:     $\mathcal{T}' := \mathcal{T}$                                                (CS reuse)
 6: else
 7:     $\mathcal{T}'$ :=compute-one-CS($\mathcal{S}, \beta$)
 8: if $\mathcal{T}' \neq \emptyset$ then
 9:     $\mathbf{C} := \mathbf{C} \cup \{\mathcal{T}'\}$
10:     for each $a \in \mathcal{T}'$ do
11:         expand-hst($\mathcal{S}, (c, \not\geq \ell_g, \ell_t \oplus \mathsf{lab}(a)), H \cup \{a\}$)
12: else
13:     $\mathbf{H} := \mathbf{H} \cup \{H\}$                                    (normal termination)
```

previous branch. The second optimization is the reuse of sets. When expanding a tree, we only ask for a set of axioms satisfying the property that is contained in the current search space. If these conditions hold in a previously computed label, then we can reuse it, avoiding this way a possibly expensive call to Algorithm 1.

To find all "global" minimal change sets, we use an additional HST algorithm that computes all diagnoses, and for each of these, calls Algorithm 2. This algorithm uses the same kind of optimizations. However, to improve the functionality of the reuse of solutions, the set of all change sets computed so far is kept in a global variable, accessible from every call to compute-all-CS. Thus, a change set that has been previously computed from a diagnosis $\mathcal{S}$, can be reused in a call with a different diagnosis $\mathcal{S}'$.

It is worth noticing that in some cases, a diagnosis may contain several axioms labeled with the same lattice element. Moreover, the condition for obtaining a minimal CS from Lemma 10 depends only on the labeling, and not in the axiom itself. Thus, it is sometimes possible to optimize the search for the minimal CS by considering only the labels and not the individual axioms, as described in Algorithm 3. The correctness of this algorithm is justified by the following lemma, whose proof is analogous to the one of Lemma 10.

Algorithm 3 Compute one minimal CS contained in a diagnosis (optimized)

```
Procedure compute-one-CS($\mathcal{S}, \beta$)
Input: $\mathcal{S}$: diagnosis; $\beta = (c, \not\geq \ell_g, \ell_t)$: boundary constraint;
Output: $\mathcal{T} \subseteq \mathcal{S}$ minimal CS for $\beta$
 1: if $\ell_t \geq \ell_g$ then
 2:     return no CS
 3: $\mathcal{T} := \mathcal{S}$
 4: $\ell := \ell_t$
 5: $L := \{\mathsf{lab}(a) \mid a \in \mathcal{S}\}$
 6: for every $m \in L$ do
 7:     if $\ell \oplus m \not\geq \ell_g$ then
 8:         $\mathcal{T} := \mathcal{T} \setminus \{a \mid \mathsf{lab}(a) = m\}$
 9:         $\ell := \ell \oplus m$
10: return $\mathcal{T}$
```

Lemma 13. *Let $\mathcal{S}$ be a diagnosis for $\mathcal{O}, c$, $\beta = (c, \not\geq \ell_g, \ell_t)$ a boundary constraint, and $L_{\mathcal{S}} = \{\mathsf{lab}(a) \mid a \in \mathcal{S}\}$. If $\mathcal{M} \subseteq L_{\mathcal{S}}$ is such that $\ell_t \oplus \bigoplus_{\ell \in \mathcal{M}} \ell \not\geq \ell_g$, then $\mathcal{S} \setminus \{a \mid \mathsf{lab}(a) \in \mathcal{M}\}$ is a CS for β.*

As in the case for positive conditions, these algorithms can be further optimized if one is only interested in a change set of minimal cardinality. Notice simply that in Algorithms 1 and 3, whenever the condition in the **for** loop is violated, then at least an axiom is ensured to belong to the output change set. Thus, it is easy to adapt these algorithms to include a cardinality bound, returning a partial CS once it has reached a given size. Since our method uses an HST approach, the proofs of correctness of the variant of HST capable of exploiting cardinality restrictions [8] hold also in this case. In other words, Algorithm 2 can be further optimized to compute only one change set of minimal cardinality.

4 Conclusions

We have presented algorithms for correcting the boundary of a consequence in a more flexible manner than previous approaches. Our framework allows the knowledge engineer to set bounds on what the new boundary should be, and specify a label as the target of the relabeling. This flexibility is useful if, for instance, she wants to grant access to a consequence to some user, but is not willing to specify the exact set of users that should access it.

We developed algorithms that output all the minimal change sets. Additionally, we show how these algorithms can be optimized if one is only interested in an arbitrary change set of minimal cardinality.

As future work, we will first implement and test the performance of our methods on large-scale real-world ontologies and applications. We also plan to generalize our framework to allow axioms to be relabeled to different elements of the lattice, according to an adequate minimality criterion.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
2. F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In A. B. et al., editor, *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, Washington, DC, 2009.
3. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, February 2010. Special Issue: Tableaux and Analytic Proof Methods.
4. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In J. Hertzberg, M. Beetz, and R. Englert, editors, *Proceedings of the 30th German Annual Conference on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer-Verlag.
5. C. Farkas and S. Jajodia. The inference problem: a survey. *SIGKDD Explor. Newsl.*, 4(2):6–11, 2002.
6. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In K. Aberer, K.-S. Choi, N. F. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *Proc. of the 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf. (ISWC'07,ASWC'07)*, volume 4825 of *LNCS*, pages 267–280, Busan, Korea, 2007. Springer-Verlag.
7. M. Knechtel and R. Peñaloza. Correcting access restrictions to a consequence. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proceedings of the 2010 International Workshop on Description Logics (DL'10)*, volume 573 of *CEUR-WS*, Waterloo, Canada, 2010.
8. M. Knechtel and R. Peñaloza. A generic approach for correcting access restrictions to a consequence. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *Lecture Notes in Computer Science*, pages 167–182, 2010.
9. R. Peñaloza. Using sums-of-products for non-standard reasoning. In A.-H. Dediu, H. Fernau, and C. Martín-Vide, editors, *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, volume 6031 of *Lecture Notes in Computer Science*, pages 488–499. Springer-Verlag, 2010.
10. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
11. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.
12. B. Suntisrivaraporn. *Polynomial-time Reasoning Support for Design and Maintenance of Large-scale Biomedical Ontologies.* PhD thesis, Technische Universität Dresden, 2009.

Ontology Design and Integration with ICOM 3.0 - Tool description and methodology

Pablo R. Fillottrani[1], Enrico Franconi[2], and Sergio Tessaris[2]

[1] Universidad Nacional del Sur, Argentina
[2] Free University of Bozen-Bolzano, Italy

1 Introduction

ICOM is an advanced conceptual modelling tool, which allows the user to design multiple ER or UML class diagrams with inter- and intra-model constraints. Complete logical reasoning is employed by the tool to verify the specification, infer implicit facts, devise stricter constraints, and manifest any inconsistency.

For the ontology creation and maintenance tasks, ICOM interface supports ontology engineers in engineering ontologies that meets clear and measurable quality criteria. Indeed, recently we observe the development of large numbers of ontologies which have, however, usually been developed in an ad hoc manner by domain experts, often with only a limited understanding of the semantics of ontology languages. The result is that many ontologies are of low quality - they make poor use of the languages in which they are written and do not accurately capture the author's rich knowledge of the domain. This problem becomes even more acute as ontologies are maintained and extended over time, often by multiple authors. Poor quality ontologies usually require localised "tuning" in order to achieve the desired results within applications. This leads to further degradation in their overall quality, increases the brittleness of the applications that use them, and makes interoperability and reuse difficult or impossible. To overcome these problems tools are needed which support the design and the development of the basic infrastructure for building, merging, and maintaining ontologies.

The leverage of automated reasoning to support the domain modelling is enabled by a precise semantic definition of all the elements of the class diagrams. The diagrams and inter-model constraints are internally translated into a class- based logic formalism. The same underlying logic enables the use of a view definition language to specify additional constraints, not captured at the diagram level. The conceptual modelling language supported by ICOM can express most of features of the Extended Entity-Relationship data model or the UML class diagrams (we are working on supporting *Object-Role Modelling* [5] as well). Moreover it extends with disjoint and covering constraints and definitions attached to classes and relations by means of view expressions over other classes and relationships in the ontology; as well as inter-ontology mappings, as inclusion and equivalence statements between view expressions involving classes and relationships possibly belonging to different ontologies.

The main purpose of the ICOM project is not to provide to the ontology community a robust tool potentially replacing the many other tools available, and we do not claim that ICOM is currently more usable than any of the existing conceptual modelling tools for ontology design (such as, for example, [8, 1]). ICOM is meant to be a proof of concept, willing to showcase two main points: (1) the effectiveness of using a class diagram graphical syntax for expressing ontologies, even with complex languages; (2) the emphasis to the use of complex automated reasoning tasks to deduce implied facts, as opposed to mere subsumption (classification) and consistency.

The two above points are novel and in our opinion very important in the context of the existing ontology design tools and methodologies (see next Section). Indeed ICOM proves (point 1) the feasibility and the ease of use of a class diagram graphical syntax for expressing ontologies, even with complex ontology languages, by relying on the notion of views (which roughly correspond to OCL constructs) in order to capture the (typically very few) cases where a larger expressivity than graphical class diagrams is needed.
ICOM is based on a *deduction-complete* notion of reasoning support relative to the class diagram graphical syntax (point 2). Users will see the original ontology *graphically completed* with all the deductions making sense given the provided ontology, and expressed in the graphical class diagram language itself. This includes checking class and relationship consistency, discovering implied class and relationship inter-relations (e.g., subsumption) or cardinality constraints, and in general discovering any implied but originally implicit class diagram graphical construct. Customarily, ontology design tools just provide a support limited to class subsumption and consistency.

ICOM is a fairly mature project, its first release has been published in 2000 (see [7, 4]). The version 3.0 of the ICOM tool is loosely based on the ICOM tool previously released in 2000 as an Entity-Relationship editor (which had around 3,000 registered installations, mostly in academic environments and for teaching purposes in industry), and a demo of a preliminary version was presented few years ago [3]. The foundations of the user-computer interaction have been radically changed according to the experience of the first ICOM and the research in this last decade. The system has been completely re-implemented, using different graphic libraries. The graphical interface has been completely rewritten to improve the usability and intuitiveness of the tool. Interoperability with other tools is a crucial aspect; so, import and export modules have been developed for XMI 2.x and Description Logics based ontology languages via DIG.

The ICOM tool is written in standard Java 5.0, and it is distributed on Linux, Mac, and Windows machines.[3] ICOM communicates via the DIG 1.1 protocol with a description logic server, such as, for example, RACER. ICOM provides an interface for importing and exporting ontologies in UML-XMI class diagrams format.

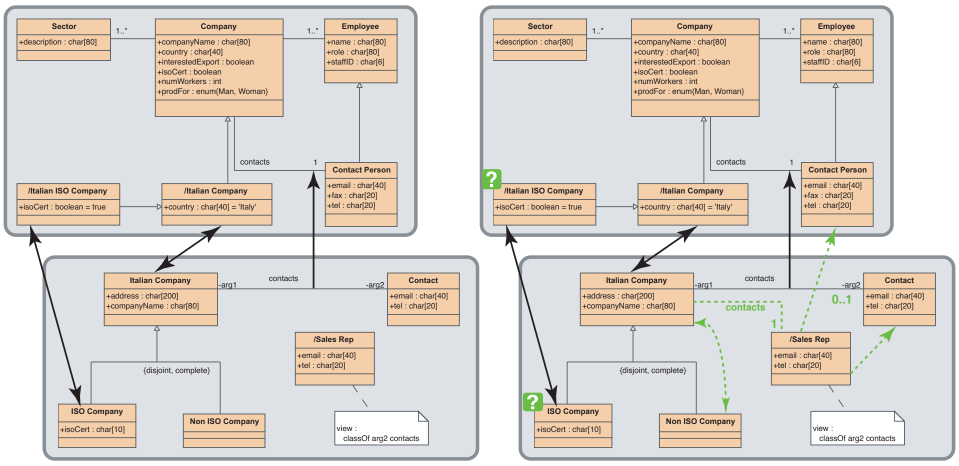
2 Ontology Integration and Views in ICOM

In this section we introduce a scenario of usage of the tool in the context of ontology integration by making use of the view facility of ICOM.

Figure 1a shows two ontologies in the phase to be integrated by the ontology engineer. The top ontology describes concepts where information about Italian ISO certified companies is held; in particular, the information about their contact person is specified.

[3] Available as a free download at `http://www.inf.unibz.it/~franconi/icom/`

The facts described by the diagram state that a company should have at least one employee, and that it should be involved in at least one sector. Among the employees there is the contact person of the company, which should be unique. Moreover, the Italian companies are exactly defined as those companies which are in a country called Italy, while the ISO certified Italian companies are exactly those Italian companies having an ISO certification (specified as a boolean property called `isoCert`). Please note the particular use of the 'slash' "/" operator in front of the completely intensionally defined classes—in the ontology the classes `Italian Company` and `Italian ISO Company` are completely defined by means of the properties specified in the diagram. This is the simplest case of a view defined in the ontology.



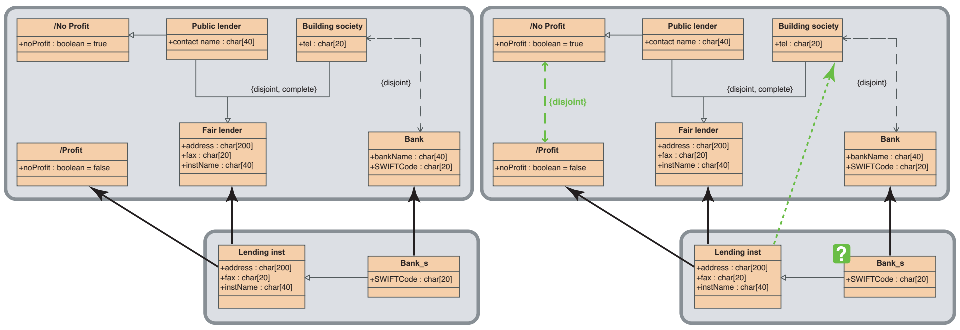
(a) Without deductions (b) With deductions

Fig. 1: The first integration scenario.

The lower ontology of Figure 1a describes a slightly different perspective about the same domain. Still, there are Italian companies and their contact persons (but now without any cardinality constraint, and without mentioning that contact persons of companies should be employees), plus the specification that the companies are either ISO certified or not—here the ISO companies are identified by the code of their ISO certification institution. In addition, the ontology includes the view class `Sales Rep`, which is completely defined by means of its attributes together with the view expression stating that sales representative is the range of the `contacts` association. Note that the view definition can be written in any *reasonable* textual ontology language, such as an OCL constraint, or an OWL axiom, or a SQL check constraint, or first order logic sentence. The view definition mechanism is the hook that allows to use the full power of the ontology language—if the user wants. Most of the ontologies will not need to use this hook, and they will be more directly understandable by the engineers. In the case when subtle integration constraints have to be written, views will come in handy, by providing an expressive language to the engineers in a way which is perfectly integrated with the diagrammatical paradigm proposed here.

Figure 1a includes also the mappings between the two ontologies. You can see that the Italian ISO companies in the top ontology are declared to be the same as the (Italian) ISO companies in the lower ontology, and that the Italian companies in both ontologies are declared to be equivalent as well. Moreover, the `contacts` association in the lower ontology is declared to specialise the homologous association in the top ontology. Inter-ontology mappings are declared by simply drawing directed links between pairs of classes (or pairs of associations) belonging to different ontologies; these can state either equivalence, or containment, or disjointness.

Now, the whole picture seems very reasonable to any ontology engineer; however there are interesting, unexpected, and clarifying consequences that our design tool will automatically draw—still in a diagrammatic fashion. These are shown in Figure 1b.



(a) Without deductions (b) With deductions

Fig. 2: The second integration scenario.

The first consequence relates to the equivalence stated between the two Italian ISO certified company class definitions in the two ontologies. The two classes have a type incompatibility in the attribute `isoCert`: one is declared to be a boolean value, while the other is declared to be a string of ten characters. Indeed, the system deduces that if such an integration has to be taken seriously, then the two classes have to be empty in any possible context, since an object in one context which, say, represents an Italian ISO Company by having an attribute `isoCert` with the value true, can not be at the same time an instance of a class whose `isoCert` is declared to be of an incompatible type (i.e., string). Therefore, such an instance can not exist, and, as a matter of fact, no instances of the two classes can exist at all. This first deduction by the tool (indicated by the question marks on the corner of the two classes) is actually a hint to the designer to actually take care of this *data reconciliation* problem, by, for example, providing local conversion functions between the two attribute types. Please note that the tool also correctly derives the fact that any object which is instance of the Italian Company class (in any of the two contexts/ontologies) should also be an instance of a non ISO company. In fact, since no Italian ISO companies can exist in the current version of the scenario, any Italian company will necessarily be a non ISO certified company. This is made explicit by the dashed equivalence link added by the tool between the `Italian Company` class and the `Non ISO Company` class in the lower ontology.

If we go on with the analysis of the deductions made by the tool, we see that a stricter cardinality constraint has been deduced: now any Italian company can have at most one contact person (in the sense of the lower ontology)—this is the `[0..1]` cardinality constraint found at the right end of the `contacts` association. The system has deduced this stricter constraint by observing that Italian companies are companies, which have exactly one contact person (in the sense of the top ontology); moreover, each Italian company should have contact persons (in the lower ontology sense) among the contact persons in the top ontology sense. Therefore, no Italian company can have more than one contact person (in the lower ontology sense). The lower bound is not derived since the specialisation of the `contacts` association may not necessarily consider all the Italian companies. So, this is an example of a deduction which is not just an IS-A link or an inconsistent class, which are the only kind of deductions that the most advanced ontology design tools (like, e.g., OILEd, or Protege) are capable of.

Another deduction which can not be done by any other ontology design tool is the one which makes explicit a `contacts` association in the lower ontology between Italian companies and sales representative, plus the now stricter cardinality constraint stating that each Italian company has exactly one sales representative. Please note how powerful this deduction mechanism is: an isolated class is automatically fully put in context, by considering all the possible constraints which may relate it to the other terms of the integrated ontologies. As a matter of fact it can be proved that the design tools derives *all* (and only) the implied constraints representable within the diagrammatic ontology language.

Finally, we note that the tool derives also that sales representative are both contact persons in the top ontology sense and contact persons in the lower ontology sense.

All these deductions may help the ontology engineer in validating the design—if the derived constraints make sense to the engineer; they may help in suggesting changes; or they may show serious but subtle conceptual mistakes. The next case scenario shown in Figure 2a is an example of the latter case.

In this new integration scenario, the top ontology describes fair lenders which are partitioned into public lenders and building societies. Public lenders are no profit companies, and in addition it is stated that banks are not building societies.

In the lower less detailed ontology, we have the generic class of lending institutions which specialises into the bank class. We also assume that actually the lower ontology, in spite of the fact that it uses more generic terms, describes a world which is actually a portion of the world described more accurately by the top ontology.

A very natural integration between the two ontologies is pursued by the ontology engineer: she/he states that banks of the lower ontology are among the banks of the top ontology, and that lending institutions of the lower ontology are fair lenders and profit companies as defined in the top ontology.

The consequences of this integration attempt are immediately drawn by the tool as depicted in Figure 2b. As a first (more or less obvious) deduction we can observe that the profit and the no profit classes are derived to be disjoint, as expected. However, it turns out—from the big question mark at the corner of the `Bank_s` class—that no banks can exist according to the lower ontology! This is somehow unexpected, since we thought we were playing a rather simple game in this case. Why is this? A quick glance at the attribute types shows that they are perfectly compatible this time. The reason is the following. First of all, we can derive that lending institutions are building society (as pointed out by the tool); in fact, lending institutions are fair lenders, which can be either public lenders or building societies. On the other hand we have that lending institution are profit companies, which are provably disjoint from public lenders. therefore, any lending institution has necessarily to be a building society. At this point, we are closer to the answer to our original question about the inconsistency of the `Bank_s` class. Those lower ontology banks are at the same time a kind of top ontology banks and building societies (by transitivity). However the two latter classes are disjoint, hence no common instance can exist—i.e., no bank can be a lending institution according to this integration system.

Of course, there should be something wrong the way the two ontologies have been integrated, and this calls for a revision of the mappings. The engineer should either omit the mapping stating that lending institutions are necessarily profit companies, or the mapping stating that lending institutions are necessarily fair lenders. In both cases, the outcome will be acceptable by the engineer, and she/he should choose the one that best fits further analysis of the domain that she/he may have done after this unexpected discovery.

3 The Ontology Editor

The Ontology Editor works on *projects*, which may contain one or more UML class diagrams. The diagrams are referred as *models*. Multiple projects can be opened at the same time, but objects cannot be moved across them. Only one project is visible at a time and the editing of each project is independent. The user can switch between different projects using the tabs at the top of the project area. Classes are represented by boxes and n-ary associations by diamonds. Associations may have so-called association classes specifying their attributes. IsA relationships are represented as arrows with a disc in the middle (e.g. see `MobileCall` and `Call`).

The tool does not implement special visual techniques for handling very large ontologies. The tasks that it supports, i.e. authoring of concept description and structuring the ontology, are not aimed at working simultaneously with thousands of concepts. However, a set of functionalities that are very useful in managing such ontologies are available. First, the interface is zoomable, that is, the level of detail and size of the icons that represent the model can be smoothly changed by pressing the right mouse button and dragging left to zoom our or right to zoom in. Also, the window can be panned by pressing the middle button and dragging. This allows the user to focus the attention in a specific region of the ontology. There are also two dedicated buttons for zooming: one will show the complete graph, and the other will zoom in to show the selected elements. Selection works by left-clicking on icons or by left click and drag; also, there is a button for expanding the selection to all connected nodes, which is very useful in combination with the zoom-to-selection button. Finally, custom automatic layout algorithms for ontologies are under development. These combine known layout algorithms for drawing large graphs, with special conventions used in ontologies, like IsAs hierarchies are drawn top-down and associations are drawn in the middle of related concepts. New metrics to measure the "quality" of ontology graphs were developed for this purpose.

Editing Models Most of the model editing is done in the project panel, where each model in the project is displayed in a separate model panel. In addition, two dialogues are used to elicit additional information about model objects. The attribute domain dialogue allows the domain of attributes to be set. The definition dialogue enables the characterisation of a class or association by means of a view written in the language described in the next Section.

Objects can be created by selecting the appropriate button in the toolbar, or an entry under the Diagram menu, or a contextual menu in the project area. Most object-creating operations require further inputs to complete the operation. Usually, the user is requested to select an existing object in the diagram (e.g. during the creation of an IsA relationship). In this case, the system will highlight only objects in the diagram suitable for the specific operation.

All the objects of the diagram have a name. Upon their creation the system allocates a new fresh name, which can be edited by the user. To improve the identification of the nodes, when icons become smaller because of the zoom level, all the nodes show their name on a tool-tip when the mouse is hovering over them. Names are scoped by the model they belong; e.g. classes with the same name in different models are considered different.

Metadata fields can be associated to every kind of objects. These fields are ignored in the reasoning process.

The creation of a new class adds a new box in the diagram with a new default name. Every class can optionally have attributes. Attributes are added and edited by means of a specific attribute dialogue. Similar to classes, attributes of the same name in different models are considered different. Attributes of the same name within the same model represent the same attribute. For each attribute, a domain should be indicated. There the set of possible domains is not predefined, and the user is allowed to enter an arbitrary name. Unlike the classes and associations, domains have a global context. Therefore, domains of the same name in different models are considered be the same.

Associations are created by default with no roles. N-ary associations can be specified by adding new roles to existing ones. The creation of a new association introduces a corresponding association class, which can be edited as a normal class (e.g. it can have attributes).

Adding new roles to an existing association requires the user to select the association and a class which restricts the domain of the argument of the association corresponding to the role. Similar to class and associations names, role names have a model scope.

For example, assume there are two models `M1` and `M2`, each one with a binary association `lives` having the roles `subject` and `object`. Note that, being association scoped over models, from the global perspective there are two associations `M1:lives` and `M2:lives`. Now, the modelling of the domain requires that `M2:lives` is more specific (i.e. a subset) of `M1:lives`. Since also role names are scoped over each model, overall there are four different roles. Therefore, the more specific association (`M2:lives`) *inherits* the roles of the general one, ending up being of arity four (namely the roles `M2:object`, `M2:subject` and `M1:object`, `M1:subject`).

Roles denote the connection of a class to an association and are also used to express the cardinality constraints of a class in an association. A role may have two constraints: *totality*, or the minimum cardinality, and *uniqueness* representing the maximum cardinality. In the current version of the system, the numbers expressing cardinality are restricted to be 0 and 1. A minimum cardinality of 1 indicates that all instances of a class must participate in the association at least once (i.e. mandatory constraint). A maximum cardinality of 1 indicates that all instances of a class can only participate once in the association (i.e. functional constraint).

Within a project, equivalence and subset role mappings can be defined between roles in the same or different models. These allow a better characterisation of the relationship between associations across different models. In the former example, `M2:lives` can be set as a binary association by saying that `M1:object` contains `M2:object`, and that `M1:subject` contains `M2:subject`.

The system enables the user to specify inheritance relationships among classes and associations. The relationships can be arbitrary (e.g. cycles are allowed) provided that classes can only inherit from classes, and associations from associations. Formally, the inheritance is expressed by the inclusion (subclass) constraint.

On the diagram, inheritance is specified by means of IsA links (in the diagram indicated by arrows with a circle in the middle) connecting nodes. IsA links can be specified one-to-one, or many-to-one. The latter groups together more than one (association) class and restrict all of them to be a subclass of the link target.

The possibility of grouping more than one descendant, not only provides way of visually organising the layout of the model; but enables the user to specify additional constraints among the (association) classes. In particular, the *covering* and *disjointness* constraints. The first one expresses the fact that the (association) class is equivalent to the union of the specified descendant, while the second constraints the grouped (association) classes to be mutually disjoint.

Note that disjointness among classes is not assumed by default; so, in absence of a specific constraint, (association) classes may overlap.

Inter-Model Axioms Additional constraints among classes and associations can be expressed by means of intra- as well as inter-model axioms. The Ontology Editor provides four types of axioms: Node Definition, Equivalence, Subsumption and Disjointness. As discussed in Section 1 these constraints provide a powerful modelling tool in the context of data integration and ontology mapping.

Each class and association can be fully defined by means of a view expression. The view expression language is more expressive that the diagrammatic definition language, so enables the expert user to add constraints that cannot be expressed by the UML diagram alone.

The adopted view language is based on the DLR description logic. A definition has a global context, meaning it can express inter-model relationships as well as intra-model relationships. The view language includes two syntactic sorts: one for classes and one for associations. Full boolean operators are allowed, plus a *selection* operator (selecting tuples in an association with a specific class type in some named role argument) and a unary *projection* operator (projecting an association over a named role argument). A generalised projection operator with cardinality restrictions is available as well.

Since a definition can refer to objects in different models, a name-prefix is used in definitions to distinguish objects with the same name but from different models. The name-prefix used is the model's name followed by a colon symbol. For example, `class1` in `Model1` and `class1` in `Model2` would be referred to as `Model1:class1` and `Model2:class1` respectively.

Any two (associations) classes in any model can be related by semantic relationships stating their equivalence, subsumption, or disjointness. Creating one of these relationships requires the user to specify source and target node. The system prevents the creation of a relationship between non-homogeneous nodes by restricting the scope of the second node to be selected.

Exporting and Importing Projects ICOM projects can saved and retrieved in an own XML format, preserving the meaning of all elements including view definitions. It is also possible to import UML class diagrams saved in the XMI format. The tool only recognises the subset of XMI determined by classes, associations, attributes, roles and primitive datatypes defined within an UML model. Functional and mandatory constraints on roles are the only type of imported constraints. Aggregation relationships in the UML model are ignored. We are currently working on exporting projects in XMI files, but this translation would be necessarily carried out with some loss of meaning because, for example, not all view definitions can be expressed in XMI even with attaching OCL expressions to the model elements (e.g. names in OCL expressions have a scope which is local to a given class, rather than global as in ICOM).

4 Automated Reasoning

Although the Ontology Editor can be used as a standalone modelling tool, exploiting its full capabilities requires the coupling of the system with a Description Logic reasoner. Without such an automated reasoning tool the Ontology Editor would be unable to perform *deduction-complete* automated reasoning over the models. As we noted, this includes checking class and relationship consistency, discovering implied class and relationship inter-relations (e.g., subsumption) or cardinality constraints, and in general discovering any implied but originally implicit class diagram graphical construct.

Instead of implementing its own dedicated reasoner, the Ontology Tool can exploit any DIG enabled DL reasoner (see [2]). Being DIG a standardised communication protocol, the user can choose the most suitable DL reasoner (e.g. the one used by other in-house project), or upgrade to the latest version of the preferred reasoner without being forced to upgrade to a different version of the Ontology Editor.

The so called *verification process* can be computationally expensive, so it is activated only on user's request. This process includes the following operations. The selected project is encoded into an appropriate Description Logics knowledge base and shipped to the DIG reasoner. Each class, association in the project is checked for satisfiability (i.e. non-emptiness). For each class, association in the project, its equivalent peers, and super-classes are determined. For each class-role-association triple, the system calculates the stricter minimum and maximum cardinality constraints. To perform these operations, the system formulates a sequence of queries to be sent to the DIG reasoner. Accordingly to the received answers the Ontology Editor infers properties of the models in the project. To perform these operations, the system formulates a sequence of queries to be sent to the DIG reasoner, which is linear in the number of project elements. Accordingly to the received answers the Ontology Editor infers properties of the models in the project. The algorithm for this inference is quadratic in the number of concepts and roles, and linear in the number of axioms and IsA links. Thus, the tool can reasonable manage projects with several hundreds of elements, calling a current state-of- the-art reasoner.

After the verification process, the system provides the user with a visual account of the deductions by modifying the appearance of the model diagrams in the project. All unsatisfiable objects will appear in red in the model diagrams. An object is unsatisfiable when necessarily describes an empty set of tuples of objects. Additional non explicit deductions will appear in green, to be distinguished from the user specified elements of the diagrams. Semantically equivalent objects are connected with newly inserted equivalent axiom links. Objects discovered to hold an inclusion relationships between them are connected with subsumption axiom links. Cardinality constraints which are stricter than those originally specified. Although the deductions are displayed on the actual diagrams, it is up to the user to decide whether they should be permanently added to the models or discarded. The rational behind this behaviour is that the automated reasoning process may detect unwanted deductions caused by a wrong modelling of the domain. In this case the user should correct the project before any subsequent editing. Another reason is that, in spite of the fact that only the non-trivial deductions are presented, the user is satisfied by the fact that they are implicit without the need of having them explicitly asserted.

The user can discard the deductions and the entire project will be returned to its original state (and any information about unsatisfiability will be discarded). Editing one of the models in the project will also discard the deductions before the editing is carried out. Alternatively, the equivalence, subsumption association, and role cardinality deductions can be added permanently to the project by committing them.

5 Conclusions and Future Works

In this paper we presented ICOM, an advanced conceptual modelling tool grounded on more than ten years of research on the use of automated reasoning to support the development and integration of ontologies. ICOM employes a diagrammatic based language to represent most of the constructs used in ontology design; although it enables the use of non graphical ontology languages, experience with users demonstrates that the design of the diagrammatic language is sufficiently expressive to describe rich domains. Moreover, deductions are expressed within the same diagrammatic language, providing a uniform view over design and analysis of models.

By means of use cases we demonstrated the importance of exploiting basic reasoning tasks (such as subsumption) in order to provide richer information on ontologies.

This is a crucial step towards guaranteeing the quality of the ontologies designed using a tool like ICOM.

The research and development of ICOM continues on two main tracks: from one side we are improving the modelling workflow by considering alternative modelling languages and reasoning services, while on the other hand we are enhancing the user experience by improving the graphical user interface and the interoperability.

We are currently considering the adoption of modelling features from ORM [5] conceptual modelling methodology and representation. Its adoption would have the advantage of leveraging the vast research which has been carried on supporting the user in the modelling tasks, including the integration of natural language generation. The use of ORM modelling style would require also a redesign of the reasoning tasks in order to align the inferences to the new graphical representation.

On the interface we are improving the automatic layout algorithms and working on the support of undo actions. We also plan to include a role browser tab to show the role hierarchy in the same style of the class browser. Moreover, we are improving the interoperability with other tools by tackling the import and export compatibility with XMI and OWL.

References

[1] S. Bechhofer, I. Horrocks, C. Goble, and G. Stevens. Oiled: a reason-able ontology editor for the semantic web. In *Proceedings of KI2001*, pages 396–408. Springer-Verlag, 2001.

[2] S. Bechhofer, R. Möller, and P. Crowther. The dig description logic interface. In *Proceedings of DL 2003*, volume 81 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[3] P. Fillottrani, E. Franconi, and S. Tessaris. The new icom ontology editor. In *Proceedings of DL 2006*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[4] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modeling. In *Proceedings of the 7th International Workshop on Knowledge Representation meets Databases (KRDB 2000)*, volume 29 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2000.

[5] T.A. Halpin, A.J. Morgan, and T. Morgan. *Information modeling and relational databases*. Morgan Kaufmann series in data management systems. Elsevier/Morgan Kaufman Publishers, 2008.

[6] M. Horridge, S. Bechhofer, and O. Noppens. Igniting the owl 1.1 touch paper: The owl api. In *Proceedings of OWLED 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[7] M. Jarke, C. Quix, D. Calvanese, M. Lenzerini, E. Franconi, S. Ligoudistianos, P. Vassiliadis, and Y. Vassiliou. Concept based design of data warehouses: The dwq demonstrators. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, page 591. ACM, 2000.

[8] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The protégé owl plugin: An open development environment for semantic web applications. In *Proceedings of ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*. Springer, 2004.

[9] T. Liebig, M. Luther, O. Noppens, M. Rodriguez, D. Calvanese, M. Wessel, M. Horridge, S. Bechhofer, D. Tsarkov, and E. Sirin. Owllink: Dig for owl 2. In *Proceedings of OWLED 2008*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[10] C. Lutz, F. Baader, E. Franconi, D. Lembo, R. Möller, R. Rosati, U. Sattler, B. Suntisrivaraporn, and S. Tessaris. Reasoning support for ontology design. In *Proceedings of OWLED 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

Efficient Reasoning in Combinations of $\mathcal{EL}$ and (Fragments of) $\mathcal{FL}_0$

Francis Gasse[1,2] and Viorica Sofronie-Stokkermans[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] Universität des Saarlandes, Saarbrücken, Germany

Abstract. We study possibilities of combining (fragments) of the lightweight description logics $\mathcal{FL}_0$ and $\mathcal{EL}$, and identify classes of subsumption problems in a combination of $\mathcal{EL}$ and Horn-$\mathcal{FL}_0$, which can be checked in PSPACE resp. PTIME. Since $\mathcal{FL}_0$ allows universal role restrictions and $\mathcal{EL}$ allows existential role restrictions, we thus have a framework where subsumption between expressions including both types of role restrictions (but for disjoint sets of roles) can be checked in polynomial space or time.

1 Introduction

Description logics [5] are a family of knowledge representation formalisms that can model the terminological knowledge of a given domain; they are, for instance, the logical foundation of the W3C language for the Semantic Web. Their most interesting feature is that they aim at maximizing expressive power while retaining decidability. However, with the size of the ontologies appearing in many applications, decidability alone is not enough because the complexity of the reasoning procedures combined with the size of the ontologies makes reasoning too costly. This consideration triggered the development of lightweight sub-families of description logics. Among them, we mention $\mathcal{EL}$ (which only allows the use of conjunction and existential role restrictions) [1] and some of its extensions such as $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$ [2, 4, 3]. These logics can model some very interesting domains sufficiently well to be used widely, for example in the SNOMED ontology [16]. Another lightweight description logic is $\mathcal{FL}_0$ (which only allows the use of conjunction and universal role restrictions). While subsumption without TBoxes in $\mathcal{FL}_0$ is decidable in PTIME, its subsumption problem is in PSPACE for standard terminologies and EXPTIME for general terminologies [8, 4]. Since some very interesting forms of knowledge require universal restrictions in order to be modeled adequately, recent research has identified tractable fragments of $\mathcal{FL}_0$, such as the Horn-$\mathcal{FL}_0$ fragment (defined by syntactic restrictions) for which the subsumption problem is in PTIME [9].

A combination of $\mathcal{EL}$ and (fragments of) $\mathcal{FL}_0$ is clearly interesting because of the added expressivity it offers. At the same time, if we allow an unrestricted combination we lose the lower complexity of the components. In this paper we

Table 1. Constructors and their semantics

Constructor name	Syntax	Semantics
negation	$\neg C$	$D^{\mathcal{I}} \backslash C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \mid \exists y((x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}})\}$
universal restriction	$\forall r.C$	$\{x \mid \forall y((x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$

present a way to combine these description logics such that we can verify subsumption between two mixed concept expressions w.r.t. TBoxes efficiently, and identify situations in which this can be done in PSPACE, resp. PTIME.

Structure of the Paper. In Sect. 2 we give general definitions and introduce the description logics $\mathcal{ALC}, \mathcal{EL}$ and $\mathcal{FL}_0$ and their combination. Sect. 3 presents the algebraic semantics for each logic and their combination. Sect. 4 presents generalities on local theory extensions and hierarchical reasoning (which we use in our approach). These methods are used in Sect. 5, where we present possibilities of hierarchical reasoning in a combination of $\mathcal{EL}$ and (fragments of) $\mathcal{FL}_0$.

2 Description Logics

The central notions in description logics are concepts and roles. In any description logic a set N_C of *concept names* and a set N_R of *roles* is assumed to be given. Complex concepts are defined starting with the concept names in N_C, with the help of a set of *concept constructors*. The semantics of description logics is defined in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, and the function $\cdot^{\mathcal{I}}$ maps each concept name $C \in N_C$ to a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Table 1 shows the constructor names used in $\mathcal{ALC}$ and their semantics. The extension of $\cdot^{\mathcal{I}}$ to concept descriptions is inductively defined using the semantics of the constructors.

Terminology. A *terminology* (TBox, for short) is a finite set of *primitive concept definitions* of the form $C \equiv D$, where C is a concept name and D a concept description; and *general concept inclusions* (GCI) of the form $C \sqsubseteq D$, where C and D are concept descriptions. A TBox which only contains primitive concept definitions and every concept name is defined at most once is called *standard.* (As definitions can be expressed as double inclusions, by TBox (or general TBox) we will refer to a TBox consisting of general concept inclusions only.) An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies:

– all concept definitions in $\mathcal{T}$, i.e. $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all definitions $C \equiv D \in \mathcal{T}$;
– all general concept inclusions in $\mathcal{T}$, i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every $C \sqsubseteq D \in \mathcal{T}$.

Constraint Box. A *constraint box* (CBox, for short) consists of a TBox $\mathcal{T}$ and a set RI of role inclusions of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$. (We will view CBoxes as unions $GCI \cup RI$ of general concept inclusions (GCI) and role inclusions (RI).) An interpretation $\mathcal{I}$ is a model of the CBox $\mathcal{C} = GCI \cup RI$ if it is a model of GCI and satisfies all role inclusions in $\mathcal{C}$, i.e. $r_1^{\mathcal{I}} \circ \ldots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for all $r_1 \circ \ldots \circ r_n \sqsubseteq s \in RI$.

Definition 1. *Let C_1, C_2 be two concept descriptions.*

- *If $\mathcal{T}$ is a TBox, we say that C_1 is subsumed by C_2 w.r.t. $\mathcal{T}$ (denoted $C_1 \sqsubseteq_{\mathcal{T}} C_2$) i ff$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$.*
- *If $\mathcal{C}$ is a CBox, then $C_1 \sqsubseteq_{\mathcal{C}} C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{C}$.*

The simplest propositionally closed description logic is $\mathcal{ALC}$ which allows for conjunction, disjunction, negation and existential and universal role restrictions. For description logics that allow full negation, subsumption tests w.r.t. TBoxes or CBoxes are reducible to satisfiability testing for concepts (i.e. checking if there exists a model of the TBox resp. CBox for which the interpretation of the concept is non-empty). It is well-known that for $\mathcal{ALC}$ subsumption checking (w.r.t. TBoxes and CBoxes) is in EXPTIME (cf. [5]). For *lightweight description logics* which do not allow negation, things are different: The main reasoning task is subsumption testing, which is the problem we consider in this paper.

We now define the fragments of the description logics $\mathcal{FL}_0$ used in this paper as well as the description logic $\mathcal{EL}$. [3]

The Description Logic $\mathcal{FL}_0$. $\mathcal{FL}_0$ is a lightweight description logic that only allows as concept constructors conjunction, universal role restrictions, and top concept. The subsumption problem w.r.t. general TBoxes is known to be in EXPTIME [4]. Fragments of $\mathcal{FL}_0$ resp. specific classes of subsumption for which the complexity is known to be lower include:

- **Subsumption w.r.t. standard TBoxes** has PSPACE complexity [8].
- **Subsumption w.r.t. acyclic TBoxes** is co-NP complete (where an acyclic TBox is a standard TBox that does not contain concept definitions $A_1 \equiv C_1, \ldots, A_k \equiv C_k$ such that $A_{i+1 \bmod k}$ is used in C_i for all $i < k$ [10]).
- **Horn-$\mathcal{FL}_0^+$** [9] is a variant of $\mathcal{FL}_0$ that both extends and restricts its expressivity in such a way that the subsumption problem remains in PTIME. It restricts $\mathcal{FL}_0$ axioms to the form shown in Table 2. The form of the axioms is limited in such a way that they can be rewritten into 3-variable function-free Horn-logic. It follows from this correspondence that verifying consistency of a Horn-$\mathcal{FL}_0^+$ knowledge base can be done in polynomial time. A Horn-$\mathcal{FL}_0$ TBox (CBox) consists only of inclusions of the form indicated in the first two lines of Table 2.

The Description Logic $\mathcal{EL}^+$. The description logic $\mathcal{EL}$ [1] allows as concept constructors only conjunction, existential role restrictions, and the bottom concept. $\mathcal{EL}^+$ [2, 4, 3] additionally allows for nominals and role composition. For $\mathcal{EL}^+$, checking CBox subsumption can be done in PTIME [4, 2].

[3] For the sake of simplicity, everywhere in what follows we consider fragments of these logics without nominals and without ABoxes.

$A \sqsubseteq C$	$\top \sqsubseteq C$	$R \sqsubseteq T$	$A \sqsubseteq \forall R.C$
$A \sqcap B \sqsubseteq C$	$A \sqsubseteq \bot$	$R \circ S \sqsubseteq T$	
$R(i, j)$	$A(i)$	$i \approx j$	

Table 2. Normal form for Horn $\mathcal{FL}_0^+$. A, B, C are names of atomic concepts; R, S, T are (possibly inverse) role names.

2.1 Combining $\mathcal{FL}_0$ and $\mathcal{EL}$

Let N_C be a set of concept names, and $N_R, N_{R'}$ be disjoint sets of role names. We propose a combination of $\mathcal{EL}$ (with roles in N_R) and $\mathcal{FL}_0$ (with roles in $N_{R'}$). The problem we study for such combinations is subsumption between concept expressions using constructs from both logics (such that existential restriction is used only for roles in N_R and universal restriction only for roles in $N_{R'}$) w.r.t. mixed TBoxes, consisting of an $\mathcal{EL}$ part and an $\mathcal{FL}_0$ part. We allow these TBoxes to share concept names (but the role names used in each type of axioms have to be disjoint). We have to impose the restriction that $N_R \cap N_{R'} = \emptyset$ in order to be sure that fine-grained complexity results can be obtained for TBox subsumption in such combinations, since the description logic combining these features freely, $\mathcal{ALEU}$, has an EXPTIME complexity for the subsumption problem w.r.t. TBox[4].

Definition 2. *A* mixed TBox *is a TBox $\mathcal{T} = \mathcal{T}_E \cup \mathcal{T}_F$ which consists of two distinct parts: A set $\mathcal{T}_E$ of $\mathcal{EL}$ GCI (with role names N_R), and a set $\mathcal{T}_F$ of $\mathcal{FL}_0$ GCI (with role names $N_{R'}$), each respecting the syntactic restrictions imposed by their logic. In a* mixed TBox with acyclic $\mathcal{FL}_0$ part*, $\mathcal{T}_F$ is a standard acyclic TBox; in a* mixed TBox with standard $\mathcal{FL}_0$ part*, $\mathcal{T}_F$ is a standard TBox.*

We will use the names $\mathcal{EL}$-TBox and $\mathcal{FL}$-TBox to denote the set of $\mathcal{EL}$ (resp. Horn-$\mathcal{FL}_0$) inclusion axioms in a mixed TBox.

3 Algebraic Semantics

We assume known notions such as partially-ordered set, semilattice, lattice and Boolean algebra. For further information cf. [11]. We define a translation of concept descriptions into terms in a signature naturally associated with the set of constructors. For every role name r, we introduce unary function symbols, $f_{\exists r}, f_{\forall r}$. The renaming is inductively defined by:

- $\overline{C} = C$ for every concept name C;
- $\overline{\neg C} = \neg \overline{C}$; $\quad \overline{C_1 \sqcap C_2} = \overline{C}_1 \wedge \overline{C}_2$, $\quad \overline{C_1 \sqcup C_2} = \overline{C}_1 \vee \overline{C}_2$;
- $\overline{\exists r.C} = f_{\exists r}(\overline{C})$, $\quad \overline{\forall r.C} = f_{\forall r}(\overline{C})$.

There exists a one-to-one correspondence between interpretations $\mathcal{I} = (D, \cdot^{\mathcal{I}})$ and Boolean algebras of sets $(\mathcal{P}(D), \cup, \cap, \neg, \emptyset, D, \{f_{\exists r}, f_{\forall r}\}_{r \in N_R})$, together with

[4] This follows from the fact that $\mathcal{ALEU}$ can simulate $\mathcal{ALC}$ [7].

valuations $v : N_C \to \mathcal{P}(D)$, where $f_{\exists r}, f_{\forall r}$ are defined, for every $U \subseteq D$, by:

$$f_{\exists r}(U) = \{x \mid \exists y((x, y) \in r^{\mathcal{I}} \text{ and } y \in U)\}$$
$$f_{\forall r}(U) = \{x \mid \forall y((x, y) \in r^{\mathcal{I}} \Rightarrow y \in U)\}.$$

Consider the following classes of algebras:

– BAO_{N_R}, the class of all Boolean algebras with operators $(B, \vee, \wedge, \neg, 0, 1, \{f_{\exists r}, f_{\forall r}\}_{r \in N_R})$, where
  - $f_{\exists r}$ is a join-hemimorphism, i.e. $f_{\exists r}(x \vee y) = f_{\exists r}(x) \vee f_{\exists r}(y)$, $f_{\exists r}(0) = 0$;
  - $f_{\forall r}$ is a meet-hemimorphism, i.e. $f_{\forall r}(x \wedge y)=f_{\forall r}(x) \wedge f_{\forall r}(y)$, $f_{\forall r}(1)=1$;
  - $f_{\forall r}(x) = \neg f_{\exists r}(\neg x)$ for every $x \in B$.

– $\mathsf{BAO}^{\exists}_{N_R}$ the class of boolean algebras with operators $(B, \vee, \wedge, \neg, 0, 1, \{f_{\exists r}\}_{r \in N_R})$, such that $f_{\exists r}$ is a join-hemimorphism.

– $\mathsf{BAO}^{\forall}_{N_{R'}}$ the class of boolean algebras with operators $(B, \vee, \wedge, \neg, 0, 1, \{f_{\forall r}\}_{r \in N_{R'}})$, such that $f_{\forall r}$ is a meet-hemimorphism.

– $\mathsf{SLO}^{\exists}_{N_R}$ the class of all $\wedge$-semilattices with operators $(S, \wedge, 0, 1, \{f_{\exists r}\}_{r \in N_R})$, such that $f_{\exists r}$ is monotone and $f_{\exists r}(0) = 0$.

– $\mathsf{SLO}^{\forall}_{N_{R'}}$ the class of all $\wedge$-semilattices with operators $(S, \wedge, 0, 1, \{f_{\forall r}\}_{r \in N_{R'}})$, such that $f_{\forall r}$ is a meet-hemimorphism and $f_{\forall r}(1)$=1.

– $\mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}$ the class of all $\wedge$-semilattices with operators $(S, \wedge, 0, 1, \{f_{\exists r}\}_{r \in N_R}, \{f_{\forall r}\}_{r \in N_{R'}})$, such that $f_{\exists r}$ is monotone and $f_{\exists r}(0) = 0$, and $f_{\forall r}$ is a meet-hemimorphism and $f_{\forall r}(1)$=1.

It is known that the TBox subsumption problem for $\mathcal{ALC}$ can be expressed as a uniform word problem for Boolean algebras with suitable operators (cf. e.g. [6]).

Let RI, RI' be sets of axioms of the form $r \sqsubseteq s$ and $r_1 \circ r_2 \sqsubseteq r$, with $r, s, r_1, r_2 \in N_R$ (resp. $r, s, r_1, r_2 \in N_{R'}$). We associate with RI, RI' the following set of axioms:

$$\begin{aligned} RI_a = &\{\forall x \quad (f_{\exists r_2} \circ f_{\exists r_1})(x) \leq f_{\exists r}(x) \mid r_1 \circ r_2 \sqsubseteq r \in RI\} \cup \\ &\{\forall x \quad f_{\exists r}(x) \leq f_{\exists s}(x) \mid r \sqsubseteq s \in RI\} \\ RI^{'}_a = &\{\forall x \quad (f_{\forall r_2} \circ f_{\forall r_1})(x) \geq f_{\forall r}(x) \mid r_1 \circ r_2 \sqsubseteq r \in RI'\} \cup \\ &\{\forall x \quad f_{\forall r}(x) \geq f_{\forall s}(x) \mid r \sqsubseteq s \in RI'\} \end{aligned}$$

where $f \circ g$ denotes the composition of the functions f, g. Let $\mathsf{BAO}^{\exists}_{N_R}(RI)$ (resp. $\mathsf{SLO}^{\exists}_{N_R}(RI)$) be the subclass of $\mathsf{BAO}^{\exists}_{N_R}$ ($\mathsf{SLO}^{\exists}_{N_R}$) consisting of those algebras which satisfy RI_a, and $\mathsf{BAO}^{\forall}_{N_{R'}}(RI')$ (resp. $\mathsf{SLO}^{\forall}_{N_{R'}}(RI')$) be the subclass of $\mathsf{BAO}^{\forall}_{N_{R'}}$ ($\mathsf{SLO}^{\forall}_{N_{R'}}$) consisting of the algebras satisfying $RI^{'}_a$.

In [13] we studied the link between TBox subsumption in $\mathcal{EL}$ and uniform word problems in the corresponding classes of semilattices with monotone functions, and in [14] we studied an extension to $\mathcal{EL}^+$. We will present these results here, together with an algebraic semantics for $\mathcal{FL}_0$.

Theorem 1 ([13]) *Assume that the only concept constructors are intersection and existential restriction. Then for all concept descriptions D_1, D_2 and every $\mathcal{EL}^+$ CBox $\mathcal{C}$=GCI$\cup$RI, with concept names $N_C = \{C_1, \dots, C_n\}$:*

$D_1 \sqsubseteq_{\mathcal{C}} D_2$ *iff* $\mathsf{SLO}^{\exists}_{N_R}(RI) \models \forall C_1 \dots C_n((\bigwedge_{C \sqsubseteq D \in GCI} \overline{C} \leq \overline{D}) \to \overline{D_1} \leq \overline{D_2})$.

We give a similar result for $\mathcal{FL}_0^+$.

Theorem 2 *Assume that the only concept constructors are intersection and universal restriction. Then for all concept descriptions D_1, D_2 and every $\mathcal{FL}_0^+$ CBox $\mathcal{C}{=}GCI{\cup}RI$, with concept names $N_C = \{C_1, \dots, C_n\}$:*

$$D_1 {\sqsubseteq_{\mathcal{C}}} D_2 \qquad \textit{iff} \qquad \mathsf{SLO}^{\forall}_{N_{R'}}(RI) \models \forall C_1 \dots C_n ((\textstyle\bigwedge_{C \sqsubseteq D \in GCI} \overline{C} {\leq} \overline{D}) \rightarrow \overline{D_1} {\leq} \overline{D_2}).$$

3.1 Algebraic Semantics for a Combination of $\mathcal{EL}$ and $\mathcal{FL}_0$

Theorem 3 *Assume the only concept constructors are intersection, existential restriction over roles in N_R and universal restriction over roles in $N_{R'}$. Let $\mathcal{T}$ be a mixed TBox consisting of an $\mathcal{EL}$-TBox $\mathcal{T}_E$ (with roles in N_R) and an $\mathcal{FL}_0$-TBox $\mathcal{T}_F$ (with roles in $N_{R'}$), where $N_R {\cap} N_{R'} = \emptyset$. Then for all concept descriptions D_1, D_2 in the combined language, with concept names $N_C = \{C_1, \dots, C_n\}$:*

$$D_1 {\sqsubseteq_{\mathcal{T}}} D_2 \qquad \textit{iff} \qquad \mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}} \models \forall C_1 \dots C_n ((\textstyle\bigwedge_{C \sqsubseteq D \in \mathcal{T}} \overline{C} {\leq} \overline{D}) \rightarrow \overline{D_1} {\leq} \overline{D_2}).$$

Note: The results can be extended in a natural way to $\mathcal{EL}^+$, $\mathcal{FL}_0^+$ and CBoxes (we will then take the combination of the role inclusions RI, RI', and the corresponding subclass $\mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}(RI, RI')$ satisfying the axioms $RI_a \cup RI'_a$).

In what follows we show that we can reduce, in polynomial time and with a polynomial increase in the length of the formulae, the validity tasks w.r.t. $\mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}$ to satisfiability tasks w.r.t. $\mathsf{SLO}^{\forall}_{N_{R'}}$ which can in general be solved in EXPTIME. We obtain the following finer grained results:

- If $\mathcal{T}_F$ is a standard TBox, the subsumption tasks are in PSPACE;
- If $\mathcal{T}_F$ is in the Horn-$\mathcal{FL}_0$ fragment, the reduction generates formulae whose satisfiability can be checked in PTIME.

For obtaining these results, we use the notion of local theory extensions, which is briefly introduced in what follows.

4 Local Theories and Local Theory Extensions

We here consider theories specified by their sets of axioms, and extensions of theories, in which the signature is extended by new *function symbols*. Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, where Σ_0 a set of function symbols, and Pred a set of predicate symbols. We consider extensions $\mathcal{T}_1$ of $\mathcal{T}_0$ with signature $\Pi = (\Sigma, \mathsf{Pred})$, where $\Sigma = \Sigma_0 \cup \Sigma_1$ (i.e. the signature is extended by new function symbols). We assume that $\mathcal{T}_1$ is obtained from $\mathcal{T}_0$ by adding a set $\mathcal{K}$ of (universally quantified) clauses in the signature Π, each of them containing at least a function symbol in Σ_1 and denote this by writing $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$.

Locality. Let $\mathcal{K}$ be a set of (universally quantified) clauses in the signature Π. In what follows, when referring to sets G of ground clauses we assume they are in the signature $\Pi^c = (\Sigma \cup \Sigma_c, \mathsf{Pred})$ where Σ_c is a set of new constants. An extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *local* if satisfiability of a set G of clauses w.r.t. $\mathcal{T}_0 \cup \mathcal{K}$ only depends on $\mathcal{T}_0$ and those instances $\mathcal{K}[G]$ of $\mathcal{K}$ in which the terms starting with extension functions are in the set $\mathsf{st}(\mathcal{K}, G)$ of ground terms which already occur in G or $\mathcal{K}$, i.e. if condition (Loc) is satisfied:

(Loc) For every finite set G of ground clauses $\mathcal{T}_1 \cup G \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \perp$

$$\text{where } \mathcal{K}[G] = \{C\sigma \mid \begin{array}{l} C \in \mathcal{K}, \text{ for each subterm } f(t) \text{ of } C, \text{ with } f \in \Sigma_1, \\ f(t)\sigma \in \mathsf{st}(\mathcal{K}, G), \text{ and for each variable } x \text{ which does not} \\ \text{occur below a function symbol in } \Sigma_1, \sigma(x) = x\}. \end{array}$$

Hierarchical Reasoning. In local theory extensions hierarchical reasoning is possible. All clauses in $\mathcal{K}[G] \cup G$ have the property that the function symbols in Σ_1 have as arguments only ground terms. Therefore, $\mathcal{K}[G] \cup G$ can be purified (i.e. the function symbols in Σ_1 are separated from the other symbols) by introducing, in a bottom-up manner, new constants c_t for subterms $t = f(g_1, \dots, g_n)$ with $f \in \Sigma_1$, g_i ground $\Sigma_0 \cup \Sigma_c$-terms (where Σ_c is a set of constants which contains the constants introduced by flattening, resp. purification), together with corresponding definitions $c_t \approx t$. The set of clauses thus obtained has the form $\mathcal{K}_0 \cup G_0 \cup D$, where D is a set of ground unit clauses of the form $f(g_1, \dots, g_n) \approx c$, where $f \in \Sigma_1$, c is a constant, $g_1, \dots, g_n$ are ground terms without function symbols in Σ_1, and $\mathcal{K}_0$ and G_0 are clauses without function symbols in Σ_1.

For the sake of simplicity in what follows we will always first flatten and then purify $\mathcal{K}[G] \cup G$. Thus we ensure that D consists of ground unit clauses of the form $f(c_1, \dots, c_n) \approx c$, where $f \in \Sigma_1$, and $c_1, \dots, c_n, c$ are constants.

Theorem 4 ([12]) *Let $\mathcal{K}$ be a set of clauses. Assume that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a local theory extension. For any set G of ground clauses, let $\mathcal{K}_0 \cup G_0 \cup D$ be obtained from $\mathcal{K}[G] \cup G$ by flattening and purification, as explained above. Then the following are equivalent:*

(1) $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$.
(2) $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \perp$.
(3) $\mathcal{T}_0 \cup \mathcal{K}_0 \cup G_0 \cup N_0 \models \perp$, *where*

$$N_0 = \{\bigwedge_{i=1}^{n} c_i \approx d_i \rightarrow c \approx d \mid f(c_1, \dots, c_n) \approx c, f(d_1, \dots, d_n) \approx d \in D\}.$$

Theorem 5 ([15]) *The extension of any semilattice-ordered theory with monotone functions is local. In particular, the extension $\mathsf{SLO}^{\forall}_{N_{R'}} \subseteq \mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}$ of the theory of semilattices with meet-hemimorphisms in a set $\{f_{\forall R} \mid R \in N_{R'}\}$ with monotone functions in a set $\{f_{\exists R} \mid R \in N_R\}$, where $N_R \cap N_{R'} = \emptyset$, is local.*

Thus, the method for hierarchical reasoning described in Theorem 4 can be used in this context to reduce the proof tasks in $\mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}$ to proof tasks in $\mathsf{SLO}^{\forall}_{N_{R'}}$. We describe the approach in the next section. For the sake of simplicity, in what follows we use the notation $\exists R.C$ for $f_{\exists R}(C)$ and $\forall S.D$ for $f_{\forall S}(D)$. [5]

[5] In [14] we proved generalized locality results also for extensions with monotone functions satisfying axioms of the form RI_a, so the results can be further extended to give a reduction of proof tasks in $\mathsf{SLO}^{\exists\forall}_{N_R, N_{R'}}(RI, RI')$ to proof tasks in $\mathsf{SLO}^{\forall}_{N_{R'}}(RI')$.

5 The Combination of $\mathcal{EL}$ and $\mathcal{FL}_0$

We consider the subsumption problem for the combination of $\mathcal{EL}$ and $\mathcal{FL}_0$ introduced in Section 3.1 and illustrate the way hierarchical reasoning can be used for reasoning in this combination, and for identifying fragments of this combination and subsumption tasks which can be checked in PSPACE/PTIME. [6]

We first have to *purify* the expressions for which we want to verify subsumption. Consider for instance the subsumption $C \sqsubseteq \exists R.D$, where C and D are resp. an $\mathcal{FL}_0$ and an $\mathcal{EL}$ concept description. To purify it, we add the axiom $D' \equiv \exists R.D$ to the $\mathcal{EL}$-TBox (where D' is a new concept name) and rewrite the subsumption as $C \sqsubseteq D'$. We apply this process in an "inside-out" fashion such that the final result is checking subsumption between concept names w.r.t. to an augmented TBox. This procedure does not affect complexity when we use new names for $\mathcal{EL}$ concept descriptions ($\mathcal{EL}$ allows for equalities and inequalities TBoxes). In what follows, $C[\exists R.C']$ is a notation indicating that C is a concept description in the combination of $\mathcal{EL}$ and $\mathcal{FL}_0$ containing a subterm of the form $\exists R.C'$, $R \in N_R$; the notation $C[C'']$ indicates the concept description obtained by replacing $\exists R.C'$ with C'' in C.

Theorem 6 *Consider the subsumption problem $C[\exists R.C'] \sqsubseteq_{\mathcal{T}} D$ (where C' is an $\mathcal{EL}$ concept description) w.r.t. a mixed TBox $\mathcal{T} = \mathcal{T}_E \cup \mathcal{T}_F$ and the subsumption problem $C[C''] \sqsubseteq_{\mathcal{T}'} D$ w.r.t. the extension $\mathcal{T}'$ of $\mathcal{T}$ with a new concept name C'' together with its definition $C'' \equiv \exists R.C'$. Then the following are equivalent:*

(1) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2) \rightarrow C[\exists R.C'] \leq D$
(2) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2 \quad \wedge \quad C'' \approx \exists R.C') \rightarrow C[C''] \leq D$

This also holds for subsumption problems of the form $C \sqsubseteq D[\exists R.D']$.

Theorem 7 *Consider the subsumption problem $C[\forall S.C'] \sqsubseteq_{\mathcal{T}} D$ (where C' is an $\mathcal{FL}_0$ concept description) w.r.t. a mixed TBox $\mathcal{T} = \mathcal{T}_E \cup \mathcal{T}_F$ and the subsumption problem $C[C''] \sqsubseteq_{\mathcal{T}'} D$ w.r.t. the extension $\mathcal{T}'$ of $\mathcal{T}$ with a new concept name C'' and a definition for it ($C'' \equiv \forall S.C'$). Then the following are equivalent:*

(1) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2) \rightarrow C[\forall S.C'] \leq D$
(2) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2 \ \wedge \ C'' \approx \forall S.C') \rightarrow C[C''] \leq D.$

This also holds for subsumption problems of the form $C \sqsubseteq D[\forall S.D']$.

$\mathcal{FL}_0$ with Standard TBoxes. Assume that we consider a combination of $\mathcal{EL}$ with the fragment of $\mathcal{FL}_0$ with standard TBoxes. Then $\mathcal{T}_F$ is a standard $\mathcal{FL}_0$-TBox, hence also $\mathcal{T}_F \cup \{C'' \equiv \forall S.C'\}$ is a standard TBox.

$\mathcal{FL}_0$ with Acyclic TBoxes. Assume that we consider a combination of $\mathcal{EL}$ with the fragment of $\mathcal{FL}_0$ with acyclic standard TBoxes, i.e. $\mathcal{T}_F$ is a standard

[6] The results can be extended to combinations of $\mathcal{EL}^+$ and $\mathcal{FL}_0^+$ and to subsumption tasks w.r.t. CBoxes. Due to space constraints this extension is not presented here.

acyclic TBox $\{A_i \equiv C_i \mid i = 1, ..., k\}$. Assume that C' does not contain any of the atomic concept names A_i. Since C'' is a new concept name, the $\mathcal{FL}_0$-TBox $\mathcal{T}_F \cup \{C'' \equiv \forall S.C'\}$ is an acyclic TBox. After the elimination of $\exists R.C$ concepts and introduction of new concept names and definitions, the resulting TBox is a standard $\mathcal{FL}_0$-TBox (which is acyclic only if additional acyclicity assumptions are made on $\mathcal{T}_E$).

Horn-$\mathcal{FL}_0$. The restriction imposed on the form of the TBox axioms in Horn-$\mathcal{FL}_0$ prevents purification by adding definitions of the form $C'' \equiv \forall S.C'$ (we cannot allow universal restriction on the left-hand side of an axiom). For the case where we have to purify the left-hand side that causes no problem since if $\forall S.C'$ occurs on the left-hand side we only need to add $C'' \sqsubseteq \forall S.C'$ to the TBox:

Theorem 8 *Consider the subsumption problem $C[\forall S.C'] \sqsubseteq_{\mathcal{T}} D$ (where C' is an $\mathcal{FL}_0$ concept description) w.r.t. a mixed TBox $\mathcal{T} = \mathcal{T}_E \cup \mathcal{T}_F$, and the subsumption problem $C[C''] \sqsubseteq_{\mathcal{T}'} D$ w.r.t. the extension $\mathcal{T}'$ of $\mathcal{T}$ with a new concept name C'' and an inclusion of the form ($C'' \sqsubseteq \forall S.C'$). Then the following are equivalent:*

(1) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2) \rightarrow C[\forall S.C'] \leq D.$
(2) $\mathsf{SLO}^{\exists\forall}_{N_R,N_{R'}} \models (\bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}_E \cup \mathcal{T}_F} C_1 \leq C_2 \ \wedge \ C'' \leq \forall S.C') \rightarrow C[C''] \leq D.$

However, we cannot replace universal restriction on the right-hand side with a name in general which prevents us to purify arbitrary expressions.

Hierarchical Reasoning. Consider the purified form of the problem. We replace all terms of the form $\exists R.C$ in $\mathcal{T}_E$ with a new constant, say $C_{\exists R.C}$. Let Def be the set of all definitions for these new constants, of the form $C_{\exists R.C} \equiv \exists R.C$. Let M_0 be the set of corresponding instances of monotonicity axioms:

$$M_0 = \{C_1 \leq C_2 \rightarrow C_{\exists R.C_1} \leq C_{\exists R.C_2} \mid C_{\exists R.C_i} = \exists R.C_i \in \mathsf{Def}\}.$$

Let $(\mathcal{T}_E)_0$ be the purified form of $\mathcal{T}_E$. By Theorem 4, the following are equivalent:

(i) $\mathsf{SLO}^{\forall\exists}_{N_R,N_{R'}} \models \bigwedge_{(D \sqsubseteq D') \in \mathcal{T}} D \leq D' \rightarrow C_1 \leq C_2$.
(ii) $G_0 \wedge M_0$ is unsatisfiable in $\mathsf{SLO}^{\forall}_{N_{R'}}$, where $G_0 = (\mathcal{T}_E)_0 \wedge \mathcal{T}_F \wedge (\neg(C_1 \leq C_2))_0$.

Note that in the presence of the monotonicity axioms, the instances of the congruence axioms in N_0 (cf. notation in Theorem 4) are redundant.

Theorem 9 *Assume that the only concept constructors are intersection and existential restrictions over roles in N_R and universal restrictions over roles in $N_{R'}$. Assume that we have a mixed TBox, consisting of an $\mathcal{EL}$-TBox $\mathcal{T}_E$ (with roles in a set N_R) and an $\mathcal{FL}_0$-TBox $\mathcal{T}_F$ (with roles in a set $N_{R'}$), where $N_R \cap N_{R'} = \emptyset$. Then for all concept descriptions D_1, D_2 with concept names $N_C = \{C_1, \ldots, C_n\}$ over this signature, the following hold:*

(1) If $\mathcal{T}_F$ is a standard TBox, then:
  (a) For any subsumption problem purification yields a new mixed TBox $\mathcal{T}' = \mathcal{T}'_E \cup \mathcal{T}'_F = \mathcal{T}_E \wedge \mathsf{Def} \wedge \mathcal{T}_F$ with a standard $\mathcal{FL}_0$ part, and after the elimination of $\exists R.C$ concepts, $(\mathcal{T}'_E)_0 \cup \mathcal{T}'_F$ is a standard $\mathcal{FL}_0$ TBox.

(b) Checking whether $D_1 \sqsubseteq_{\mathcal{T}_E \cup \mathcal{T}_F} D_2$ can be done in PSPACE.

(2) If $\mathcal{T}_F$ is a Horn-$\mathcal{FL}_0$ TBox and C is an arbitrary concept description in the combined language and D does not contain terms of the form $\exists R.D_1$, where $R \in N_R$ with subterms of the form $\forall S.D_2$, $S \in N_{R'}$, then:

(a) Purification yields a new mixed TBox with a Horn-$\mathcal{FL}_0$ part; after the elimination of $\exists R.C$ concepts, $(\mathcal{T}'_E)_0 \cup \mathcal{T}'_F$ is a Horn-$\mathcal{FL}_0$ TBox. Since
(i) $C \sqsubseteq_{\mathcal{T}} D_1 \sqcap D_2$ iff($C \sqsubseteq_{\mathcal{T}} D_1$ and $C \sqsubseteq_{\mathcal{T}} D_2$), and
(ii) $\forall S$ commutes with intersections,
we can consider, w.l.o.g. only subsumption problems $D_1 \sqsubseteq_{\mathcal{T}} \forall S_1. \dots \forall S_n.D$, $n \geq 0$, where D_2, D are concept names.

(b) Checking whether $D_1 \sqsubseteq_{\mathcal{T}_E \cup \mathcal{T}_F} D_2$ where $D_2 = \forall S_1. \dots \forall S_n.D$ (where $n \geq 0$ and C, D are concept names) can be done in PTIME.

Proof. (1)(a) and (2)(a) are simple consequences of the purification procedure. Consider the purified form of the problem By Theorems 3 and 4, $D_1 \sqsubseteq_{\mathcal{T}_E \cup \mathcal{T}_F} D_2$ iff $\mathsf{SLO}^{\forall\exists}_{N_R, N_{R'}} \models \bigwedge_{D \sqsubseteq D' \in \mathcal{T}} (D \leq D' \rightarrow D_1 \leq D_2$ iff $G_0 \wedge M_0$ is unsatisfiable in $\mathsf{SLO}^{\forall}_{N_{R'}}$, where $G_0 = (\mathcal{T}_E)_0 \wedge \mathcal{T}_F \wedge (\neg(C_1 \leq C_2))_0$. In order to test the unsatisfiability of the latter problem we proceed as follows. We first note that, due to the convexity of $\mathsf{SLO}^{\forall}_{N_{R'}}$, if $G_0 \wedge M_0 \models \perp$, then there exists a clause $C = (c_1 \leq d_1 \rightarrow c \leq d) \in M_0$ such that $G_0 \models c_1 \leq d_1$ and $G_0 \wedge \{c \leq d\} \wedge M_0 \backslash \{C\} \models \perp$. By iterating the argument above we can always successively entail sufficiently many premises of monotonicity axioms in order to ensure that there exists a set $\{C_1, \dots, C_n\}$ of clauses in M_0 with $C_j = (c^j_1 \leq d^j_1 \rightarrow c^j \leq d^j)$, such that for all $k \in \{0, \dots, n-1\}$, $G_0 \wedge \bigwedge_{j=1}^{k} (c^j \leq d^j) \models \bigwedge c^{k+1}_i \leq d^{k+1}_i$ and $G_0 \wedge \bigwedge_{j=1}^{n} (c^j \leq d^j) \models \perp$. Conversely, if the last condition holds, then $G_0 \wedge M_0 \models \perp$. This means that in order to test satisfiability of $G_0 \wedge M_0$ we need to: (i) test entailment of the premises of M_0 from G_0; when all premises of some clause are provably true we delete the clause and add its conclusion to G_0, and (ii) in the end check whether $G_0 \wedge \bigwedge_{j=1}^{n} (c^j \leq d^j) \models \perp$.

Under the assumptions in (1), every entailment task in (i) and the test in (ii) are in PSPACE. Since space can be reused, the process terminates and is in PSPACE. Under the assumptions in (2), $\mathcal{T}_0 = (\mathcal{T}_E)_0 \cup \mathcal{T}_F$ and G_0 are in Horn $\mathcal{FL}_0$. Therefore, every entailent task in (i) above can be done in PTIME. The task (ii) - for the case that G_0 is derived from a subsumption problem of the form $C \sqsubseteq_{\mathcal{T}} \forall S_1. \dots \forall S_n.D$, where $n \geq 0$, and C, D are concept names, can be translated to a satisfiability test in Horn-$\mathcal{FL}_0$, so it can be done in PTIME. □

6 Conclusion

We identified a class of subsumption problems in a combination of $\mathcal{EL}$ and Horn-$\mathcal{FL}_0$, which can be checked in PTIME. Since $\mathcal{FL}_0$ allows universal role restriction and $\mathcal{EL}$ allows existential role restrictions, we thus have a framework where subsumption between expressions including both types of role restrictions (but for disjoint sets of roles) can be checked in polynomial space or time.

Acknowledgments. We thank the referees for their helpful comments. This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) (see also `www.avacs.org`).

References

1. F. Baader. Terminological cycles in a description logic with existential restrictions. In: G. Gottlob and T. Walsh, editors, *Proc. of the 18th International Joint Conference in Artificial Intelligence*, pages 325–330, Morgan Kaufmann, 2003.
2. F. Baader, C. Lutz, B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? *Journal of Logic, Language and Information* (M4M special issue).
3. F. Baader, S. Brandt, C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* IJCAI-05, Morgan-Kaufmann Publishers, 2005.
4. F. Baader, S. Brandt, C. Lutz. Pushing the $\mathcal{EL}$ envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005.
5. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
6. F. Baader, C. Lutz, H. Sturm, F. Wolter. Fusions of description logics and abstract description systems. *J. Artif. Int. Res.*, 16:1–58, January 2002.
7. F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf. *Reasoning in description logics*, pages 191–236. Center for the Study of Language and Information, Stanford, CA, USA, 1996.
8. Y. Kazakov, H. De Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ with respect to descriptive semantics is PSPACE-complete. *Proc. DL'03*, CEUR-WS, Vol-81, 2003.
9. M. Krötzsch, S. Rudolph, P. Hitzler. Complexity boundaries for horn description logics. *In Proc. AAAI 2007*, pages 22–26. AAAI Press, 2007.
10. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
11. B.A. Davey, H.A. Priestley. *ntroduction to Lattices and Order (2. ed.)* Cambridge University Press, 2002.
12. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In R. Nieuwenhuis, editor, *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2005.
13. V. Sofronie-Stokkermans. Automated theorem proving by resolution in non-classical logics. *Ann. Math. Artif. Intell.*, 49(1-4):221–252, 2007.
14. V. Sofronie-Stokkermans. Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In C. Areces and R. Goldblatt, eds, *Advances in Modal Logic*, pages 315–339. College Publications, 2008.
15. V. Sofronie-Stokkermans, C. Ihlemann. Automated reasoning in some local extensions of ordered structures. In *Proc. ISMVL 2007*, Paper 1. IEEE Comp. Soc., 2007.
16. K.A. Spackman, K.E. Campbell, R.A. Côté. Snomed rt: A reference terminology for health care. In *J. of the American Medical Informatics Association*, pages 640–644, 1997.

Status $\mathcal{QIO}$: An Update

Birte Glimm[1], Yevgeny Kazakov[1], and Carsten Lutz[2]

[1] The University of Oxford, Department of Computer Science, UK
[2] Universität Bremen, Germany

Abstract. We prove co-N2ExpTime-hardness for conjunctive query entailment in the description logic $\mathcal{ALCOIF}$, thus improving the previously known 2ExpTime lower bound. The result transfers to OWL DL and OWL2 DL, of which $\mathcal{ALCOIF}$ is an important fragment. A matching upper bound remains open.

1 Introduction

Due to its importance for ontology-based data access and data integration, conjunctive query (CQ) answering has developed into one of the most widely studied reasoning tasks in description logic (DL). Nevertheless, the precise complexity (and sometimes even decidability) of CQ answering in several important expressive DLs is still an open problem. In particular, this concerns fragments of the W3C-standardized OWL DL ontology language that comprise nominals, inverse roles, and number restrictions, a combination of expressive means that is notorious for interacting in intricate ways. In this paper, we concentrate on the basic such fragment $\mathcal{ALCOIF}$ in which number restrictions take the form of global functionality constraints.

Decidability of CQ answering in $\mathcal{ALCOIF}$ and its extension $\mathcal{ALCOIQ}$ with qualified number restrictions has been shown only very recently [1]. Since the proof is based on a mutual enumeration of finite models and theorems of first-order logic, it does not yield any upper complexity bound. The best known lower bound for CQ answering in $\mathcal{ALCOIF}$ is 2ExpTime, inherited from the fragment $\mathcal{ALCI}$ of $\mathcal{ALCOIF}$ that does not include nominals and functionality constraints [2, 3]. The aim of this paper is to improve upon this lower bound by establishing co-N2ExpTime-hardness. Note that CQ answering in the fragment $\mathcal{ALCIF}$ of $\mathcal{ALCOIF}$ that does not include nominals is in 2ExpTime [4], and the same is true for the fragment $\mathcal{ALCQO}$ that does not include inverse roles [5] and $\mathcal{ALCOI}$ that does not include functionality restrictions [6]. Thus, our result shows that the combination of nominals, inverse roles, and number restrictions leads to an increase of complexity of CQ answering from 2ExpTime to (at least) co-N2ExpTime. This parallels the situation for the subsumption problem, which is co-NExpTime-complete for $\mathcal{ALCOIF}$, but ExpTime-complete in any of $\mathcal{ALCIF}$, $\mathcal{ALCQO}$, and $\mathcal{ALCOI}$. Since $\mathcal{ALCOIF}$ is a fragment of OWL DL (in both the OWL1 and the OWL2 version), our co-N2ExpTime lower bound obviously also applies to CQ answering in this language.

We prove our result by a reduction of the tiling problem that requires to tile a torus of size $2^{2^n} \times 2^{2^n}$. Our construction combines elements of two existing hardness proofs, but also requires the development of novel ideas. We follow the general strategy of the proofs that show N2ExpTime-hardness of satisfiability in $\mathcal{SROIQ}$ [7] and in the extension of $\mathcal{SHOIF}$ with role conjunctions [8]. One central part of those proofs is the realization of a counter that counts up to 2^{2^n}. We realize this counter using a (rather subtle!) adaptation of the conjunctive queries that have been developed in [2, 3] to establish 2ExpTime-hardness of CQ-answering in $\mathcal{ALCI}$.

An extended technical report including proofs and further details is available [9].

2 Preliminaries

We assume standard notation for the syntax and semantics of $\mathcal{ALCOIF}$ knowledge bases [10]. The presence of nominals allows for only working with TBoxes, which consist of concept inclusions (CIs) $C \sqsubseteq D$. A *knowledge base (KB)* is then simply a TBox. Let N_V be a countably infinite set of variables. An atom is an expression $C(v)$ or $r(v, v')$, where C is a (potentially compound) $\mathcal{ALCOIF}$-concept, r is an atomic role, and $v, v' \in N_V$.[3] A conjunctive query q is a finite set of atoms. We use $\mathsf{Var}(q)$ to denote the set of variables that occur in the query q. Let $\mathcal{K}$ be an $\mathcal{ALCOIF}$ KB, $\mathcal{I} = (\cdot^\mathcal{I}, \Delta^\mathcal{I})$ a model of $\mathcal{K}$, q a conjunctive query, and $\pi\colon \mathsf{Var}(q) \to \Delta^\mathcal{I}$ a total function. We write $\mathcal{I} \models^\pi C(v)$ if $\pi(v) \in C^\mathcal{I}$ and $\mathcal{I} \models^\pi r(v, v')$ if $\langle \pi(v), \pi(v') \rangle \in r^\mathcal{I}$. If $\mathcal{I} \models^\pi at$ for all $at \in q$, we write $\mathcal{I} \models^\pi q$ and call π a *match* for $\mathcal{I}$ and q. We say that $\mathcal{I}$ *satisfies* q and write $\mathcal{I} \models q$ if there is a match π for $\mathcal{I}$ and q. If $\mathcal{I} \models q$ for all models $\mathcal{I}$ of a KB $\mathcal{K}$, we write $\mathcal{K} \models q$ and say that $\mathcal{K}$ *entails* q. The *conjunctive query entailment problem* is, given a knowledge base $\mathcal{K}$ and a query q, to decide whether $\mathcal{K} \models q$. This is the decision problem corresponding to query answering, see e.g. [11].

A *domino system* is a triple $D = (T, H, V)$, where $T = \{1, \ldots, k\}$ is a finite set of *tiles* and $H, V \subseteq T \times T$ are *horizontal* and *vertical matching relations*. A *tiling* of $m \times m$ for a domino system D with *initial condition* $c^0 = \langle t_1^0, \ldots, t_n^0 \rangle$, $t_i^0 \in T$ for $1 \leq i \leq n$, is a mapping $t\colon \{0, \ldots, m-1\} \times \{0, \ldots, m-1\} \to T$ such that $\langle t(i, j), t(i+1 \bmod m, j) \rangle \in H$, $\langle t(i, j), t(i, j+1 \bmod m) \rangle \in V$, and $t(i, 0) = t_{i-1}^0$ $(0 \leq i, j < m)$. There exists a domino system D_0 for which it is N2ExpTime-complete to decide, given an initial condition c^0 of length n, whether D_0 admits a tiling of $2^{2^n} \times 2^{2^n}$ with initial condition c^0 [12].

3 Conjunctive Query Entailment in $\mathcal{ALCOIF}$

Our aim is to construct, for an initial condition c^0 of length n, an $\mathcal{ALCOIF}$-KB $\mathcal{K}_0$ and conjunctive query q_0 such that $\mathcal{K}_0 \not\models q_0$ iff D_0 admits a tiling of $2^{2^n} \times 2^{2^n}$ with initial condition c^0.

Intuitively, the models of $\mathcal{K}_0$ that we are interested in have the form depicted in Figure 1: a torus of dimension $2^{2^n} \times 2^{2^n}$, where the lower left corner is identified by the nominal o, the upper right corner by the nominal e, each horizontal dashed arrow denotes the role h, and each vertical dotted arrow the role v. We will install two counters that identify the vertical and horizontal position of torus nodes. To store the counter values, we use binary trees of (roughly) depth n below the torus nodes, where each

[3] Complex concepts C in atoms $C(x)$ are used w.l.o.g.; to eliminate them, we can replace $C(x)$ with $A_C(x)$ for a fresh atomic concept A_C and add $C \sqsubseteq A_C$ to the TBox.

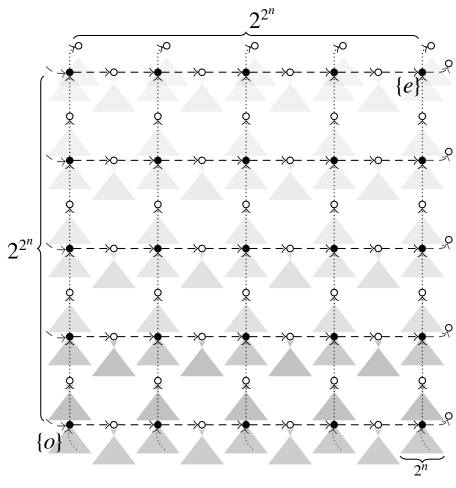


Fig. 1. Schematic depiction of the torus

of the 2^n leaves store one bit of each counter (represented via concept names X and Y). The filled circles in Figure 1 denote true torus nodes, which are labeled by a tile later on, while the unfilled circles denote auxiliary nodes that will help us in properly incrementing the counters. This incrementation is the main difficulty of the reduction, and it is achieved with the help of the query q_0. As the details are intricate, we defer a discussion of the details until later, and first concentrate on the construction of $\mathcal{K}_0$.

The following concept inclusions (1) to (9) of $\mathcal{K}_0$ lay the foundation for enforcing the torus structure with attached trees. Successors in trees are connected via the composition of the roles r^- and r, from now on denoted by $r^-; r$. This is needed in the query construction later on, similar to the use of symmetric roles in [2, 3]. We call additional nodes between r^- and r the 'intermediate' tree nodes. Note that no branching occurs at intermediate nodes. Also for the query construction, the root of a tree below a true torus node is the torus node itself while the root of a tree below an auxiliary torus node is reachable by traveling one step along the role r (see Figure 1). To distinguish these two kinds of trees, we label trees of the former kind with the concept name B and call them black trees, and trees of the latter kind with the concept name W and call them white trees. Later on, we will use white trees that are on the vertical axis to increment the vertical counter and white trees that are on the horizontal axis to increment the horizontal counter. To support this, we further label white trees of the former kind with V and white trees of the latter kind with H. The basic idea for constructing the torus itself is similar to what is done in [13, 7, 8]: the maximum value of both counters (indicated by the concept names M_X and M_Y) identifies the upper right corner, which has to satisfy the nominal e and is thus unique. Inverse functionality for h and v then guarantees uniqueness of elements for all other values of the horizontal and vertical counters, and that the torus 'closes' in the expected way. We use concept names $L_0, \ldots, L_n$ to mark the levels of the trees, to deal with the symmetry of the composition $r^-; r$. Thus, the

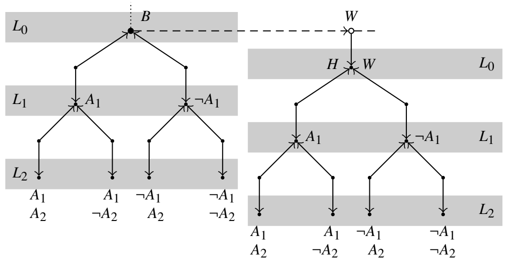


Fig. 2. A black and a white tree, for $n = 2$

concept $B \sqcap L_0$ identifies the true torus nodes.

$$\{o\} \sqsubseteq B \sqcap L_0 \tag{1}$$

$$B \sqcap L_0 \sqsubseteq \exists h.(W \sqcap \exists r.(H \sqcap W \sqcap L_0) \sqcap \exists h.(B \sqcap L_0)) \tag{2}$$

$$B \sqcap L_0 \sqsubseteq \exists v.(W \sqcap \exists r.(V \sqcap W \sqcap L_0) \sqcap \exists v.(B \sqcap L_0)) \tag{3}$$

$$B \sqcap L_0 \sqcap M_X \sqcap M_Y \sqsubseteq \{e\} \tag{4}$$

$$L_i \sqsubseteq \exists r^-.\exists r.(A_{i+1} \sqcap L_{i+1}) \sqcap \exists r^-.\exists r.(\neg A_{i+1} \sqcap L_{i+1}) \quad {\scriptstyle i<n} \tag{5}$$

$$A_i \sqcap L_j \sqsubseteq \forall r^-.\forall r.(L_{j+1} \rightarrow A_i) \qquad {\scriptstyle 1 \le i \le j < n} \tag{6}$$

$$\neg A_i \sqcap L_j \sqsubseteq \forall r^-.\forall r.(L_{j+1} \rightarrow \neg A_i) \qquad {\scriptstyle 1 \le i \le j < n} \tag{7}$$

$$\mathsf{C} \sqsubseteq \forall r.\mathsf{C} \sqcap \forall r^-.\mathsf{C} \qquad \text{for all } \mathsf{C} \in \{B, W, H, V\} \tag{8}$$

$$\top \sqsubseteq \leqslant 1 h^-.\top \qquad \top \sqsubseteq \leqslant 1 v^-.\top \tag{9}$$

Note that the concept names $A_1, \ldots, A_n$ implement a binary counter for the leafs of the trees, i.e., for counting the bit positions in the horizontal and vertical counters. In summary, the internal structure of the trees is as shown in Figure 2 where branching tree nodes have dark background and intermediate nodes have light background.

The next step is to make sure that the horizontal and vertical counter have value 0 at the origin and that M_X is true at the root of a tree when the horizontal counter has reached the maximum value, and similarly for M_Y. We use $\forall(r^-; r)^n.C$ to denote the $2n$-quantifier prefixed $\forall r^-.\forall r. \cdots \forall r^-.\forall r.C$. Recall that the concept name X represents the truth value of bits of the horizontal counter, and likewise for Y and the vertical counter.

$$\{o\} \sqsubseteq \forall(r^-; r)^n.(\neg X \sqcap \neg Y) \tag{10}$$

$$L_n \sqsubseteq (X \leftrightarrow M_X) \sqcap (Y \leftrightarrow M_Y) \tag{11}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap A_i \sqcap M_X) \sqcap \exists r^-.\exists r.(L_i \sqcap \neg A_i \sqcap M_X) \sqsubseteq M_X \quad {\scriptstyle 0<i\le n} \tag{12}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap A_i \sqcap M_Y) \sqcap \exists r^-.\exists r.(L_i \sqcap \neg A_i \sqcap M_Y) \sqsubseteq M_Y \quad {\scriptstyle 0<i\le n} \tag{13}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap \neg M_X) \sqsubseteq \neg M_X \quad {\scriptstyle 0<i\le n} \tag{14}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap \neg M_Y) \sqsubseteq \neg M_Y \quad {\scriptstyle 0<i\le n} \tag{15}$$

The general strategy for updating the horizontal and vertical counter is as follows. We introduce additional concept names X' and Y', which represent the truth value of the bits of two additional binary counters, the 'primed versions' of the horizontal and vertical counter. Using $\mathcal{K}_0$, we ensure that, in black trees, the X-counter has the same value as the X'-counter, and likewise for the Y- and Y'-counter. In white trees, we distinguish between horizontal incrementation indicated by the concept name H and vertical incrementation indicated by the concept name V: if the tree satisfies H, then the value of the X'-counter is the value of the X-counter incremented by one, while the values of the Y- and Y'-counter coincide; if the tree satisfies V, it is the other way around. The remaining job to be accomplished by the query q_0 is then to

($*$) ensure that the value of the X'-counter (resp. Y'-counter) in a (black or white) tree is identical to the value of the X-counter (resp. Y-counter) in its 'successor trees', i.e., in trees that can be reached by traveling a single step in the torus along the roles h or v.

This behavior of the counters, with the exception of ($*$), is implemented by the subsequent concept inclusions. To increment a counter, we use a concept name F to mark the bits that have to be flipped. Another concept name S, which is propagated down from the root to a single leaf, is used to mark the unique bit of the incremented counter such that (i) all bits to the right are flipped from 1 to 0, (ii) the bit itself is flipped from 0 to 1, and (iii) all bits to the left remain unchanged. As a special case, all bits flip when the maximum counter value has been reached. In the following, CIs (16) to (20) implement the proper marking by F and S, CIs (21) to (23) realize the actual incrementation of the X-counter to the X'-counter in (white) H-trees, and CI (24) ensures that the Y- and Y'-counters have the same value in H-trees and in black trees. We also need CIs (21) to (24) with H replaced by V, X by Y, X' by Y', Y by X, and Y' by X'.

$$L_0 \sqcap (\neg M_X \sqcup \neg M_Y) \sqsubseteq S \tag{16}$$

$$L_0 \sqcap (M_X \sqcap M_Y) \sqsubseteq F \sqcap \neg S \tag{17}$$

$$\begin{aligned} L_{i-1} \sqcap S \sqsubseteq{} & \forall r^-.\forall r.[L_i \rightarrow (A_i \sqcap \neg F \sqcap \neg S) \sqcup (\neg A_i \sqcap S)] \sqcup \\ & \forall r^-.\forall r.[L_i \rightarrow (A_i \sqcap S) \sqcup (\neg A_i \sqcap F \sqcap \neg S)] \qquad {\scriptstyle 0<i\leq n} \end{aligned} \tag{18}$$

$$L_{i-1} \sqcap F \sqcap \neg S \sqsubseteq \forall r^-.\forall r.(L_i \rightarrow F \sqcap \neg S) \qquad {\scriptstyle 0<i\leq n} \tag{19}$$

$$L_{i-1} \sqcap \neg F \sqcap \neg S \sqsubseteq \forall r^-.\forall r.(L_i \rightarrow \neg F \sqcap \neg S) \qquad {\scriptstyle 0<i\leq n} \tag{20}$$

$$H \sqcap L_n \sqcap F \sqcap \neg S \sqsubseteq X \sqcap \neg X' \tag{21}$$

$$H \sqcap L_n \sqcap S \sqsubseteq \neg X \sqcap X' \tag{22}$$

$$H \sqcap L_n \sqcap \neg F \sqcap \neg S \sqsubseteq (X \sqcap X') \sqcup (\neg X \sqcap \neg X') \tag{23}$$

$$(B \sqcup H) \sqcap L_n \sqsubseteq (Y \sqcap Y') \sqcup (\neg Y \sqcap \neg Y') \tag{24}$$

To enable the construction of a query q_0 that enforces ($*$), we add a further (single) $r^-; r$-successor to each leaf in each tree. At this extra node, which is marked with the concept name L_{n+1}, the truth value of all concept names A_i, X, X', Y, Y' is complemented compared to its predecessor L_n-node. We also introduce a marker concept Q that is true at the intermediate node between each L_n-node and L_{n+1}-node. This is similar to what is done in [2, 3]. We call such intermediate nodes Q-nodes.

$$L_n \sqsubseteq \exists r^-.(Q \sqcap \exists r.L_{n+1}) \tag{25}$$

$$\begin{gathered} L_n \sqcap \mathsf{C} \sqsubseteq \forall r^-.\forall r.(L_{n+1} \rightarrow \neg \mathsf{C}) \qquad L_n \sqcap \neg \mathsf{C} \sqsubseteq \forall r^-.\forall r.(L_{n+1} \rightarrow \mathsf{C}) \\ \text{for all } \mathsf{C} \in \{A_1, \ldots, A_n, X, X', Y, Y'\} \end{gathered} \tag{26}$$

The construction of $\mathcal{K}_0$ is not yet finished. However, it will be more convenient to construct the remaining part along with the query q_0. The query is assembled from

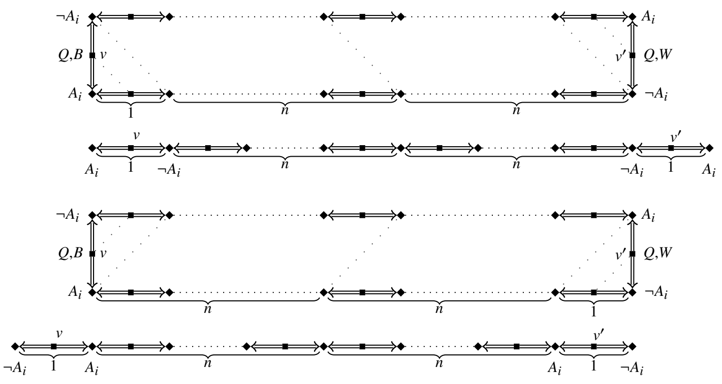


Fig. 3. A counting component of the query q_0 and the two ways to fold it

two types of components: *counting components* and *copying components*. We start with presenting and explaining a simplified version of counting components, which are then refined in a second step. The final query q_0 will contain one counting component for each bit of the counter $A_1, \ldots, A_n$ that counts the leaves of our trees. The simplified version of the counting component for A_i is shown as the topmost cycle in Figure 3, where, for the moment, every arrow should be interpreted as a role atom that uses the role name r. The goal is that matches of this query component should map (i) v to a Q-node of a black tree and v' to the Q-node of a white successor tree such that the two predecessor L_n-nodes agree on the value of A_i or, symmetrically, (ii) v' to a Q-node of a white tree and v to the Q-node of a black successor tree such that the two predecessor L_n-nodes agree on the value of A_i. By taking the union of all counting queries for $A_1, \ldots, A_n$ such that the variables v and v' are shared, we thus link leaves of successor trees that represent the same bit position for the horizontal and vertical counter, which is the first important step towards enforcing $(*)$.

Due to the Q-concept at v and v', each variable labeled with A_i or $\neg A_i$ is matched to an L_n-node or an L_{n+1}-node. Ignoring the presence of the role names h and v in the torus and pretending that white trees are rooted directly on the torus, each match of the counting component gives rise to one of the two 'foldings' presented in Figure 3. These foldings are obtained by identifying variables that are matched to the same domain element, as indicated by the dotted lines. Intuitively, the two foldings correspond to the bit A_i being false (upper folding) and true (lower folding). For brevity, we omit the concept names Q, B, W in the foldings. Since the long sides of the counting component are of length $2n + 1$ (counted in terms of compositions $r^-; r$) and trees are of depth n, the two trees involved in a match cannot be further away than one step in the torus. Due to the use of B and W, they cannot be identical.

In the discussion of the simplified counting components above, we have neglected the presence of the roles h and v in the torus that we need to 'cross' when matching the query in the described way. Refining the counting queries to deal with these roles is the major challenge in the current reduction, compared to the 2ExpTime lower bound in [2, 3] where only a single role r is used. Note that we cannot just introduce a single

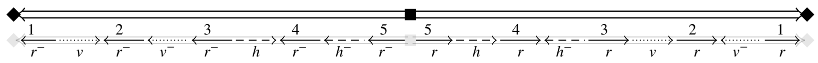


Fig. 4. Internal structure of a meta role consisting of 18 roles

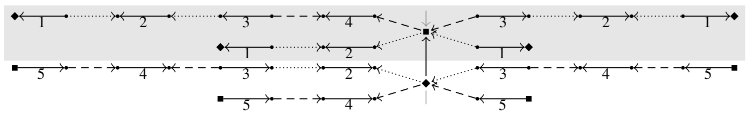


Fig. 5. Side chains for branching (upper) and intermediate (lower) nodes

h-arrow and v-arrow into the counting components since we want to match either h or v, but not both; moreover, the position of the h-arrow/v-arrow would shift back and forth with the different ways to fold the query. To solve the problem, we replace each role composition $r^-; r$ in the query (but not in the trees!) with a composition of 18 roles that we call a 'meta role', see Figure 4.

Note that the meta role is symmetric, like the composition $r^-; r$. The aim is that each meta-role in the refined counting query matches one $r^-; r$-role composition that connects two successor nodes in a tree. To resolve the mismatch between the role composition $r^-; r$ of length two and the meta role of length 18, the meta role is designed such that the remaining parts can be folded away into 'side chains' that we will add to each tree, i.e., chains of roles that start at each tree node. There are five ways to achieve such a folding, one for each corresponding pair of r^-- and r-arrows in the left and right half of the meta role. For example, we can use the 3rd r^--arrow and the 3rd r-arrow to match the $r^-; r$-composition in the tree, and then have to fold away the prefix composition $r^-; v; r^-; v^-$ before the 3rd r^--arrow, the infix composition $h; r^-; h^-; r^-; r; h; r; h^-$ between the 3rd r^--arrow and the 3rd r-arrow, and the postfix composition $v; r; v^-; r$ following the 3rd r-arrow. Observe that the infix composition is symmetric and thus can be folded into a chain. The postfix composition is the converse of the infix composition, which will allow us to leave a side chain that we have entered with the postfix composition using the prefix composition of the subsequent meta role. Similar foldings allow us to match the $r^-; r; h; r$-compositions required to move up one level in a black tree and then cross via an h-edge to the root of a white successor tree, the $r^-; h; r^-; r$-compositions that allows us to cross from the root of a white tree to a black tree and then move down one level, and to perform the two remaining crossing with h replaced by v.

The scheme for adding side chains is shown in Figure 5, where intermediate tree nodes (lower node on the center line) receive different chains than branching tree nodes (upper node on the center line). These chains are added to every node in each tree with the exception of the roots of black trees, as those are directly on the torus and adding side chains would violate inverse functionality of h and v. Note that the side chains attached to branching tree nodes are precisely the possible postfix compositions mentioned above, while the side chains attached to intermediate tree nodes are foldings of what we called infix compositions above. The chains are generated by the following CIs, to be added to $\mathcal{K}_0$. We use the concept $N_B = (L_0 \sqcap W) \sqcup \bigsqcup_{1 \leq i \leq n+1} L_i$ to identify

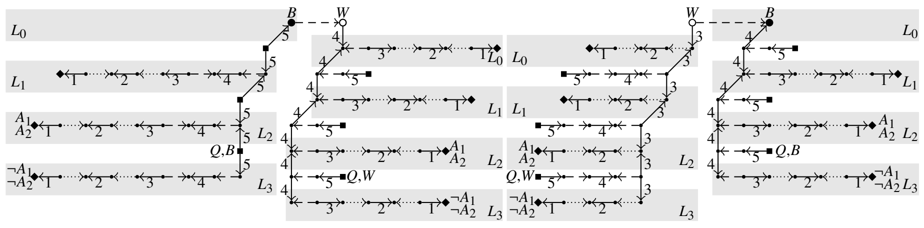


Fig. 6. From black to white trees via h (left) and from white to black trees via h (right)

branching tree nodes and $N_I = \exists r. \bigsqcup_{1 \leq i \leq n+1} L_i$ to identify intermediate tree nodes.

$$\begin{aligned} N_B \sqsubseteq\ & \exists h.\exists r.\exists h^-.\exists r.\exists v.\exists r.\exists v^-.\exists r.\top \sqcap \exists v.\exists r.\exists v^-.\exists r.\top \sqcap \\ & \exists h^-.\exists r.\exists v.\exists r.\exists v^-.\exists r.\top \sqcap \exists v^-.\exists r.\top \end{aligned} \tag{27}$$

$$\begin{aligned} N_I \sqsubseteq\ & \exists v.\exists r^-.\exists v^-.\exists r^-.\exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \\ & \exists v^-.\exists r^-.\exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \exists h^-.\exists r^-.\top \end{aligned} \tag{28}$$

In matches of the refined query, the endpoints of the two foldings shown in Figure 3 will match at the end of (possibly empty) side chains at the level $n+1$, v and v' will match at the end of the side chains between the levels n and $n+1$, and the adjacent inner nodes labeled with $\neg A_i$ resp. A_i will match at the end of the side chains at the level n. For this reason, we propagate all relevant concept names to the end of those chains. For $\mathsf{C} \in \{A_1, \ldots, A_n, \neg A_1, \ldots, \neg A_n, X, \neg X, Y, \neg Y\}$, $\mathsf{C}' \in \{Q, B, W\}$, add the concept inclusions

$$\begin{aligned} (L_n \sqcup L_{n+1}) \sqcap \mathsf{C} \sqsubseteq\ & \forall h.\forall r.\forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \\ & \forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \forall v^-.\forall r.\mathsf{C} \end{aligned} \tag{29}$$

$$\begin{aligned} Q \sqcap \mathsf{C}' \sqsubseteq\ & \forall v.\forall r^-.\forall v^-.\forall r^-.\forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \\ & \forall v^-.\forall r^-.\forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \forall h^-.\forall r^-.\mathsf{C}' \end{aligned} \tag{30}$$

Figure 6 shows, for $n = 2$, how to fold the refined counting query such that v is mapped to a Q-node of a black tree and v' to a Q-node of a white successor tree that can be reached via crossing an h-edge in the torus, and likewise for the case where v' is mapped to a white tree, and v to a black successor tree reachable via h. We display only those side chains that are needed for accommodating the query match. To get started, note that in the left part of Figure 6, the h-edge in the right half of a meta role as shown in Figure 4 is matched onto the crossing h-edge in the model. The square and diamond nodes indicate where the middle and end parts of each meta role in the query match. Crossings of v-edges are similar.

We now define counting query parts in a more precise way. Note that each counting query consists of $4n+4$ meta roles. In the subsequent definition, $q_m^{i,j}$ is a meta role used in the counting query for A_i, where j ranges over $0, \ldots, 4n+3$.

Definition 1. *For all i, j with* $1 \leq i \leq n$ *and* $0 \leq j < 4n + 4$, *put*

$$\begin{aligned} q_m^{i,j} := \{\, & r(v_1^{i,j}, v_0^{i,j}), v(v_1^{i,j}, v_2^{i,j}), r(v_3^{i,j}, v_2^{i,j}), v(v_4^{i,j}, v_3^{i,j}), r(v_5^{i,j}, v_4^{i,j}), h(v_5^{i,j}, v_6^{i,j}), \\ & r(v_7^{i,j}, v_6^{i,j}), h(v_8^{i,j}, v_7^{i,j}), r(v_9^{i,j}, v_8^{i,j}), r(v_9^{i,j}, v_{10}^{i,j}), h(v_{10}^{i,j}, v_{11}^{i,j}), r(v_{11}^{i,j}, v_{12}^{i,j}), \\ & h(v_{12}^{i,j}, v_{13}^{i,j}), r(v_{13}^{i,j}, v_{14}^{i,j}), v(v_{14}^{i,j}, v_{15}^{i,j}), r(v_{15}^{i,j}, v_{16}^{i,j}), v(v_{17}^{i,j}, v_{16}^{i,j}), r(v_{17}^{i,j}, v_0^{i,j+1})\} \end{aligned}$$

with $v_0^{i,4n+4} = v_0^{i,0}$. *For each i with* $1 \leq i \leq n$, *the* counting query for A_i *is*

$$q_c^i := \{\, A_i(v_0^{i,0}), \neg A_i(v_0^{i,1}), A_i(v_0^{i,2n+2}), \neg A_i(v_0^{i,2n+3})\} \ \cup \bigcup_{0 \leq j < 4n+4} q_m^{i,j}$$

with $v_9^{i,0} = v, v_9^{i,2n+2} = v'$. *The counting query* q_c *for the whole counter is*

$$q_c := \{B(v), Q(v), W(v'), Q(v')\} \cup \bigcup_{1 \leq i \leq n} q_c^i$$

Note that each counting query q_c^i is a cycle as intended since $v_0^{i,4n+4} = v_0^{i,0}$.

As explained above, the overall counting query q_c links a Q-node x_1 of a tree to the Q-nodes x_2 of its successor trees that represent the same bit position for the horizontal and vertical counters. To establish the central property $(*)$ in models of $\mathcal{K}_0$ that do *not* match the query q_0 to be constructed, it thus remains to modify q_0 such that it matches only if the truth assignment at the L_n-predecessor x_1' of x_1 to X' and Y' is *not* identical to the truth assignment at the L_n-predecessor x_2' of x_2 to X and Y. This is achieved by the second type of component queries in q_0, the copying components.

To prepare for these components, let us distinguish two types of side chains in the tree nodes: the *outgoing chains* starting with h and v shown to the left in Figure 5 and the *incoming chains* starting with the inverses of h and v show to the right in Figure 5. As can be seen in Figure 6, when the query has a match, the incoming chains are used in a predecessor tree and the outgoing chains are used in a successor tree. We will distinguish the ends of incoming chains from the ends of the outgoing chains on the levels n and $n + 1$ using an additional concept P:

$$\begin{aligned} (L_n \sqcup L_{n+1}) \sqsubseteq {} & \forall h.\forall r.\forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.P \sqcap \forall v.\forall r.\forall v^-.\forall r.P \sqcap \\ & \forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.\neg P \sqcap \forall v^-.\forall r.\neg P \end{aligned} \tag{31}$$

The copying components take the form displayed in Figure 7, i.e., there are 8 such components in total. Each component is like the upper half of a counting component, except that the concept labels have changed to negated conjunctions. In Figure 7, the four copying components in each row take care of each possible truth assignment to X' and Y' for the predecessor and the corresponding assignment to X and Y for the successor. We need two queries per truth assignment to deal with the two possible ways in which a counting query can match for the variables v and v':

(a) v matches into a tree that satisfies B, and v' into a successor tree that satisfies W;
(b) v' matches into a tree that satisfies W, and v into a successor tree that satisfies B;

To explain in detail how the copying queries work, consider case (a). Due to the Q-label in the *counting* queries, the variables v and v' can only be matched to Q-nodes. Thus assume that v is matched to a Q-node x_1 of a predecessor tree that satisfies B and v' to a Q-node x_2 of a successor tree that satisfies W. The relevant queries are those

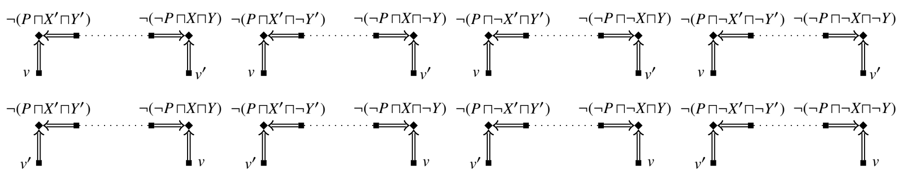


Fig. 7. The 8 query copying components

from the first row in Figure 7. Let u be the neighboring variable of v in the copying components, and u' the neighboring variable of v'; thus, both u and u' are labeled with negated conjunctions. Similar to the situation in Figure 6, u can only be matched to the ends of two outgoing chains (satisfying P): one chain is at level n and another one is at level $n+1$. Let us refer to the ends of these chains as x_1' and x_1'' respectively. Likewise, u' can only be matched to one of the two ends x_2' and x_2'' of incoming chains (satisfying $\neg P$) at the levels n and $n+1$, respectively.

First assume that the truth assignment at x_1' to X' and Y' is identical to the truth assignment at x_2' to X and Y and thus there should be no match of the overall query q_0. Take the corresponding counting component from the first row, e.g., the first one when X', Y', X, and Y are all interpreted as true. It can be seen that, in this situation, the matches with $u \mapsto x_1'$ or with $u' \mapsto x_2'$ are not possible because they violate the concept labels of u and respectively u'. The match $u \mapsto x_1''$ and $u' \mapsto x_2''$ is also not possible because the path from u to u' is not long enough. Thus, there is no match of this component, whence no match of the overall query q_0.

Conversely, assume that the truth assignment at x_1' to X' and Y' is different from the truth assignment at x_2' to X and Y, e.g., that x_1' satisfies X' but x_2' does not satisfy X. Since by (26) the truth values of X' and X are complemented at the level $n+1$, x_1'' does not satisfy X' and x_2'' satisfies X. Then the first two components from the first row have a match $u \mapsto x_1''$ and $u' \mapsto x_2'$ and the next two components have a match $u \mapsto x_1'$ and $u' \mapsto x_2''$. All components in the second row have a match due to the use of the concept name P in the labels (note the swapped v, v'). Thus,the overall query q_0 matches.

A formal definition of copying queries can be found in [9].

This finishes the construction of the query q_0 and of the part of $\mathcal{K}_0$ that enforces the torus structure. It remains to encode tilings of the domino system D_0:

$$\top \sqsubseteq T_1 \sqcup \cdots \sqcup T_k \qquad T_i \sqcap T_j \sqsubseteq \bot \qquad 1 \le i < j \le k \tag{32}$$

$$T_i \sqcap \exists h.T_j \sqsubseteq \bot \qquad T_k \sqcap \exists v.T_\ell \sqsubseteq \bot \qquad \langle i,j\rangle \notin H, \langle k,\ell\rangle \notin V \tag{33}$$

Finally, we enforce the initial condition $c^0 = \langle t_1^0, \ldots, t_n^0\rangle$ of the torus.

$$\{o\} \sqsubseteq T_{t_1^0} \sqcap \forall h.(T_{t_2^0} \sqcap \forall h.(T_{t_3^0} \sqcap \forall h.(T_{t_4^0} \sqcap \ldots \forall h.T_{t_n^0} \ldots))) \tag{34}$$

More details regarding the correctness of the reduction can be found in [9]. The most challenging issue is to show that when D_0 admits a tiling with initial condition c^0 and we build a model $\mathcal{I}$ of $\mathcal{K}$ that has the intended torus shape, then $\mathcal{I} \not\models q_0$: we need to prove that there are no unintended foldings and matchings of the query q_0.

Theorem 1. *Conjunctive query entailment by $\mathcal{ALCOIF}$ knowledge bases is* co-N2EXPTIME-*hard.*

4 Conclusions

We have shown that conjunctive query entailment in the Description Logic $\mathcal{ALCOIF}$ is hard for co-N2ExpTime. The challenging problem of finding a matching upper bound, or in fact *any* elementary upper bound, remains open.

Acknowledgments B. Glimm is supported by EPSRC grant EP/F065841/1, Y. Kazakov by EPSRC grant EP/G02085X, and C. Lutz by DFG SFB/TR 8 “Spatial Cognition”.

References

1. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries. J. of Artificial Intelligence Research **39** (2010) 429–481
2. Lutz, C.: Inverse roles make conjunctive queries hard. In: Proc. of the 2007 Description Logic Workshop (DL 2007). (2007)
3. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-08), LNCS (2008) 179–193
4. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the Seventeenth ACM SIGACT SIGMOD Sym. on Principles of Database Systems (PODS-98), ACM Press and Addison Wesley (1998) 149–158
5. Glimm, B., Horrocks, I., Sattler, U.: Unions of conjunctive queries in $\mathcal{SHOQ}$. In: Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08), AAAI Press/The MIT Press (2008)
6. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of the 21st Int. Joint Conf. on AI (IJCAI-09). (2009) 714–720
7. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08), AAAI Press/The MIT Press (2008)
8. Glimm, B., Kazakov, Y.: Role conjunctions in expressive description logics. In: Proc. of the 15th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2008). Volume 5330 of LNCS., Springer (2008) 391–405
9. Glimm, B., Kazakov, Y., Lutz, C.: Status $\mathcal{QIO}$: An update. Technical report, The University of Oxford (2011) `http://www.comlab.ox.ac.uk/files/3969/GlKL11a.pdf`.
10. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook. Cambridge University Press (2003)
11. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. J. of Artificial Intelligence Research **31** (2008) 151–198
12. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic. Springer (1997) Second printing, Springer Verlag, 2001.
13. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. of Artificial Intelligence Research **12** (2000)

Analysing Multiple Versions of an Ontology: A Study of the NCI Thesaurus

Rafael S. Gonçalves, Bijan Parsia, and Uli Sattler

School of Computer Science, University of Manchester, UK
`{goncalvj,bparsia,sattler}@cs.man.ac.uk`

Abstract. The detection of changes between OWL ontologies is an important service for ontology engineering. There are several approaches to this problem, both syntactic and semantic. A purely syntactic analysis of changes is insufficient to detect changes with logical effect, while the current state of the art in semantic diffing ignores logically ineffectual changes, which might be of great interest to the user. We develop an exhaustive categorisation of ineffectual changes, based on their justifications. In order to verify the applicability of our approach, we collected 88 OWL versions of the National Cancer Institute (NCI) Thesaurus (NCIt), and extracted all pairwise, consecutive diffs. We discovered a substantial number of ineffectual changes and, as a result, argue that the devised categorisation of changes is beneficial for ontology engineers. We devised and applied a method for performance impact analysis (*culprit finding*) based on the diff between ontologies, and identified a number of culprits between two NCIt versions.

1 Motivation

The comparison of ontologies is a valuable service whether for purely analytic purposes, versioning systems [3], or collaboration. When comparing two ontologies it is desirable to detect both syntactic and logical changes. OWL defines a high level notion of syntactic equivalence, so-called "structural equivalence", which abstracts from such concrete details as the order of axioms. Associated with structural equivalence is *structural difference*. A different syntactic approach is that of an edit-based diff, wherein change records are produced within the ontology editor being used thereby capturing the history of change, as implemented in Swoop [8]. The diffs mentioned so far, as well as PROMPTDIFF [12], do not recognize the logical impact of changes. When analysing the impact of changes, it is sensible to inspect not only logically effectual changes, but also ineffectual ones since these might have been intended to have logical impact, and thus may be of interest to users. Semantic diffs, such as CEX [10, 4], OWLDiff [11] or ContentCVS [7] detect only effectual changes. So on the one hand, syntactic diffs detect without distinction both effectual and ineffectual changes, and on the other hand semantic diffs do not analyse ineffectual changes.

In this paper we propose a diff notion that builds on structural diff with a logical impact analysis, which we refer to as *intentional difference*, incorporating a categorisation of ineffectual axioms based on their justifications. The goal of this categorisation is to suggest on the intent behind such changes. For the purpose of verifying the suitability of our approach, we collected all 88 versions of the National Cancer Institute (NCI)

Thesaurus (NCIt) available in OWL format, freely downloadable[1] from the web, and conducted a diachronic study of the corpus. This study consisted of the extraction of all pairwise, consecutive diffs between NCIt versions. Our diff revealed a fairly high number of ineffectual changes across the corpus, averaging at 13% and even reaching values above 90%. In addition to this we carried out a reasoner performance test to inspect the performance impact of both effectual and ineffectual changes throughout the NCIt. While ineffectual changes carry no logical impact, it is still the case that they have a performance impact.[2] The test revealed an unusual performance increase between 2 versions, the latter of which was 89% faster and also slightly bigger in number of axioms. This motivated a more in-depth performance impact analysis, wherein we attempt to find subsets of the slow ontology without which the ontology performs considerably faster (referred to as *culprits*). We devise a culprit finding method based on the diff between ontologies, and demonstrate its applicability with a number of culprits for the NCIt case.

2 Preliminaries

We assume the reader to be reasonably familiar with OWL [13], as well as the underlying description logics (DLs) [5], though detailed knowledge is not required. We do use the notion of entailment [2], which is identical to the standard first order logic entailment (albeit restricted to certain syntactic forms for consequences, typically atomic subsumption). When comparing two versions of an ontology we refer to the earlier version as $\mathcal{O}_1$, and the more recent as $\mathcal{O}_2$. A justification $\mathcal{J}$ of a consequence α is a minimal subset of an ontology $\mathcal{O}$ that is sufficient for α to hold [9]. The signature of an ontology $\mathcal{O}$ is denoted $\widetilde{\mathcal{O}}$. An axiom $\alpha \in \mathcal{O}_1$ is logically ineffectual for an ontology $\mathcal{O}_2$ iff $\alpha \notin \mathcal{O}_2$ and $\mathcal{O}_2 \models \alpha$, and we often describe it as having no impact.

3 Ontology Difference

The problem of computing the difference between pairs of ontologies has been approached both syntactically and semantically. We distinguish two major aspects of ontology diffing: *(i)* the detection of changes, and *(ii)* the presentation of changes to the end-user. As we analyse existing diff approaches, we point out that most effort has been largely dedicated to *(i)*. It is often the case that the output of diff operations is the set of axioms or terms in the diff. While this may reflect the desired identification of change, it does not convey sufficient information to the user w.r.t. the intent of changes, or whether these are effectual or not.

3.1 Diff Desiderata

Table 1 summarises useful features of an ontology diff, and whether existing approaches exhibit such desiderata.

[1] `http://evs.nci.nih.gov/ftp1/NCI_Thesaurus`

[2] A trivial example is adding all inferred subsumptions, therefore speeding up reasoning tasks.

Table 1. Desiderata of ontology diffing approaches.

Properties	**ContentCVS**	**CEX**	**OWLDiff [11]**	**PROMPTDIFF**	**Swoop Diff**
Difference Detection					
Syntactic analysis	✓	✗	✓	✓	✓
Semantic analysis	✓	✓	✓	✗	✗
Effectively computable	✓	✗	✓	✓	✓
OWL 2 adequacy	✓	✗	✓	✗	✓
Difference Output					
Axiom-based	✓	✓	✓	✗	✓
Term-based	✗	✓	✗	✓	✓
Effectual change analysis	✓	✓	✓	N/A	N/A
Ineffectual change analysis	✗	✗	✗	N/A	N/A

Among the stated properties, an ideal logical diff should combine effective computability for OWL 2 ontologies while providing some analysis of the impact of changes, whether these be effectual or ineffectual. Although this is a complex task in itself, from Table 1 we see that some diffs analyse effectual changes, but none of them inspects ineffectual changes. This desideratum leads to the categorisation method proposed in this paper for the latter type of changes.

3.2 Intentional Diff

Given the limitations of diff approaches described in Table 1 w.r.t. (ii) (as described at the beginning of Section 3), we build on the notion of structural difference with a categorisation mechanism for ineffectual axioms. This requires checking if axioms in the first ontology are entailed by the second (and vice-versa), if that is not the case then those axioms are regarded as effectual changes.

Consider the following ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, which are referred to in examples throughout this section:

$$\begin{aligned}
\mathcal{O}_1 = \{\alpha_1 :&\ A \sqsubseteq C, \\
\alpha_2 :&\ B \sqsubseteq C, \\
\alpha_3 :&\ E \equiv D, \\
\alpha_4 :&\ D \sqsubseteq F, \\
\alpha_5 :&\ F \sqsubseteq G, \\
\alpha_6 :&\ G \sqsubseteq H \sqcap \exists s.H, \\
\alpha_7 :&\ F \sqsubseteq I, \\
\alpha_8 :&\ F \sqsubseteq G \sqcap I \sqcap J\}
\end{aligned}
\qquad
\begin{aligned}
\mathcal{O}_2 = \{\beta_1 :&\ A \sqsubseteq B \sqcup C, \\
\beta_2 :&\ A \sqsubseteq B, \\
\beta_3 :&\ B \sqsubseteq C, \\
\beta_4 :&\ E \sqsubseteq D, \\
\beta_5 :&\ D \sqsubseteq E, \\
\beta_6 :&\ E \sqsubseteq B \sqcup \exists r.C, \\
\beta_7 :&\ D \sqsubseteq E \sqcup G, \\
\beta_8 :&\ G \sqsubseteq \exists s.H \sqcap H, \\
\beta_9 :&\ F \sqsubseteq G \sqcap I\}
\end{aligned}$$

The notion of structural difference is based on OWL's notion of structural equivalence (denoted $\equiv_s$) [13]. The latter deems the order of axioms in an ontology as irrele-

[2] For DLs up to $\mathcal{SROIQ}$.

vant, as well as the order of disjunctions or conjunctions between concepts. Therefore one can rule out differences that an otherwise syntactic equality based diff would detect.

Definition 1 (Structural Difference [6]) *The structural difference between $\mathcal{O}_1$ and $\mathcal{O}_2$ are the following sets:*

- $\text{Additions}(\mathcal{O}_1, \mathcal{O}_2) = \{\beta \in \mathcal{O}_2 \mid \textit{there is no } \alpha \in \mathcal{O}_1 \textit{ s.t. } \alpha \equiv_s \beta\}$
- $\text{Removals}(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha \in \mathcal{O}_1 \mid \textit{there is no } \beta \in \mathcal{O}_2 \textit{ s.t. } \alpha \equiv_s \beta\}$

So if there is an axiom β s.t. $\beta \in \text{Additions}$, this implies that $\beta \in \mathcal{O}_2 \setminus \mathcal{O}_1$, and similarly for Removals. Examine the following example:

Example 1 *From the defined ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ we have that:*

- ◇ $\text{Additions}(\mathcal{O}_1, \mathcal{O}_2) = \{\beta_1, \beta_2, \beta_4, \beta_5, \beta_6, \beta_7, \beta_9\}$
- ◇ $\text{Removals}(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha_1, \alpha_3, \alpha_4, \alpha_5, \alpha_7, \alpha_8\}$

Note that the axiom α_2 is syntactically equal to β_3; $\alpha_2 = \beta_3$. We also have that $\alpha_6 \equiv_s \beta_8$. Therefore these axioms are not reported as changes.

Based on these two sets, the logical difference pinpoints which axioms in Additions (or Removals) affect the set of entailments of $\mathcal{O}_1$ (or $\mathcal{O}_2$). In other words, it distinguishes between those axioms in the structural difference which are entailed by $\mathcal{O}_1$ (or $\mathcal{O}_2$), as follows:

Definition 2 (Logical Difference) *The logical difference between $\mathcal{O}_1$ and $\mathcal{O}_2$ are the following sets:*

- $\text{EffectualAdditions}(\mathcal{O}_1, \mathcal{O}_2) = \{\beta \in \text{Additions}(\mathcal{O}_1, \mathcal{O}_2) \mid \mathcal{O}_1 \not\models \beta\}$
- $\text{EffectualRemovals}(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha \in \text{Removals}(\mathcal{O}_1, \mathcal{O}_2) \mid \mathcal{O}_2 \not\models \alpha\}$
- $\text{IneffectualAdditions}(\mathcal{O}_1, \mathcal{O}_2) = \text{Additions} \setminus \text{EffectualAdditions}$
- $\text{IneffectualRemovals}(\mathcal{O}_1, \mathcal{O}_2) = \text{Removals} \setminus \text{EffectualRemovals}$

The resulting sets $\text{IneffectualAdditions}$ and $\text{IneffectualRemovals}$ are composed of those axioms which do not change the set of entailments of $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. An axiom β is in $\text{IneffectualAdditions}$ iff $\mathcal{O}_1 \models \beta$, and similarly for $\text{IneffectualRemovals}$ (Example 2).

Example 2 *Given the sets* Additions *and* Removals *(from Example 1) we have that:*

- ◇ $\text{EffectualAdditions}(\mathcal{O}_1, \mathcal{O}_2) = \{\beta_2, \beta_6\}$
- ◇ $\text{EffectualRemovals}(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha_4, \alpha_8\}$
- ◇ $\text{IneffectualAdditions}(\mathcal{O}_1, \mathcal{O}_2) = \{\beta_1, \beta_4, \beta_5, \beta_7, \beta_9\}$
- ◇ $\text{IneffectualRemovals}(\mathcal{O}_1, \mathcal{O}_2) = \{\alpha_1, \alpha_3, \alpha_5, \alpha_7\}$

In order to characterise ineffectual changes, we devise a categorisation of axioms based on their justifications as follows:

Definition 3 (Intentional difference) *An axiom $\alpha \in$* $\text{IneffectualRemovals}$ *is:*

- $\mathrm{Strengthened}$, *if there is a $\mathcal{J}$ for α with $\mathcal{J} \cap \mathrm{EffectualAdditions} \neq \emptyset$.*
- $\mathrm{Rewritten}$, *if there is a justification $\mathcal{J}$ for α with $\mathcal{J} \cap \mathrm{Additions} \neq \emptyset$, and $\alpha \models \mathcal{J}$. If $\mathcal{J} \subseteq \mathrm{Additions}$ then α is a complete rewrite, otherwise a partial rewrite.*
- $\mathrm{Redundant}$, *if there is a $\mathcal{J}$ for α with $\mathcal{J} \subseteq (\mathcal{O}_1 \cap \mathcal{O}_2)$. If $\mathcal{J} \subseteq (\mathcal{O}_1 \cap \mathcal{O}_2) \cup \mathrm{IneffectualAdditions}$ then α is an avoided redundancy.*

To obtain the corresponding categories for added axioms $\beta \in \mathrm{IneffectualAdditions}$, replace α, $\mathrm{Additions}$, $\mathrm{EffectualAdditions}$ *and* $\mathrm{IneffectualAdditions}$ *with β,* $\mathrm{Removals}$, $\mathrm{EffectualRemovals}$ *and* $\mathrm{IneffectualRemovals}$ *respectively. In* $\mathrm{IneffectualAdditions}$ *the label for the criteria of* $\mathrm{Strengthened}$ *axioms changes to* $\mathrm{Weakened}$ *axioms.*

The intentional difference gives possibly overlapping sets of axioms, as demonstrated in Example 3. Also we note that these categories are exhaustive, in the sense that there is no axiom such that the justifications of which do not imply one of the defined categories. Consider an axiom α and ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, with $\alpha \in \mathcal{O}_1$ but $\alpha \notin \mathcal{O}_2$, and $\mathcal{O}_2 \models \alpha$. Then there must be a justification $\mathcal{J} \subseteq \mathcal{O}_2$ for α. If $\mathcal{J} \subseteq (\mathcal{O}_1 \cap \mathcal{O}_2) \cup \mathrm{IneffectualAdditions}$ then α is redundant, otherwise if $\mathcal{J} \cap \mathrm{EffectualAdditions} \neq \emptyset$, then α is strengthened.

Example 3 *Given the sets* $\mathrm{IneffectualAdditions}$ *and* $\mathrm{IneffectualRemovals}$ *(from Example 2) we have that:*

	$\mathcal{O}_1 \longrightarrow \mathcal{O}_2$		$\mathcal{O}_2 \longrightarrow \mathcal{O}_1$
◇	$\mathrm{Rewritten} = \{\alpha_3\}$	◇	$\mathrm{Rewritten} = \{\beta_9\}$
◇	$\mathrm{Strengthened} = \{\alpha_1\}$	◇	$\mathrm{Weakened} = \{\beta_7, \beta_9\}$
◇	$\mathrm{Redundant} = \{\alpha_1, \alpha_3, \alpha_5, \alpha_7\}$	◇	$\mathrm{Redundant} = \{\beta_1, \beta_4, \beta_5, \beta_7, \beta_9\}$

Note that the existence of a rewritten axiom from $\mathcal{O}_1$ to $\mathcal{O}_2$ does not imply that the same holds in the opposite direction. This is applicable to all categories. Also we can have that an axiom is in more than one categorical set, exemplified as follows:

Rewritten and redundant *The axiom α_3 has been rewritten from $\mathcal{O}_1$ to $\mathcal{O}_2$. The justification for α_3 is $\mathcal{J}_1 = \{\beta_4, \beta_5\}$, which is categorised as a rewrite since $\alpha_3 \models \mathcal{J}_1$. However, since $\{\beta_4, \beta_5\} \in$* $\mathrm{IneffectualAdditions}$*, $\mathcal{J}_1$ also indicates a redundancy. So the axiom α_3 is part rewritten part redundant.*

Strengthened and redundant *Consider axiom α_1; we can see that $\mathcal{O}_2 \models \alpha_1$. A justification $\mathcal{J}_1$ for α_1 is $\mathcal{J}_1 = \{\beta_2, \beta_3\}$, which indicates a strengthening (since* $\beta_2 \in \mathrm{EffectualAdditions}$*), as well as a redundancy ($\beta_3 \in \mathcal{O}_1 \cap \mathcal{O}_2$). Another justification $\mathcal{J}_2 = \{\beta_1, \beta_3\}$ indicates a strict redundancy;* $\beta_1 \in \mathrm{IneffectualAdditions}$.

Rewritten, weakened and redundant *Axiom β_9 is categorised as rewritten, weakened and redundant. A justification for β_9 is $\mathcal{J}_1 = \{\alpha_5, \alpha_7\}$, where $\beta_9 \models \mathcal{J}_1$, pointing to a rewrite. We also have that* $\{\alpha_5, \alpha_7\} \in \mathrm{IneffectualRemovals}$*, therefore being categorised as redundant as well. A second justification is $\mathcal{J}_2 = \{\alpha_8\}$, and since* $\alpha_8 \in \mathrm{EffectualRemovals}$*, β_9 is categorised as weakened.*

While the logical diff identifies those logically ineffectual axioms in the difference, it does not suggest on the intent of change or present appropriate reasons for it, i.e. justifications. With the categorisation method described, users have, at the very least, an indicator as to why such axioms have no impact. Note that these categories are merely suggestive of the developers' intent. In order to ensure the real intent one would require either a detailed edit-based diff or contact with the ontology developers.

4 Empirical results

In order to substantiate our approach to ontology diffing, we carried out a diachronic study of the NCIt using the methods described. The NCIt archive[3] contains 88 versions of the ontology in OWL format, two of which were unparsable (releases 05.03F and 05.04d) with the OWL API,[4] and consequently Protégé.[5] The experiment machine is an Intel Xeon Quad-Core 3.20GHz, with 12Gb DDR3 RAM dedicated to the Java Virtual Machine (JVM v1.5). The system runs Mac OS X 10.6.7, and all tests were run using the OWL API (v3.1). All gathered test data is available from `http://owl.cs.manchester.ac.uk/ncit`, a part of it is published on Google Public Data Explorer,[6] and can be visualised at `http://bit.ly/jFKU3R`.

4.1 Axioms Difference

The logical difference throughout the NCIt time-line consists mostly of subclass axioms (see Figure 1, and for complete results the mentioned website), with an average of 75% (excluding $\mathcal{O}_{14}$ and $\mathcal{O}_{16}$). The average proportion of logical changes is 15%, and the remaining are annotation changes. It should be noted that, despite the large number of annotations, NCIt developers devoted considerable effort towards the logical part of the ontology. Version $\mathcal{O}_6$ is a curious case, where a large number of classes (5170) were renamed,[7] and around 220,000 annotations and 14,418 subclass axioms were deleted. This indicates a possible re-modelling, or mass-renaming of classes in the NCIt at this point. More evidence to support this includes the addition of 30,859 subclass axioms, 9,070 classes and 23 object properties (and roughly 240,000 entity annotations). Similarly in $\mathcal{O}_{25}$ a series of changes were carried out to the subsumption hierarchy, with the removal of 8,231 subclass axioms and 2,899 equivalent class axioms compared to the previous version, and also the addition of 10,591 subclass axioms and 3,011 equivalent class axioms.

There is a fair amount of ineffectual removals in the corpus, reaching values of 93% in $\mathcal{O}_{29}$ or 97% in $\mathcal{O}_{16}$, and with an average of 35% of all logical removals (see Figure 1). Out of these ineffectual removals 92% turned out to be strengthened axioms (e.g. $\mathcal{O}_{27}$ has 3,104 strengthened axioms out of 3,843 removals), while 42% were removed redundancies. On average 5% of logical additions are ineffectual, yet there are some high values such as 61% in $\mathcal{O}_{24}$. Among these 73% are added redundancies, and 82% are weakened axioms. We also identified a number of rewrites in the corpus. Particularly

[3] `http://evs.nci.nih.gov/ftp1/NCI_Thesaurus`
[4] `http://owlapi.sourceforge.net/`
[5] `http://protege.stanford.edu/`
[6] `http://www.google.com/publicdata/home`
[7] Since throughout the NCIt evolution no classes are removed.

from $\mathcal{O}_{32}$ to $\mathcal{O}_{33}$ there are 227 rewritten axioms, typically taking a form as shown in Example 4.

Example 4 $A \equiv B \sqcap (\exists r.D) \sqcap (\exists s.F) \sqcap (\forall t.G)$ *rewritten into:*
$A \equiv B \sqcap ((\exists r.D) \sqcap (\exists s.F)) \sqcap (\forall t.G)$

This kind of change is not only syntactic but also trivial and easily detected. While ideally the underlying structural diff would not include these, at least with our categorisation and alignment with source axioms, it is easy to spot and recognize the triviality. One can also argue that certain ineffectual changes are in fact refactorings of one version into another, albeit in the case of strengthened and weakened axioms one could say that the intention was exactly that but turned out not to have the desired effect. The distinction here should be made that the strengthening of an axiom does not necessarily mean strengthening of the ontology. Consider an ontology $\mathcal{O}_1 = \{\alpha_1 : A \sqsubseteq B, \alpha_2 : A \sqsubseteq C\}$, and a change of α_1 into $A \sqsubseteq B \sqcap C$. The axiom α_1 was strengthened, but the resulting ontology $\mathcal{O}_2 = \{\alpha_1 : A \sqsubseteq B \sqcap C, \alpha_2 : A \sqsubseteq C\}$ was not. However, if we change $\alpha_2 \in \mathcal{O}_2$ into $A \sqsubseteq C \sqcap D$, then we can say both the axiom α_2 and the ontology $\mathcal{O}_2$ are strengthened.

We noted a recurring trend throughout the NCIt corpus, which is the addition of redundancies. This trend has more incidence up until $\mathcal{O}_8$, but there are high values in the rest of the corpus as well, such as $\mathcal{O}_{35}$ with 174 added redundant axioms (see Figure 1). The highest value found is in $\mathcal{O}_{17}$, where 482 redundant axioms were added. Upon investigating this phenomenon, we found that such added redundancies are, in most or all cases, entailments from previous versions. These entailments are those derived from the transitivity of the subclass relationship, e.g. $\mathcal{O}_1 = \{\alpha_1 : A \sqsubseteq \exists r.B, \alpha_2 : C \sqsubseteq A\}$, $\mathcal{O}_2 = \{\alpha_1, \alpha_2, \alpha_3 : C \sqsubseteq \exists r.B\}$. From the example we see that α_3 is redundant; $C \sqsubseteq A$ suffices for $C \sqsubseteq \exists r.B$ to hold.

Overall the average of ineffectual changes is 13%, while the remaining are effectual. However there are cases where the number of ineffectual changes is quite high, such as $\mathcal{O}_{24}$ where 52% of logical changes are ineffectual, as well as $\mathcal{O}_{27}$, $\mathcal{O}_{29}$ and $\mathcal{O}_{30}$ with 48% each. In retrospect this is a high amount of changes that would go unexplained by existing diffs, and while structural diff captures this it does not analyse the logical impact of such changes.

4.2 Reasoner Performance

It is often the case that, for reasoner testing, only a few or even one ontology version is tested against. There is no reported reasoner benchmark using a corpus of the same kind as the one here described. So, in the process of analysing the NCIt, we evaluated how modern reasoners handle all published OWL versions of the NCIt. Three major DL reasoners were put to the test; FaCT++ (v1.5.1), Pellet (v2.2.2) and HermiT (v1.3.3). Since we also possess the axioms in the difference between NCIt versions, this allows us to test incremental reasoning as well.[8] In Figure 2 we plot the reasoning times in a

[8] As implemented within Pellet.

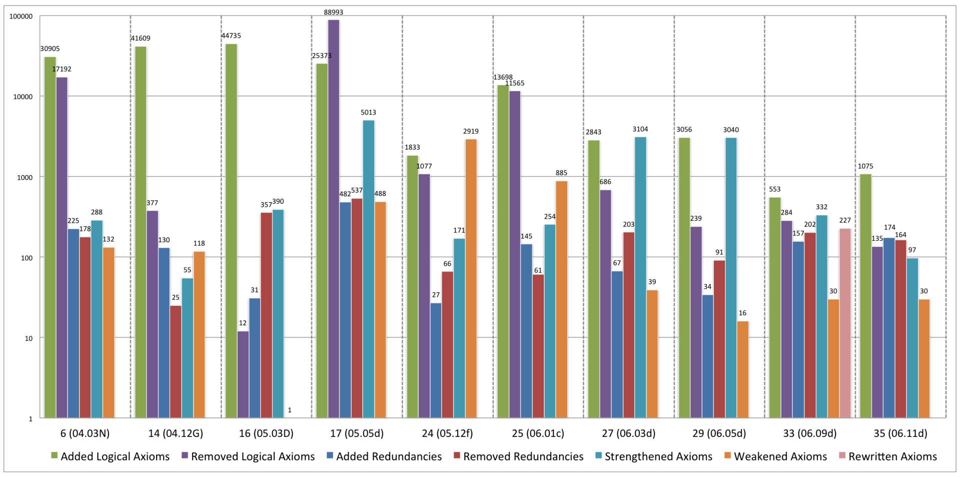


Fig. 1. Logical diff across selected versions of the NCIt (number of axioms).

logarithmic scale of each reasoner, comprising consistency checking, classification and concept satisfiability (denoted $\mathrm{RT}(\mathcal{O})$). Out of the three reasoners put to test, FaCT++ behaves consistently faster than Pellet and HermiT ($\mathcal{O}_{14}$ and $\mathcal{O}_{16}$ aside).

This performance test also shows that, to some degree, incremental reasoning provides a big advantage when handling the NCIt (or other large ontologies) in terms of reasoning time. However it did not terminate upon classifying $\mathcal{O}_{14}$ and, like HermiT,[9] $\mathcal{O}_{16}$. This is due to the abundance of individuals: incremental reasoning is based on locality-based modules [1], and these behave poorly in the presence of individuals. Aside from these two cases, the timings gathered using the incremental classifier were consistently below 5 seconds per version, across the corpus.

5 Culprit Finding

Upon completing the reasoner performance test we noted that, from $\mathcal{O}_{79}$ to $\mathcal{O}_{80}$ (in Figure 2), there is a significant performance improvement in HermiT. While our initial premise was to categorise logical diff-based impact between ontologies, now we encounter another problem: identifying and dissecting performance impact. We ascertained that the source of the bad performance is in the diff removals between those versions ($R = \mathrm{Removals}(\mathcal{O}_{79}, \mathcal{O}_{80})$), as with the additions of $\mathcal{O}_{80}$ the reasoning time was substantially lower. In order to investigate this phenomenon, we started with a brute-force *culprit finding* approach: for each axiom $\alpha \in R$ check if $\mathrm{RT}(\mathcal{O}_{80} \cup \{\alpha\}) \gg \mathrm{RT}(\mathcal{O}_{80})$. The size of R is 4,583 axioms, making this an expensive approach. It is also naive in the sense that culprits are not necessarily singleton sets. Nevertheless we examined $\mathrm{RT}(\mathcal{O}_{80} \cup \{\alpha \in R\})$ and found 13 (effectual) axioms which yield reasoning times ranging from 76 to 8,490 seconds. Surprisingly adding all

[9] HermiT returns a "StackOverflowError" when classifying $\mathcal{O}_{16}$, both in Protégé and OWL API.

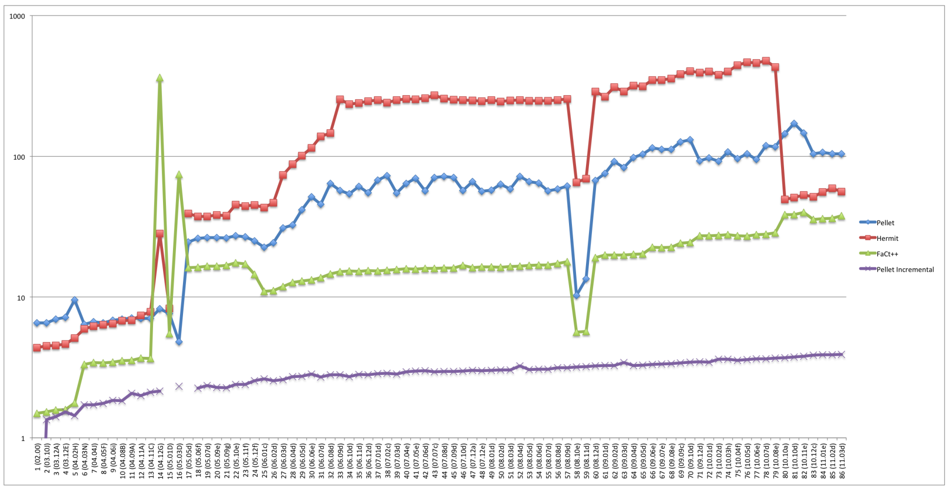


Fig. 2. Reasoner performance across NCIt (in seconds).

13 axioms to $\mathcal{O}_{80}$ results in a reasoning time of little over 9 hours. Thus some of the non-culprit additions exhibit a protective effect.

However, this approach is not only computationally expensive, but also relies on the existence of a diff which is not always available. We might want, given an "unmanageable" ontology, to find a subset thereof with which one can work with. As such we carried out a test partly based on the method described in [14], wherein we test the satisfiability checking time of each concept in the ontology. Such a test may be suggestive of the amount of time the reasoner spends on those concepts during classification (our culprit finding method is described in Algorithm 1). In order to extract a logically coherent subset of the ontology, which would be useful for repairing the culprit, we use the notion of a locality-based module [1]. We found a total of 12 concepts which have satisfiability checking times far greater than the average (see Table 2). The locality-based modules for the signature of the usage closure of each concept are significantly smaller than $\mathcal{O}_{79}$, the largest of which has 4,305 axioms (out of 116,587 logical axioms in $\mathcal{O}_{79}$). We found 9 modules $\mathcal{M}_i$ for which $\mathrm{RT}(\mathcal{O}_{79} \setminus \mathcal{M}_i)$ is nearly an order of magnitude faster than $\mathrm{RT}(\mathcal{O}_{79})$ ($\mathrm{RT}(\mathcal{O}_{79})$ = 430 seconds).

6 Discussion and Outlook

We have demonstrated with the diachronic study of the NCIt that merely syntactic diffs do not provide nearly enough insight into the impact of changes carried out, since logical differences are not identified. We found that ineffectual changes exist and account for a significant amount of logical changes throughout the NCIt. Such changes are discarded by semantic diffs, yet we show that they may provide helpful modelling insights. The axiom categorisation we devised allows ontology engineers to understand the lack

Algorithm 1 Identify subsets of an ontology $\mathcal{O}$ for which reasoning times are considerably better than the original ontology.

Input: Ontology $\mathcal{O}$
Output: Set of modules S, wherein for each $\mathcal{M}_i \in S$: $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}_i) \ll \mathrm{RT}(\mathcal{O})$

```
$S \leftarrow \emptyset$; $BadConcepts \leftarrow \emptyset$
for all concepts $C \in \widetilde{\mathcal{O}}$ do
  $Times \leftarrow Times \cup \langle C, SATtime(C) \rangle$
end for
for all $C \in \widetilde{\mathcal{O}}$ do
  if $SATtime(C) \geq$ average($SATtimes \in Times$) $\times 50$ then
    $BadConcepts \leftarrow BadConcepts \cup C$
  end if
end for
for all $C \in BadConcepts$ do
  $\Sigma = \{$terms $t \in$ Usage$(C)\}$
  $\mathcal{M} = \top\bot^*$-$mod(\Sigma)$
  if $\mathrm{RT}(\mathcal{O} \setminus \mathcal{M}) \ll \mathrm{RT}(\mathcal{O})$ then
    $S \leftarrow S \cup \mathcal{M}$
  end if
end for
return $S$
```

Concept	#$\mathcal{M}_i$	HermiT-RT$(\mathcal{O} \setminus \mathcal{M}_i)$	Pellet-RT$(\mathcal{O} \setminus \mathcal{M}_i)$
Cerebral_Glioblastoma	3029	56.7	38.1
TP53_Gene	3933	50.4	61.5
TP53_wt_Allele	3871	51.8	44.3
Erlotinib_Paclitaxel_Trastuzumab	4021	53.9	94.6
Tumor_Protein-p53	3894	51.4	102.2
Platelet-Derived_Growth_Factor_Receptor-Like_Protein	3201	54.8	60.6
HRAS_wt_Allele	3302	63.1	44.2
p21_H-Ras_Protein	3329	62.9	89.7
AC-T-T_Regimen	4305	50.7	97.2

Table 2. Extracted culprits and corresponding concepts found in $\mathcal{O}$ (time in seconds).

of impact of their changes, and possibly refine these before publishing newer versions, particularly if redundancies are present.

From our structural analysis, we were able to gain considerable insight into the NCIt and its evolution. By looking at the entire history, it became relatively straightforward to identify tool artefacts and significant events and thus to disentangle accidental and essential features of the ontology. We are currently confirming our interpretation of various events with the EVS and thus far it conforms to their understanding of the history. Such an analysis is proving useful to the EVS as they find instances of the OWL version that do not correspond with their intent, and thus allowing them to publish corrections.

In the future we plan to apply a similar categorization to logically effectual changes. We also intend to examine the stability of entailments, i.e., whether an entailment persists throughout some or all NCIt versions. Finally, more elaborate forms of structural analysis, such as examining the justificatory structure [9], hold great promise for exposing the axiomatic richness of the modelling.

The reasoner performance results identify areas of performance weakness that would not have been evident using standard "grab a version" methods. Furthermore, we demonstrate the advantage (in terms of time) of using incremental reasoning for ontology engineering tasks, especially when large and complex ontologies are involved. We found in the NCIt corpus a realistic case for performance impact analysis, based on which we identified a number of meaningful culprits. The preliminary culprit finding methods and results described indicate that this approach works reasonably well. However the question of how to present these culprits to, and validate our approach with users still remains.

References

1. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research 31 (2008)
2. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. of Web Semantics (2008)
3. Franconi, E., Meyer, T., Varzinczak, I.: Semantic diff as the basis for knowledge base versioning. In: Proc. of NMR-10 (2010)
4. Gatens, W., Konev, B., Ludwig, M., Wolter, F.: Versioning based on logical difference for lightweight description logic terminologies. In: Proc. of ARCOE-11 (2011)
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR-06 (2006)
6. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga Llavori, R.: Building ontologies collaboratively using ContentCVS. In: Proc. of DL 2009. CEUR (`http://ceur-ws.org/`), vol. 477 (2009)
7. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga Llavori, R.: Supporting concurrent ontology development: Framework, algorithms and tool. Data and Knowledge Engineering 70(1) (2011)
8. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., Hendler, J.: Swoop: A Web ontology editing browser. J. of Web Semantics 4(2) (2006)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Proc. of ISWC/ASWC (2007)
10. Konev, B., Lutz, C., Walther, D., Wolter, F.: Logical difference and module extraction with CEX and MEX. In: Proc. of DL 2008. CEUR (`http://ceur-ws.org/`), vol. 353 (2008)
11. Křemen, P., Abrahamčík, J., Pufler, J., Šmíd, M.: OWLDiff (2008), `http://krizik.felk.cvut.cz/km/owldiff/`
12. Noy, N.F., Musen, M.A.: PROMPTDIFF: A fixed-point algorithm for comparing ontology versions. In: Proc. of AAAI-02 (2002)
13. W3C OWL Working Group: OWL 2 Web Ontology Language: Document overview. W3C Recommendation (27 Oct 2009), `http://www.w3.org/TR/owl2-syntax/`
14. Wang, T.D., Parsia, B.: Ontology performance profiling and model examination: First steps. In: Proc. of ISWC/ASWC-07. LNCS, vol. 4825. Springer-Verlag (2007)

Rewriting Ontological Queries into Small Nonrecursive Datalog Programs[⋆]

Georg Gottlob[1] and Thomas Schwentick[2]

[1] Department of Computer Science, University of Oxford gottlob@cs.ox.ac.uk
[2] Fakultät für Informatik, TU Dortmund thomas.schwentick@udo.edu

Abstract. We consider the setting of ontological database access, where an A-box is given in form of a relational database D and where a Boolean conjunctive query q has to be evaluated against D modulo a T-box Σ formulated in DL-Lite or Linear Datalog$^{\pm}$. It is well-known that (Σ, q) can be rewritten into an equivalent nonrecursive Datalog program P that can be directly evaluated over D. However, for Linear Datalog$^{\pm}$ or for DL-Lite versions that allow for role inclusion, the rewriting methods described so far result in a nonrecursive Datalog program P of size exponential in the joint size of Σ and q. This gives rise to the interesting question of whether such a rewriting necessarily needs to be of exponential size. In this paper we show that it is actually possible to translate (Σ, q) into a polynomially sized equivalent nonrecursive Datalog program P.

1 Introduction

This paper is about query rewriting in the context of ontological database access. Query rewriting is an important new optimization technique specific to ontological queries. The essence of query rewriting, as will be explained in more detail below, is to compile a query and an ontological theory (usually formulated in some description logic or rule-based language) into a target query language that can be directly executed over a relational database management system (DBMS). The advantage of such an approach is obvious. Query rewriting can be used as a preprocessing step for enabling the exploitation of mature and efficient existing database technology to answer ontological queries. In particular, after translating an ontological query into SQL, sophisticated query-optimization strategies can be used to efficiently answer it. However, there is a pitfall here. If the translation inflates the query excessively and creates from a reasonably sized ontological query an enormous exponentially sized SQL query (or SQL DDL program), then the best DBMS may be of little use.

Main results. We show that polynomially sized query rewritings into nonrecursive Datalog exist in specific settings. Note that nonrecursive Datalog can be efficiently translated into SQL with view definitions (SQL DDL), which, in turn, can be directly executed over any standard DBMS. Our results are — for the time being — of theoretical nature and we do not claim that they will lead to better practical algorithms. This will be studied via implementations in the next future. Our main result applies to the

[⋆] Future improvements and extended versions of this paper will be published in arXive-CORR at http://arxiv.org/abs/1106.3767 setting where ontological constraints are formulated in terms of *tuple-generating dependencies (tgds)*, and we make heavy use of the well-known *chase* procedure [17, 14]. For definitions, see Section 2. The result after chasing a tgd set Σ over a database D is denoted by $chase(D,\Sigma)$.

Consider a set Σ of tgds and a database D over a joint signature $\mathcal{R}$. Let q be a Boolean conjunctive query (BCQ) issued against (D,Σ). We would like to transform q into a nonrecursive Datalog query P such that $(D,\Sigma) \models q$ iff $D \models P$. We assume here that P has a special propositional goal $goal$, and $D \models P$ means that $goal$ is derivable from P when evaluated over D. Let us define an important property of classes of tgds.

Definition 1. Polynomial witness property (PWP). *The PWP holds for a class $\mathcal{C}$ of tgds if there exists a polynomial γ such that, for every finite set $\Sigma \subseteq \mathcal{C}$ of tgds and each BCQ q, the following holds: for each database D, whenever $(D,\Sigma) \models q$, then there is a sequence of at most $\gamma(|\Sigma|, |q|)$ chase steps whose atoms already entail q.*

Our main technical result, which is more formally stated in Section 3, is as follows.
Theorem 1. *Let Σ be a set of tgds from a class $\mathcal{C}$ enjoying the PWP. Then each BCQ q can be rewritten in polynomial time into a nonrecursive Datalog program P of size polynomial in the joint size of q and Σ, such that for every database D, $(D,\Sigma) \models q$ if and only if $D \models P$. Moreover, the arity of P is* $\max(a+2, 9)$, *where a is the maximum arity of any predicate symbol occurring in Σ, in case a sufficiently large linear order can be accessed in the database, or otherwise by* $O(\max(a+2, 9) \cdot \log m)$, *where m is the joint size of q and Σ.*

Other Results. From this result, and from already established facts, a good number of further rewritabliity results for other formalisms can be derived. In particular, we can show that conjunctive queries based on other classes of tgds or description logics can be efficiently translated into nonrecursive Datalog. Among these formalisms are: linear tgds, originally defined in [5] and equivalent to inclusion dependencies, various major versions of the well-known description logic DL-Lite [9, 20], and sticky tgds [8] as well as sticky-join tgds [6, 7]. For space reasons, we will just give an overview and very short explanations of how each of these rewritability results follows from our main theorem.

Structure of the Paper. The rest of the paper is structured as follows. In Section 2 we state a few preliminaries and simplifying assumptions. In Section 3, we explain the idea of the proof of the main result. Section 4, contains the other results following from the main result. A brief overview of related work concludes the paper in Section 5.

2 Preliminaries and Assumptions

We assume the reader to be familiar with the terminology of relational databases and the concepts of *conjunctive query (CQ)* and *Boolean conjunctive query (BCQ)*. For simplicity, we restrict our attention to Boolean conjunctive queries q. However, our results can easily be reformulated for queries with output, see the extended version of this paper [13]).

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (tgd)* σ is a first-order formula of the form $\forall \boldsymbol{X} \forall \boldsymbol{Y}\, \Phi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \exists \boldsymbol{Z}\, \Psi(\boldsymbol{X}, \boldsymbol{Z})$, where $\Phi(\boldsymbol{X}, \boldsymbol{Y})$ and $\Psi(\boldsymbol{X}, \boldsymbol{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted $body(\sigma)$

and $head(\sigma)$, respectively. We usually omit the universal quantifiers in tgds. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\boldsymbol{X}, \boldsymbol{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\boldsymbol{X}, \boldsymbol{Z})$ to atoms of D. All sets of tgds are finite here. We assume in the rest of the paper that every tgd has exactly one atom and at most one existentially quantified variable in its head. A set of tgds is in *normal form* if the head of each tgd consists of a single atom. It was shown in [4, Lemma 10] that every set Σ of TGDs can be transformed into a set Σ' in normal form of size at most quadratic in $|\Sigma|$, such that Σ and Σ' are equivalent with respect to query answering. The normal form transformation shown in [4] can be achieved in logarithmic space. It is, moreover, easy to see that this very simple transformation preserves the polynomial witness property.

For a database D for $\mathcal{R}$, and a set of tgds Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D,\Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ q to D and Σ, denoted $ans(q, D,\Sigma)$, is the set of all tuples $\underline{a}$ such that $\underline{a} \in q(B)$ for all $B \in mods(D,\Sigma)$. The *answer* for a BCQ q to D and Σ is *yes* iff the empty tuple is in $ans(q, D,\Sigma)$, also denoted as $D \cup \Sigma \models q$.

Note that, in general, query answering under tgds is undecidable [2], even when the schema and tgds are fixed [4]. Query answering is, however, decidable for interesting classes of tgds, among which are those considered in the present paper.

The *chase* procedure [17, 14] uses the following *oblivious* chase rule.

TGD CHASE RULE. Consider a database D for a relational schema $\mathcal{R}$, and a tgd σ on $\mathcal{R}$ of the form $\Phi(\boldsymbol{X}, \boldsymbol{Y}) \rightarrow \exists \boldsymbol{Z}\, \Psi(\boldsymbol{X}, \boldsymbol{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(\boldsymbol{X}, \boldsymbol{Y})$ to atoms of D. Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \boldsymbol{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \boldsymbol{Z}$, $h_1(Z_j) = z_j$, where z_j is a fresh null value (i.e., a Skolem constant) different from all nulls already introduced. The *application of* σ on D adds to D the atom $h_1(\Psi(\boldsymbol{X}, \boldsymbol{Z}))$ if not already in D (which is possible when $\boldsymbol{Z}$ is empty). ■

The chase algorithm for a database D and a set of tgds Σ consists of an exhaustive application of the tgd chase rule in a breadth-first (level-saturating) fashion, which leads as result to a (possibly infinite) chase for D and Σ. Each atom from the database D is assigned a *derivation level*. Atoms in D have derivation level 0. If an atom has not already derivation level $\leq i$ but can be obtained by a single application of a tgd via the chase rule from atoms having derivation level $\leq i$, then its derivation level is $i + 1$. The set of all atoms of derivation level $\leq k$ is denoted by $chase^k(D,\Sigma)$. The *chase* of D relative to Σ, denoted $chase(D,\Sigma)$, is then the limit of $chase^k(D,\Sigma)$ for $k \rightarrow \infty$.

The (possibly infinite) chase relative to tgds is a *universal model*, i.e., there exists a homomorphism from $chase(D,\Sigma)$ onto every $B \in mods(D,\Sigma)$ [11, 4]. This result implies that BCQs q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models q$ is equivalent to $chase(D,\Sigma) \models q$.

A *chase sequence* of length n based on D and Σ is a sequence of n atoms such that each atom is either from D or can be derived via a single application of some rule in Σ from previous atoms in the sequence. If S is such a chase sequence and q a conjunctive query, we write $S \models q$ if there is a homomorphism from q to the set of atoms of S.

We assume that every database has two constants, 0 and 1, that are available via the unary predicates $Zero$ and One, respectively. Moreover, each database has a binary predicate Neq such that Neq(a, b) is true precisely if a and b are distinct values.

We finally define *N-numerical databases*. Let D be a database whose domain does not contain any natural numbers. We define D_N as the extension of D by adding the natural numbers $0, 1, \ldots, N$ to its domain, a unary relation Num that contains exactly the numbers $1, \ldots, N$, binary order relations $Succ$ and $<$ on $0, 1, \ldots, N$, expressing the natural successor and "$<$" orders on N, respectively. [3] We refer to D_N as the *N-numerical extension* of D, and, a so extended database as *N-numerical database*. We denote the total domain of a numerical database D_N by $\text{dom}_N(D)$ and the non-numerical domain (still) by $\text{dom}(D)$. Standard databases can always be considered to be N-numerical, for some large N by the standard type *integer*, with the $<$ predicate (and even arithmetic operations). A number $maxint$ corresponding to N can be defined.

3 Main Result

Our main result is more formally stated as follows:

Theorem 1. *Let $\mathcal{C}$ be a class of tgds in normal form, enjoying the polynomial witness property and let γ be the polynomial bounding the number of chase steps (with $\gamma(n_1, n_2) \geq \max(n_1, n_2)$, for all naturals n_1, n_2). For each set $\Sigma \subseteq \mathcal{C}$ of tgds and each Boolean CQ q, one can compute in polynomial time a nonrecursive Datalog program P of polynomial size in $|\Sigma|$ and $|q|$, such that, for every database D it holds $D,\Sigma \models q$ if and only if $D \models P$. Furthermore:*

(a) For N-numerical databases D, where $N \geq \gamma(|\Sigma|, |q|)$, the arity of P is $\max(a + 2, 9)$, where a is the maximum arity of any predicate symbol occurring in Σ;
(b) otherwise (for non-numerical databases), the arity of P is $\mathcal{O}(\max(a + 2, 9) \cdot \log \gamma(|\Sigma|, |q|))$, where a is as above.

We note that N is polynomially bounded in $|\Sigma|$ and $|q|$ by the polynomial γ that only depends on $\mathcal{C}$. The rest of this section explains the basic ideas of the proof of this result. A more detailed proof is given in [13].

High-level idea of the proof. We first describe the high level idea of the construction of the Datalog program P. It checks whether there is a chase sequence $S = t_1, \ldots, t_N$ with respect to D and Σ and a homomorphism h from q to (the set of atoms of) S. To this end, P consists of one large rule r_{goal} of polynomial size in N and some shorter rules that define auxiliary relations and will be explained below.

The aim of r_{goal} is to guess the chase sequence S *and* the homomorphism q at the same time. We recall that N does not depend on the size of D but only on $|\Sigma|$ and $|q|$ and thus r_{goal} can well be as long as the chase sequence and q together. One of the advantages of this approach is that we only have to deal with those null values that are actually relevant for answering the query. Thus, at most N null values need to be represented.

[3] Of course, if $\text{dom}(D)$ already contains some natural numbers we can add a fresh copy of $\{0, 1, \ldots, N\}$ instead.

One might try to obtain r_{goal} by just taking one atom A_i for each tuple t_i of S and one atom for each atom of q and somehow test that they are consistent. However, it is not clear how consistency could possibly be checked in a purely conjunctive fashion.[4] There are two ways in which disjunctive reasoning is needed. First, it is not a priori clear on which previous tuples, tuple t_i will depend. Second, it is not a priori clear to which tuples of S the atoms of q can be mapped.

To overcome these challenges we use the following basic ideas.

(1) We represent the tuples of S (and the required tuples of D) in a symbolic fashion, utilizing the numerical domain.
(2) We let P compute auxiliary predicates that allow us to express disjunctive relationships between the tuples in S.

Example 1. We illustrate the proof idea with a very simple running example, shown in Figure 1.

(a) Σ :

σ_1: $R_1(X,Y) \to \exists Z\ R_4(X,Y,Z)$
σ_2: $R_2(Y,Z) \to \exists X\ R_4(X,Y,Z)$
σ_3: $R_3(X,Z) \to \exists Y\ R_4(X,Y,Z)$
σ_4: $R_4(X_1,Y_1,Z_1), R_4(X_2,Y_2,Z_2) \to R_5(X_1,Z_2)$

(b) $q : R_5(X,Y), R_3(Y,X)$

(c) D :

R_1	
a	b
c	d

R_2	
e	g

R_3	
g	a
g	h

Fig. 1. Simple example with (a) a set Σ of tgds, (b) a query q and (c) a database D.

A possible chase sequence in this example is shown in Figure 2(a). The mapping $X \mapsto a$ and $Y \mapsto g$, maps $R_5(X,Y)$ to t_5 and $R_3(Y,X)$ to t_6, thus satisfying q.

(a)

- t_1: $R_1(a,b)$
- t_2: $R_4(a,b,\perp_2)$
- t_3: $R_2(e,g)$
- t_4: $R_4(\perp_4,e,g)$
- t_5: $R_5(a,g)$
- t_6: $R_3(g,a)$

(b)

- t_1: $R_1(a,b,a)$
- t_2: $R_4(a,b,\perp_2)$
- t_3: $R_2(e,g,e)$
- t_4: $R_4(\perp_4,e,g)$
- t_5: $R_5(a,g,a)$
- t_6: $R_3(g,a,g)$

(c)

i	r_i	f_i	x_{i1}	x_{i2}	x_{i3}	s_i	c_{i1}	c_{i2}
1	1	0	a	b	a	0	0	0
2	4	1	a	b	2	1	1	1
3	2	0	e	g	e	0	0	0
4	4	1	4	e	g	2	3	3
5	5	1	a	g	a	4	2	4
6	3	0	g	a	g	0	0	0

Fig. 2. (a) Example chase sequence, (b) its extension and (c) its encoding. t_2 is obtained by applying σ_1 to t_1. Likewise t_4 and t_5 are obtained by applying σ_2 to t_3 and σ_4 to t_2 and t_4, respectively.

Notation and conventions. Let $\mathcal{C}$ be a class of tgds enjoying the PWP, let Σ be a set of tgds from $\mathcal{C}$, and let q be a BCQ. Let $R_1, \ldots R_m$ be the predicate symbols occurring in Σ or in q. We denote the number of tgds in Σ by ℓ.

[4] Furthermore, of course, there are no relations to which the atoms A_i could possible be matched.

Let $N := \gamma(|\Sigma|, |q|)$ where γ is as in Definition 1, thus N is polynomial in $|\Sigma|$ and $|q|$. By definition of N, if $(D,\Sigma) \models q$, then q can be witnessed by a chase sequence Γ of length $\leq N$. Our assumption that $\gamma(n_1, n_2) \geq \max(n_1, n_2)$, for every n_1, n_2, guarantees that N is larger than (i) the number of predicate symbols occurring in Σ, (ii) the cardinality $|q|$ of the query, and (iii) the number of rules in Σ.

For the sake of a simpler presentation, we assume that all relations in Σ have the same arity a and all rules use the same number k of tuples in their body. The latter can be easily achieved by repeating tuples, the former by filling up shorter tuples by repeating the first tuple entry. Furthermore, we only consider chase sequences of length N. Shorter sequences can be extended by adding tuples from D.

Example 2. Example 1 thus translates as illustrated in Figure 3. The (extended) chase sequence is shown in Figure 2 (b). The query q is now satisfied by the mapping $X \mapsto a$, $Y \mapsto g$, $U \mapsto g$, $V \mapsto a$, thus mapping $R_5(X, Y, X)$ to t_5 and $R_3(Y, X, Y)$ to t_6.

(a) Σ :

σ_1: $R_1(X, Y, X), R_1(X, Y, X) \rightarrow \exists Z\ R_4(X, Y, Z)$

σ_2: $R_2(Y, Z, Y), R_2(Y, Z, Y) \rightarrow \exists X\ R_4(X, Y, Z)$

σ_3: $R_3(X, Z, X), R_3(X, Z, X) \rightarrow \exists Y\ R_4(X, Y, Z)$

σ_4: $R_4(X_1, Y_1, Z_1), R_4(X_2, Y_2, Z_2) \rightarrow$
$R_5(X_1, Z_2, X_1)$

(b) $q : R_5(X, Y, U), R_3(Y, X, V)$

(c) D :

R_1		
a	b	a
c	d	c

R_2		
e	g	e

R_3		
g	a	g
g	h	g

Fig. 3. Modified example with (a) a set Σ of tgds, (b) a query q and (c) a database D.

Proof idea (continued). On an abstract level, the atoms that make up the final rule r_{goal} of P can be divided into three groups serving three different purposes. That is, r_{goal} can be considered as a conjunction $r_{\text{tuples}} \wedge r_{\text{chase}} \wedge r_{\text{query}}$. Each group is "supported" by a sub-program of P that defines relations that are used in r_{goal}, and we refer to these three subprograms as P_{tuples}, P_{chase} and P_{query}, respectively.

- The purpose of r_{tuples} is basically to lay the ground for the other two. It consists of N atoms that allow to guess the symbolic encoding of a sequence $S = t_1, \ldots, t_N$.
- The atoms of r_{chase} are designed to verify that S is an actual chase sequence with respect to D.
- Finally, r_{query} checks that there is a homomorphism from q to S.

$P_{\textbf{tuples}}$ **and** $r_{\textbf{tuples}}$**.** The symbolic representation of the tuples t_i of the chase sequence S uses numerical values to encode null values, predicate symbols R_i (by i), tgds $\sigma_j \in \Sigma$ (by j) and the number of a tuple t_i in the sequence (that is: i).

In particular, the symbolic encoding uses the following numerical parameters.[5]

- r_i to indicate the relation R_{r_i} to which the tuple belongs;
- f_i to indicate whether t_i is from D ($f_i = 0$) or yielded by the chase ($f_i = 1$);

[5] We use the names of the parameters as variable names in r_{goal} as well.

– Furthermore, $x_{i1}, \ldots, x_{ia}$ represent the attribute values of t_i as follows. If the j-th attribute of t_i is a value from $\mathrm{dom}(D)$ then x_{ij} is intended to be that value, otherwise it is a null represented by a numeric value.

Since each rule of Σ has at most one existential quantifier in its head, at each chase step, at most one new null value can be introduced. Thus, we can unambiguously represent the null value (possibly) introduced in the j-th step of the chase by the number j.

The remaining parameters s_i and $c_{i1}, \ldots, c_{ik}$ are used to encode information about the tgd and the tuples (atoms) in S that are used to generate the current tuple. More precisely, s_i is intended to be the number of the applied tgd σ_{s_i} and $c_{i1}, \ldots, c_{ik}$ are the tuple numbers of the k tuples that are used to yield t_i. In the example, e.g., t_5 is obtained by applying σ_4 to t_2 and t_4. The encoding of our running example can be found in Figure 2 (c).

We use a new relational symbol T of arity $a + k + 4$ not present in the schema of D for the representation of the tuples from S. Thus, r_{tuples} is just:
$T(1, r_1, f_1, x_{11}, \ldots, x_{1a}, s_1, c_{11}, \ldots, c_{1k}), \ldots,$
$$T(N, r_N, f_N, x_{N1}, \ldots, x_{Na}, s_N, c_{N1}, \ldots, c_{Nk}).$$
The sub-program P_{tuples} is intended to "fill" T with suitable tuples. Basically, T contains all encodings of tuples in D (with $f_i = 0$) and all syntactically meaningful tuples corresponding to possible chase steps (with $f_i = 1$).

P_{chase} and r_{chase}. The following kinds of conditions have to be checked to ensure that the tuples "guessed" by r_{tuples} constitute a chase sequence.

(1) For every i, the relation R_{r_i} of a tuple t_i has to match the head of its rule σ_{s_i}.
 – In the example, e.g., r_4 has to be 4 as the head of σ_2 is an R_4-atom.

(2) Likewise, for each i and j the relation number of tuple $t_{c_{ij}}$ has to be the relation number of the j-th atom of σ_{s_i}.
 – In the example, e.g., r_2 must be 4, as $c_{5,1} = 2$ and the first atom of $\sigma_{s_5} = \sigma_4$ is an R_4-atom.

(3) If the head of σ_{s_i} contains an existentially quantified variable, the new null value is represented by the numerical value i.
 – This is illustrated by t_4 in the example: the first position of the head of rule 2 has an existentially quantified variable and thus $x_{4,1} = 4$.

(4) If a variable occurs at two different positions in σ_{s_i} then the corresponding positions in the tuples used to produce t_i carry the same value.

(5) If a variable in the body of σ_{s_i} also occurs in the head of σ_{s_i} then the values of the corresponding positions in the body tuple and in t_i are equal.
 – Z_2 occurs in position 3 of the second atom of the body of σ_4 and in position 2 of its head. Therefore, $x_{4,3}$ and $x_{5,2}$ have to coincide (where the 4 is determined by $c_{5,2}$.

It turns out that all these tests can be done by r_{chase}, given some relations that are precomputed by P_{chase}. More precisely, we let P_{chase} specify a 4-ary predicate IfThen(X_1, X_2, U_1, U_2) that is intended to contain all tuples fulfilling the condition: if $X_1 = X_2$ then $U_1 = U_2$. Similar predicates are defined for conditions with two and three conjuncts in the IF-part. Their definition by Datalog rules is straightforward.

$P_{\textbf{query}}$ **and** $r_{\textbf{query}}$**.** Finally, we explain how it can be checked that there is a homomorphism from q to S. We explain the issue through the little example query $R_3(x, y) \wedge R_4(y, z)$. To evaluate this query, r_{query} makes use of two additional variables q_1 and q_2, one for each atom of q. The intention is that these variables bind to the numbers of the tuples that the atoms are mapped to. We have to make sure two kinds of conditions. First, the tuples need to have the right relation symbol and second, they have to obey value equalities induced by the variables of q that occur more than once.

The first kind of conditions is checked by adding atoms $\text{IfThen}(q_1, i, r_i, 3)$ and $\text{IfThen}(q_2, i, r_i, 4)$ to r_{query}, for every $i \leq N$. The second condition is checked similarly. As we do not need any further auxiliary predicates, P_{query} is empty.

This completes the description of P. Note that P is nonrecursive, and has polynomial size in the size of q and Σ. Furthermore, the arity of P is as required. This proves part (a) of Theorem 1.

In order to prove part (b), we must get rid of the numeric domain (except for 0 and 1). This is actually very easy. We just replace each numeric value by a logarithmic number of bits (coded by our 0 and 1 domain elements), and extend the predicate arities accordingly. As a matter of fact, this requires an increase of arity by a factor of $\log N = \mathcal{O}(\log |q|)$. This concludes our explanation of the proof ideas underlying Theorem 1.

Remark 1. Note that the evaluation complexity of the Datalog program obtained for case (b) is not significantly higher than the evaluation complexity of the program P constructed for case (a). For example, in the most relevant case of bounded arities, both programs can be evaluated in NPTIME combined complexity over a database D. In fact, it is well-known that the combined complexity of a Datalog program of bounded arity is in NPTIME (see [10]). But it is easy to see that if we expand the signature of such a program (and of the underlying database) by a logarithmic number of Boolean-valued argument positions (attributes), nothing changes, because the possible values for such vectorized arguments are still of polynomial size. It is just a matter of coding. In a similar way, the data complexity in both cases (a) and (b) is the same (PTIME).

Remark 2. It is easy to generalize this result to the setting where q is actually a union of conjunctive queries (UCQ).

4 Further Results Derived From the Main Theorem

We wish to mention some interesting consequences of Theorem 1 that follow easily from the above result after combining it with various other known results.

4.1 Linear TGDs

A linear tgd [5] is one that has a single atom in its rule body. The class of linear tgds is a fundamental one in the Datalog$^{\pm}$ family. This class contains the class of *inclusion dependencies*. It was already shown in [14] for inclusion dependencies that classes of linear tgds of bounded (predicate) arities enjoy the PWP. That proof carries over to linear tgds.

By Theorem 1, we then conclude:

Theorem 2. *Conjunctive queries under linear tgds of bounded arity are polynomially rewritable as nonrecursive Datalog programs in the same fashion as for Theorem 1. So are sets of inclusion dependencies of bounded arity.*

4.2 DL-Lite

A pioneering and highly significant contribution towards tractable ontological reasoning was the introduction of the *DL-Lite* family of description logics (DLs) by Calvanese et al. [9, 20]. *DL-Lite* was further studied and developed in [1].

A DL-lite theory (or TBox) $\Sigma = (\Sigma^-, \Sigma^+)$ consists of a set of negative constraints Σ^- such as key and disjointness constraints, and of a set Σ^+ of positive constraints that resemble tgds. As shown in [9], the negative constraints Σ^- can be compiled into a polymomially sized first-order formula (actually a union of conjunctive queries) of the same arity as Σ^- such that for each database and BCQ q, $(D,\Sigma) \models q$ iff $D \not\models \Sigma^-$ and $(D,\Sigma^+) \models q$. In (the full version of) [5] it was shown that for the main DL-Lite variants defined in [9], each Σ^+ can be immediately translated into an equivalent set of linear tgds of arity 2. By virtue of this, and the above we obtain the following theorem.

Theorem 3. *Let q be a CQ and let $\Sigma = (\Sigma^-, \Sigma^+)$ be a DL-Lite theory expressed in one of the following DL-Lite variants: DL-Lite$_{\mathcal{F},\sqcap}$, DL-Lite$_{\mathcal{R},\sqcap}$, DL-Lite$^+_{\mathcal{A},\sqcap}$, DLR-Lite$_{\mathcal{F},\sqcap}$, DLR-Lite$_{\mathcal{R},\sqcap}$, or DLR-Lite$^+_{\mathcal{A},\sqcap}$. Then Σ^+ can be rewritten into a nonrecursive Datalog program P such that for each database D, $(D,\Sigma^+) \models q$ iff $D \models P$. Regarding the arities of P, the same bounds as in Theorem 1 hold.*

4.3 Sticky and Sticky Join TGDs

Sticky tgds [6] and sticky-join tgds [6] are special classes of tgds that generalize linear tgds but allow for a limited form of join (including as special case the cartesian product). They allow one to express natural ontological relationships not expressible in DLs such as OWL. For space reasons, we do not define these classes here, and refer the reader to [8]. By results of [8], which will also be discussed in detail in a future extended version [13] of the present paper, both classes enjoy the Polynomial Witness Property. By Theorem 1, we thus obtain the following result:

Theorem 4. *Conjunctive queries under sticky tgds and sticky-join tgds over a fixed signature $\mathcal{R}$ are rewritable into polynomially sized nonrecursive Datalog programs of arity bounded as in Theorem 1.*

5 Related Work on Query Rewriting

Several techniques for query-rewriting have been developed. An early algorithm, introduced in [9] and implemented in the QuOnto system[6], reformulates the given query into a union of CQs (UCQs) by means of a backward-chaining resolution procedure.

[6] http://www.dis.uniroma1.it/ quonto/

The size of the computed rewriting increases exponentially w.r.t. the number of atoms in the given query. This is mainly due to the fact that unifications are derived in a "blind" way from every unifiable pair of atoms, even if the generated rule is superfluous. An alternative resolution-based rewriting technique was proposed by Peréz-Urbina et al. [19], implemented in the Requiem system[7], that produces a UCQs as a rewriting which is, in general, smaller (but still exponential in the number of atoms of the query) than the one computed by QuOnto. This is achieved by avoiding the useless unifications, and thus the redundant rules obtained due to these unifications. This algorithm works also for more expressive non-first-order rewritable DLs. In this case, the computed rewriting is a (recursive) Datalog query. Following a more general approach, Calì et al. [3] proposed a backward-chaining rewriting algorithm for the first-order rewritable Datalog$^{\pm}$ languages mentioned above. However, this algorithm is inspired by the original QuOnto algorithm, and inherits all its drawbacks. In [12], a rewriting technique for linear Datalog$^{\pm}$ into unions of conjunctive queries is proposed. This algorithm is an improved version of the one already presented in [3]. However, the size of the rewriting is still exponential in the number of query atoms.

Of more interest to the present work are rewritings into nonrecursive Datalog. In [15, 16] a polynomial-size rewriting into nonrecursive Datalog is given for the description logics DL-Lite$^{\mathcal{F}}_{horn}$ and DL-Lite$_{horn}$. For DL-Lite$^{\mathcal{N}}_{horn}$, a DL with counting, a polynomial rewriting involving aggregate functions is proposed. It is, moreover, shown in (the full version of) [15] that for the description logic DL-Lite$_{\mathcal{F}}$ a polynomial-size pure first-order query rewriting is possible. Note that neither of these logics allows for role inclusion, while our approach covers description logics with role inclusion axioms. Other results in [15, 16] are about *combined rewritings* where both the query and the database D have to be rewritten. A recent very interesting paper discussing polynomial size rewritings is [22]. Among other results, [22] provides complexity-theoretic arguments indicating that without the use of special constants (e.g, 0 and 1, or the numerical domain), a polynomial rewriting such as ours may not be possible. Rosati et al. [21] recently proposed a very sophisticated rewriting technique into nonrecursive Datalog, implemented in the Presto system. This algorithm produces a non-recursive Datalog program as a rewriting, instead of a UCQs. This allows the "hiding" of the exponential blow-up inside the rules instead of generating explicitly the disjunctive normal form. The size of the final rewriting is, however, exponential in the number of non-eliminable existential join variables of the given query; such variables are a subset of the join variables of the query, and are typically less than the number of atoms in the query. Thus, the size of the rewriting is exponential in the query size in the worst case. Relevant further optimizations of this method are given in [18].

Acknowledgment G. Gottlob's Work was funded by the EPSRC Grant EP/H051511/1 ExODA: Integrating Description Logics and Database Technologies for Expressive Ontology-Based Data Access. We thank the anonymous referees, as well as Roman Kontchakov, Carsten Lutz, and Michael Zakharyaschev for useful comments on an earlier version of this paper.

[7] http://www.comlab.ox.ac.uk/projects/requiem/home.html

References

1. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev, *The dl-lite family and relations*, J. Artif. Intell. Res. (JAIR) **36** (2009), 1–69.
2. Catriel Beeri and Moshe Y. Vardi, *The implication problem for data dependencies*, Proc. of ICALP, 1981, pp. 73–85.
3. A. Calì, G. Gottlob, and A. Pieris, *Query rewriting under non-guarded rules*, Proc. AMW, 2010.
4. Andrea Calì, Georg Gottlob, and Michael Kifer, *Taming the infinite chase: Query answering under expressive relational constraints*, Proc. of KR, 2008, pp. 70–80.
5. Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz, *A general datalog-based framework for tractable query answering over ontologies*, Proc. of PODS, 2009, pp. 77–86.
6. Andrea Calì, Georg Gottlob, and Andreas Pieris, *Advanced processing for ontological queries*, PVLDB **3** (2010), no. 1, 554–565.
7. ______, *Query answering under non-guarded rules in datalog+/-*, Proc. of RR, 2010, pp. 175–190.
8. ______, *Towards more expressive ontology languages: The query answering problem*, Tech. report, University of Oxford, Department of Computer Science, 2011, Submitted for publication - available from the authors.
9. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, *Tractable reasoning and efficient query answering in description logics: The DL-lite family*, J. Autom. Reasoning **39** (2007), no. 3, 385–429.
10. Evgeny Dantsin, Thomas Eiter, Gottlob Georg, and Andrei Voronkov, *Complexity and expressive power of logic programming*, ACM Comput. Surv. **33** (2001), no. 3, 374–425.
11. Alin Deutsch, Alan Nash, and Jeff B. Remmel, *The chase revisisted*, Proc. of PODS, 2008, pp. 149–158.
12. Georg Gottlob, Giorgio Orsi, and Andreas Pieris, *Ontological queries: Rewriting and optimization*, Proc. of ICDE, 2011.
13. Georg Gottlob and Thomas Schwentick, *Rewriting ontological queries into small nonrecursive datalog programs*, arXiv Computing Research Repository (CoRR) **arXiv:1106.3767** (2011), extended version, available at `http://arxiv.org/abs/1106.3767`.
14. David S. Johnson and Anthony C. Klug, *Testing containment of conjunctive queries under functional and inclusion dependencies*, J. Comput. Syst. Sci. **28** (1984), no. 1, 167–189.
15. Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev, *The combined approach to query answering in dl-lite*, KR (Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, eds.), AAAI Press, 2010.
16. ______, *The combined approach to ontology-based data access*, IJCAI, 2011.
17. David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv, *Testing implications of data dependencies.*, ACM Trans. Database Syst. **4** (1979), no. 4, 455–469.
18. Giorgio Orsi and Andreas Pieris, *Optimizing query answering under ontological constraints*, PVLDB, 2011, to appear.
19. H. Pérez-Urbina, B. Motik, and I. Horrocks, *Tractable query answering and rewriting under description logic constraints*, Journal of Applied Logic **8** (2009), no. 2, 151–232.
20. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati, *Linking data to ontologies*, J. Data Semantics **10** (2008), 133–173.
21. R. Rosati and A. Almatelli, *Improving query answering over DL-Lite ontologies*, Proc. KR, 2010.
22. R.Kontchakov S. Kikot, Carsten Lutz, and M. Zakharyaschev, *On (In)Tractability of OBDA with OWL2QL*, Proc. DL, 2011.

The Cognitive Complexity of OWL Justifications

Matthew Horridge, Samantha Bail, Bijan Parsia, Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{matthew.horridge|bails|bparsia|sattler@cs.man.ac.uk}

Abstract. In this paper, we present an approach to determining the cognitive complexity of justifications for entailments of OWL ontologies. We introduce a simple cognitive complexity model and present the results of validating that model via experiments involving OWL users. The validation is based on test data derived from a large and diverse corpus of naturally occurring justifications. Our contributions include validation for the cognitive complexity model, new insights into justification complexity, a significant corpus with novel analyses of justifications suitable for experimentation, and an experimental protocol suitable for model validation and refinement.

1 Introduction

A justification is a minimal subset of an ontology that is sufficient for an entailment to hold. More precisely, given $\mathcal{O} \models \eta$, $\mathcal{J}$ is a justification for η in $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ and, for all $\mathcal{J}' \subsetneq \mathcal{J}$, $\mathcal{J}' \not\models \eta$. Justifications are the dominant form of explanation in OWL,[1] and justification based explanation is widely deployed in popular OWL editors. The primary focus of research in this area has been on explanation for the sake of debugging problematic entailments [3], whether standard entailments, such as class unsatisfiability or ontology inconsistency, or user selected entailments such as arbitrary subsumptions and class assertions. The debugging task is naturally directed toward "repairing" the ontology and the use of "standard errors" further biases users toward looking for problems in the logic of a justification.

The Description Logic that underpins OWL, $\mathcal{SROIQ}$, is N2ExpTime-complete [5], which suggests that even fairly small justifications could be quite challenging to reason with. However, justifications are highly successful in the field, thus the computational complexity argument is not dispositive. We do observe often that certain justifications are difficult and frustrating to understand for ontology developers. In some cases, the difficulty is obvious: a large justification with over 70 axioms is going to be at best cumbersome however simple its logical structure. However, for many reasonably sized difficult justifications (e.g. of size 10 or fewer axioms) the source of cognitive complexity is not clearly known.

We present the results of several experiments into the cognitive complexity of OWL justifications. Starting from a simple cognitive complexity model, we test how well the model predicts error proportions for an entailment assessment task. We find that the

[1] Throughout this paper, "OWL" refers to the W3C's Web Ontology Language 2 (OWL 2).

model does fairly well with some notable exceptions. A follow-up study with an eye tracker and think aloud protocol supports our explanations for the anomalous behaviour and suggests both a refinement to the model and a limitation of our experimental protocol.

While there have been several user studies in the area of debugging [6,4], ontology engineering anti-patterns [9], and an exploratory investigation into features that make justifications difficult to understand [1], to the best of our knowledge there have not been any formal user studies that investigate the cognitive complexity of justifications.

2 Cognitive Complexity & Justifications

In psychology, there is a long standing rivalry between two accounts of human deductive processes: (1) that people apply inferential rules [8], and (2) that people construct mental models [2].[2] In spite of a voluminous literature (including functional MRI studies), to date there is no scientific consensus [7], even for propositional reasoning.

Even if this debate were settled, it would not be clear how to apply it to ontology engineering. The reasoning problems that are considered in the literature are quite different from understanding how an entailment follows from a justification in an expressive logic. Furthermore, the artificiality of our problems may engage different mechanisms than more "natural" reasoning problems: e.g. even if mental models theory were correct, people *can* produce natural deduction proofs and might find that they outperform "reasoning natively". For ontology engineering, we do not need a true account of human deduction, but just need a way to determine how *usable* justifications are for our tasks. What is required is a theory of the *weak cognitive complexity* of justifications, not one of *strong cognitive complexity* [10].

A similar practical task is generating sufficiently difficult so-called "Analytical Reasoning Questions" (ARQs) problems in Graduate Record Examination (GRE) tests. In [7], the investigators constructed and validated a model for the complexity of answering ARQs via experiments with students. Analogously, we aim to validate a model for the complexity of "understanding" justificiations via experiments on modellers.

3 A Complexity Model

We have developed a cognitive complexity model for justification understanding. This model was derived partly from observations made during an exploratory study in which people attempted to understand justifications from naturally occuring ontologies, and partly from intuitions on what makes justifications difficult to understand. Table 1 describes the model, wherein $\mathcal{J}$ is the justification in question, η is the focal entailment, and each value is multiplied by its weight and then summed with the rest. The final value is a complexity score for the justification. Broadly speaking, there are two types of components: (1) structural components, such as C1, which require a syntactic analysis of a justification, and (2) semantic components, such as C4, which require entailment checking to reveal non-obvious phenomena.

[2] (1) can be crudely characterised as people use a natural deduction proof system and (2) as people use a semantic tableau.

Table 1. A Simple Complexity Model

Name	Base value	Weight
C1 AxiomTypes	Number of axiom types in $\mathcal{J}$ & η.	100
C2 ClassConstructors	Number of constructors in $\mathcal{J}$ & η.	10
C3 UniversalImplication	If an $\alpha \in \mathcal{J}$ is of the form $\forall R.C \sqsubseteq D$ or $D \equiv \forall R.C$ then 50 else 0.	1
C4 SynonymOfThing	If $\mathcal{J} \models \top \sqsubseteq A$ for some $A \in \mathsf{Signature}(\mathcal{J})$ and $\top \sqsubseteq A \notin \mathcal{J}$ and $\top \sqsubseteq A \neq \eta$ then 50 else 0.	1
C5 SynonymOfNothing	If $\mathcal{J} \models A \sqsubseteq \bot$ for some $A \in \mathsf{Signature}(\mathcal{J})$ and $A \sqsubseteq \bot \notin \mathcal{J}$ and $A \sqsubseteq \bot \neq \eta$ then 50 else 0.	1
C6 Domain&NoExistential	If $\mathsf{Domain}(R, C) \in \mathcal{J}$ and $\mathcal{J} \not\models E \sqsubseteq \exists R.D$ for some class expressions E and D then 50 else 0.	1
C7 ModalDepth	The maximum modal depth of all class expressions in $\mathcal{J}$.	50
C8 SignatureDifference	The number of distinct terms in $\mathsf{Signature}(\eta)$ not in $\mathsf{Signature}(\mathcal{J})$.	50
C9 AxiomTypeDiff	If the axiom type of η is not the set of axiom types of $\mathcal{J}$ then 50 else 0	1
C10 ClassConstructorDiff	The number of class constructors in η not in the set of constructors of $\mathcal{J}$.	1
C11 LaconicGCICount	The number of General Concept Inclusion axioms in a laconic version of $\mathcal{J}$	100
C12 AxiomPathLength	The number of maximal length expression paths[3] in $\mathcal{J}$ plus the number of axioms in $\mathcal{J}$ which are not in some maximal length path of $\mathcal{J}$	10

Components **C1** and **C2** count the number of different kinds of axiom types and class expression types as defined in the OWL 2 Structural Specification.[4] The more diverse the basic logical vocabulary is, the less likely that simple pattern matching will work and the more "sorts of things" the user must track.

Component **C3** detects the presence of universal restrictions where *trivial satisfaction* can be used to infer subsumption. Generally, people are often surprised to learn that if $\langle x, y\rangle \notin R^{\mathcal{I}}$ for all $y \in \Delta^{\mathcal{I}}$, then $x \in (\forall R.C)^{\mathcal{I}}$. This was observed repeatedly in the exploratory study.

Components **C4** and **C5** detect the presence of synonyms of $\top$ and $\bot$ in the signature of a justification where these synonyms are *not explicitly* introduced via subsumption or equivalence axioms. In the exploratory study, participants failed to spot synonyms of $\top$ in particular.

Component **C6** detects the presence of a domain axiom that is not paired with an (entailed) existential restriction along the property whose domain is restricted. This typically goes against peoples' expectations of how domain axioms work, and usually indicates some kind of non-obvious reasoning by cases. For example, given the two axioms $\exists R.\top \sqsubseteq C$ and $\forall R.D \sqsubseteq C$, the domain axiom is used to make a statement about objects that have R successors, while the second axiom makes a statement about those objects that do not have any R successors to imply that C is equivalent to $\top$. This is different from the typical pattern of usage, for example where $A \sqsubseteq \exists R.C$ and $\exists R.\top \sqsubseteq B$ entails $A \sqsubseteq B$.

Component **C7** measures maximum modal depth of sub-concepts in $\mathcal{J}$, which tend to generate multiple distinct but interacting propositional contexts.

Component **C8** examines the signature difference from entailment to justification. This can indicate confusing redundancy in the entailment, or synonyms of $\top$, that may not be obvious, in the justification. Both cases are surprising to people looking at such justifications.

[4] `http://www.w3.org/TR/owl2-syntax/`

Components **C9** and **C10** determine if there is a difference between the type of, and types of class expressions in, the axiom representing the entailment of interest and the types of axioms and class expressions that appear in the justification. Any difference can indicate an extra reasoning step to be performed by a person looking at the justification.

Component **C11** examines the number of subclass axioms that have a complex left hand side in a laconic version of the justification. Complex class expressions on the left hand side of subclass axioms in a laconic justification indicate that the conclusions of several intermediate reasoning steps may interact.

Component **C12** examines the number of obvious syntactic subsumption paths through a justification. In the exploratory study, participants found it very easy to quickly read chains of subsumption axioms, for example, $\{A \sqsubseteq B, B \sqsubseteq C, D \sqsubseteq D, D \sqsubseteq E\}$ to entail $A \sqsubseteq E$. This complexity component essentially increases the complexity when these kinds of paths are lacking.

The weights were determined by rough and ready empirical twiddling, without a strong theoretical or specific experimental backing. They correspond to our sense, esp. from the exploratory study, of sufficient reasons for difficulty.

4 Experiments

While the model is plausible and seems to behave reasonably well in applications, its validation is a challenging topic. In principle, the model is reasonable if it successfully predicts the difficulty an arbitrary OWL modeller has with an arbitrary justification sufficiently often. Unfortunately, the space of ontology developers and of OWL justifications (even of existing, naturally occurring ones) is large and heterogeneous enough to be difficult to randomly sample.

4.1 Design Challenges

To cope with the heterogeneity of users, any experimental protocol should require minimal experimental interaction, i.e. it should be executable over the internet from subjects' own machines with simple installation. Such a protocol trades access to subjects, over time, for the richness of data gathered. To this end, we adapted one of the experimental protocols described in [7] and tested it on a more homogeneous set of participants—a group of MSc students who had completed a lecture course on OWL.

While the general experimental protocol in [7] seems reasonable, there are some issues in adapting it to our case. In particular, in ARQs there is a restricted space of possible (non-)entailments suitable for multiple choice questions. That is, the wrong answers can straightforwardly be made plausible enough to avoid guessing. (The questions are, in essence, enumeration problems.) A justification inherently has one statement that it is a justification *for* (even though it will be a minimal entailing subset for others). Thus, there isn't a standard "multiple set" of probable answers to draw on. In the exam case, the primary task is successfully answering the question and the relation between that success and predictions about the test taker are outside the remit of the experiment (but there is an established account, both theoretically and empirically). In the justification case the standard primary task is "understanding" the relationship between the justification and the entailment. Without observation, it is impossible to distinguish between a participant who really "gets" it and one who merely acquiesces. In the exploratory study we performed to help develop the model, we had the participant rank the difficulty of the justification, but also used think aloud and follow-up questioning to verify the success in understanding by the participant. This is obviously not a minimal intervention, and requires a large amount of time and resources on the part of the investigators.

To counter this, the task was shifted from justification understanding task to something more measurable and similar to the question answering task in [7]. In particular, instead of presenting the justification/entailment pair *as* a justification/entailment pair and asking the participant to try to "understand" it, we present the justification/entailment pair as a set of axioms/candidate entailment pair and ask the participant to *determine* whether the candidate is, in fact, entailed. This diverges from the standard justification situation wherein the modeller knows that the axioms entail the candidate (and form a justification), but provides a metric that can be correlated with cognitive complexity, which is *error proportions*.

4.2 Justification Corpus

To cope with the heterogeneity of justifications, we derived a large sample of justifications from ontologies from several well known ontology repositories: The Stanford BioPortal repository[5] (30 ontologies plus imports closure), the Dumontier Lab ontology collection[6] (15 ontologies plus imports closure), the OBO XP collection[7] (17 ontologies plus imports closure) and the TONES repository[8] (36 ontologies plus imports closure). To be selected, an ontology had to (1) entail one subsumption between class names with at least one justification that (a) was not the entailment itself, and (b) contains axioms in that ontology (as opposed to the imports closure of the ontology), (2) be downloadable and loadable by the OWL API (3) processable by FaCT++.

While the selected ontologies cannot be said to generate a *truly representative* sample of justifications from the full space of possible justifications (even of those on the Web), they are diverse enough to put stress on many parts of the model. Moreover, most of these ontologies are actively developed and used and hence provide justifications that a significant class of users encounter.

For each ontology, the class hierarchy was computed, from which direct subsumptions between class names were extracted. For each direct subsumption, as many justifications as possible in the space of 10 minutes were computed (typically all justifications; time-outs were rare). This resulted in a pool of over 64,800 justifications.

While large, the actual logical diversity of this pool is considerably smaller. This is because many justifications, for different entailments, were of exactly the same "shape". For example, consider $\mathcal{J}_1 = \{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$ and $\mathcal{J}_2 = \{F \sqsubseteq E, E \sqsubseteq G\} \models F \sqsubseteq G$. As can be seen, there is an injective renaming from $\mathcal{J}_1$ to $\mathcal{J}_2$, and $\mathcal{J}_1$ is

[5] `http://bioportal.bioontology.org`
[6] `http://dumontierlab.com/?page=ontologies`
[7] `http://www.berkeleybop.org/ontologies/`
[8] `http://owl.cs.manchester.ac.uk/repository/`

therefore *isomorphic* with $\mathcal{J}_2$. If a person can understand $\mathcal{J}_1$ then, with allowances for variations in name length, they should be able to understand $\mathcal{J}_2$. The initial large pool was therefore reduced to a smaller pool of 11,600 *non-isomorphic justifications*.

4.3 Items and Item Selection

Each experiment consists of a series of test items (questions from a participant point of view). A test *item* consists of a *set of axioms*, one *following axiom*, and a *question*, "Do these axioms entail the following axiom?". A participant *response* is one of five possible answers: "Yes" (it is entailed), "Yes, but not sure", "Not Sure", "No, but not sure", "No" (it is not entailed). From a participant point of view, any item may or may not contain a justification. However, in our experiments, every item was, in fact, a justification.

It is obviously possible to have non-justification entailing sets or non-entailing sets of axioms in an item. We chose against such items since (1) we wanted to maximize the number of actual justifications examined (2) justification understanding is the actual task at hand, and (3) it is unclear how to interpret error rates for non-entailments in light of the model. For some subjects, esp. those with little or no prior exposure to justifications, it was unclear whether they understood the difference between the set merely being entailing, and it being minimal and entailing. We did observe one person who made use of this metalogical reasoning in the follow-up study.

Item Construction: For each experiment detailed below, test items were constructed from the pool of 11,600 non-isomorphic justifications. First, in order to reduce variance due primarily to size, justifications whose size was less than 4 axioms and greater than 10 axioms were discarded. This left 3199 (28%) justifications in the pool. In particular, this excluded large justifications that might require a lot of reading time, cause fatigue problems, or intimidate, and excluded very small justifications that tended to be trivial.[9]

For each justification in the pool of the remaining 3199 non-isomorphic justifications, the complexity of the justification was computed according to the model presented in Table 1, and then the justification was assigned to a complexity bin. A total of 11 bins were constructed over the range of complexity (from 0 to 2200), each with a complexity interval of 200. We discarded all bins which had 0 non-isomorphic justifications of size 4-10. This left 8 bins partitioning a complexity range of 200-1800.

Figure 1 illustrates a key issue. The bulk of the justifications (esp. without the trivial), both with and without isomorphic reduction, are in the middle complexity range. However, the model is not sophisticated enough that small differences (e.g. below a difference of 400-600) are plausibly meaningful. It is unclear whether the noise from variance in participant abilities would wash out the noise from the complexity model. In other words, just from reflection on the model, justifications whose complexity difference is 400 or less do not seem reliably distinguishable by error rates. Furthermore, non-isomorphism does not eliminate all non-significant logical variance. Consider a

[9] Note that, as a result, nearly 40% of all justifications (essentially, the 0-200 bin) have no representative in the pruned set (see Figure 2). Inspection revealed that most of these were trivial single axiom justifications (e.g. of the form $\{A \equiv B\} \models A \sqsubseteq B$ or $\{A \equiv (B \sqcap C)\} \models A \sqsubseteq B$, etc.

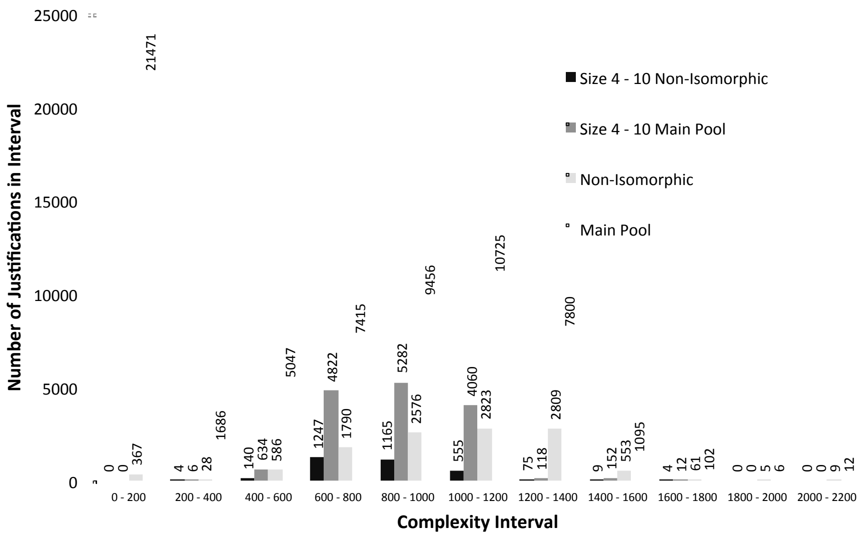
chain of two atomic subsumptions vs. a chain of three. They have the same basic logical structure, but are not isomorphic. Thus, we cannot yet say whether this apparent concentration is meaningful.

Since we did not expect to be able to present more than 6 items and keep to our time limits, we chose to focus on a "easy/hard" divide of the lowest three non-empty bins (200-800) and the highest three non-empty bins (1200-1800). While this limits the claims we can make about model performance over the entire corpus, it, at least, strengthens negative results. If error rates overall do not distinguish the two poles (where we expect the largest effect) then either the model fails or error rates are not a reliable marker. Additionally, since if there is an effect, we expect it to be largest in this scenario thus making it easier to achieve adequate statistical power.

Each experiment involved a fixed set of test items, which were selected by randomly drawing items from preselected spread of bins, as described below. Please note that the selection procedure changed in the light of the pilot study, but only to make the selection more challenging for the model.

The final stage of item construction was justification obfuscation. All non-logical terms were replaced with generated symbols. Thus, there was no possibility of using domain knowledge to understand these justifications. The names were all uniform, syntactically distinguishable (e.g. class names from property names) and quite short. The entailment was the same for all items (i.e. $C1 \sqsubseteq C2$). It is possible that dealing with these purely symbolic justifications distorted participant response from response in the field, even beyond blocking domain knowledge. For example, they could be alienating and thus increase error rates or they could engage less error prone pattern recognition.

Fig. 1. Justification Corpus Complexity Distribution



4.4 Pilot study

Participants: Seven members of a Computer Science (CS) Academic or Research Staff, or PhD Program, with over 2 years of experience with ontologies and justifications.

Materials and procedures: The study was performed using an in-house web based survey tool, which tracks times between all clicks on the page and thus records the time to make each decision.

The participants were given a series of test items consisting of 3 practice items, followed by 1 common easy item (**E1** of complexity 300) and four additional items, 2 ranked easy (**E2** and **E3** of complexities 544 and 690, resp.) and 2 ranked hard (**H1** and **H2** of complexities 1220 and 1406), which were randomly (but distinctly) ordered for each participant. The easy items were drawn from bins 200-800, and the hard items from bins 1200-1800. The expected time to complete the study was a maximum of 30 minutes, including the orientation, practice items, and brief demographic questionnaire (taken after all items were completed).

Results: Errors and times are given in Table 2(a). Since all of the items were in fact justifications, participant responses were recoded to success or failure as follows: Success = ("Yes" | "Yes, but not sure") and Failure = ("Not sure" | "No, Not sure" | "No"). Error proportions were analysed using Cochran's Q Test, which takes into consideration the pairing of successes and failures for a given participant. Times were analysed using two tailed paired sample t-tests.

An initial Cochran Q Test across all items revealed a strong significant difference in error proportions between the items [$Q(4) = 16.00, p = 0.003$]. Further analysis using Cochran's Q Test on pairs of items revealed strong statistically significant differences in error proportion between: **E1/H1** [$Q(1) = 6.00$, $p = 0.014$], **E1/H2** [$Q(1) = 6.00$, $p = 0.014$] **E2/H2** [$Q(1) = 5.00$, $p = 0.025$] and **E3/H2** [$Q(1) = 5.00$, $p = 0.025$]. The differences in the remaining pairs, while not exhibiting differences above $p = 0.05$, were quite close to significance, i.e. **E2/H1** [$Q(1) = 3.57$, $p = 0.059$] and **E3/H1** [$Q(1) = 2.70$, $p = 0.10$]. In summary, these error rate results were encouraging.

An analysis of times using paired sample t-tests revealed that time spent understanding a particular item is not a good predictor of complexity. While there were significant differences in the times for **E1/H1** [$p = 0.00016$], **E2/H1** [$p = 0.025$], and **E3/H1** [$p = 0.023$], there were no significant differences in the times for **E1/H2** [$p = 0.15$], **E2/H2** [$p = 0.34$] and **E3/H2** [$p = 0.11$]. This result was anticipated, as in the exploratory study people gave up very quickly for justifications that they felt they could not understand.

Table 2. Failures and times

(a) Pilot Study Items

Item	Failures	Mean Time (ms)	Time S.D. (ms)
E1	0	65839	39370
E2	1	120926	65950
E3	2	142126	61771
H1	6	204257	54796
H2	6	102774	88728

(b) Experiment 1

Item	Failures	Mean Time (ms)	Time S.D. (ms)
EM1	6	103454	68247
EM2	6	162928	87696
EM3	10	133665	77652
HM1	12	246835	220921
HM2	13	100357	46897
HM3	6	157208	61437

4.5 Experiment 1

Participants: 14 volunteers from a CS MSc class on OWL ontology modelling, who were given chocolate for their participation. Each participant had minimal exposure to OWL (or logic) before the class, but had, in the course of the prior 5 weeks, constructed or manipulated several ontologies, and received an overview of the basics of OWL 2, reasoning, etc. They did not receive any specific training on justifications.
Materials and procedures: The study was performed according to the protocol used in the pilot study. A new set of items were used. Since the mean time taken by pilot study participants to complete the survey was 13.65 minutes, with a standard deviation of 4.87 minutes, an additional hard justification was added to the test items. Furthermore, all of the items with easy justifications ranked easy were drawn from the highest easy complexity bin (bin 600-800). In the pilot study, we observed that the lower ranking easy items were found to be quite easy and, by inspection of their bins, we found that it was quite likely to draw similar justifications. The third bin (600-800) is much larger and logically diverse, thus is more challenging for the model.

The series consisted of 3 practice items followed by 6 additional items, 3 easy items(**EM1**, **EM2** and **EM3** of complexities: 654, 703, and 675), and 3 hard items (**HM1**, **HM2** and **HM3** of complexities: 1380, 1395, and 1406). The items were randomly ordered for each participant. Again, the expectation of the time to complete the study was a maximum of 30 minutes, including orientation, practice items and brief demographic questionnaire.
Results Errors and times are presented in Table 2(b). The coding to error is the same as in the pilot. An analysis with Cochran's Q Test across all items reveals a significant difference in error proportion [$Q(5) = 15.095, p = 0.0045$].

A pairwise analysis between easy and hard items reveals that there are significant and, highly significant, differences in errors between **EM1/HM1** [$Q(1) = 4.50$, $p = 0.034$], **EM1/HM2** [$Q(1) = 7.00, p = 0.008$], **EM2/HM1** [$Q(1) = 4.50, p = 0.034$], **EM2/HM2** [$Q(1) = 5.44, p = 0.02$], and **EM3/HM2** [$Q(1) = 5.44, p = 0.02$].

However, there were no significant differences between **EM1/HM3** [$Q(1) = 0.00$, $p = 1.00$], **EM2/HM3** [$Q(1) = 0.00, p = 1.00$], **EM3/HM3** [$Q(1) = 2.00, p = 0.16$] and **EM3/HM1** [$Q(1) = 0.67, p = 0.41$].

With regards to the nonsignificant differences between certain easy and hard items, there are two items which stand out: An easy item **EM3** and a hard item **HM3**, which are shown in Figure 2.

In line with the results from the pilot study, an analysis of times using a paired samples t-test revealed significant differences between some easy and hard items, with those easy times being significantly less than the hard times **EM1/HM1** [$p = 0.023$], **EM2/HM2** [$p = 0.016$] and **EM3/HM1** [$p = 0.025$]. However, for other pairs of easy and hard items, times were not significantly different: EM1/HM1 [$p = 0.43$], **EM2/HM1** [$p = 0.11$] and **EM3/HM2** [$p = 0.10$]. Again, time is not a reliable predictor of model complexity.
Anomalies in Experiment 1: Two items (**EM3** and **HM3**) did not exhibit their predicted error rate relations. For item **EM3**, we conjectured that a certain pattern of superfluous axiom parts in the item (not recognisable by the model) made it harder than the model predicted. That is, that the *model* was wrong.

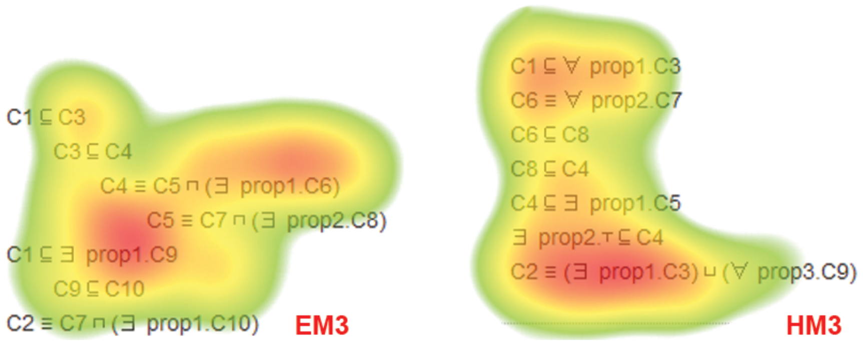
For item **HM3** we conjectured that the model correctly identifies this item as hard,[10] but that the MSc students answered "Yes" because of misleading pattern of axioms at the start and end of item **HM3**. The high "success" rate was due to an error in reasoning, that is, a *failure* in understanding.

In order to determine whether our conjectures were possible and reasonable, we conducted a follow-up study with the goal of observing the conjectured behaviours in situ. Note that this study does *not* explain what happened in Experiment 1.

4.6 Experiment 2

Participants: Two CS Research Associates and one CS PhD student, none of whom had taken part in the pilot study. All participants were very experienced with OWL.
Materials and procedures: Items and protocol were exactly the same as Experiment 1, with the addition of the think aloud protocol. Furthermore, the screen, participant vocalization, and eye tracking were recorded.
Results: With regard to **EM3**, think aloud revealed that all participants were distracted by the superfluous axiom parts in item **EM3**. Figure 2 shows an eye tracker heat map for the most extreme case of distraction in item **EM3**. As can be seen, hot spots lie over the superfluous parts of axioms. Think aloud revealed that all participants initially tried to see how the $\exists$prop1.C6 conjunct in the third axiom contributed to the entailment and struggled when they realised that this was not the case.

Fig. 2. Eye Tracker Heat Maps for **EM3** & **HM3**



In the case of **HM3**, think aloud revealed that none of the participants understood how the entailment followed from the set of axioms. However, two of them responded correctly and stated that the entailment did hold. As conjectured, the patterns formed by the start and end axioms in the item set seemed to mislead them. In particular, when disregarding quantifiers, the start axiom C1 $\sqsubseteq \forall$prop1.C3 and the end axiom C2 $\sqsubseteq \exists$prop1.C3 $\sqcup \ldots$ look very similar. One participant spotted this similarity and claimed that the entailment held as a result. Hot spots occur over the final axiom and the first axiom in the eye tracker heat map (Figure 2), with relatively little activity in the axioms in the middle of the justification.

[10] It had been observed to stymie experienced modellers in the field. Furthermore, it involves deriving a synonym for $\top$, which was not a move this cohort had experience with.

5 Discussion and Future Work

In this paper we presented a methodology for validating the predicted cognitive complexity of justifications. The main advantages of the experimental protocol used in the methodology is that minimal study facilitator intervention is required. This means that, over time, it should be possible to collect rich and varied data fairly cheaply and from geographically distributed participants. In addition to this, given a justification corpus and population of interest, the main experiment is easily repeatable with minimal resources and setup. Care must be taken in interpreting results and, in particular, the protocol is weak on "too hard" justifications as it cannot distinguish a model mislabeling from people failing for the wrong reason.

The cognitive complexity model that was presented in this paper fared reasonably well. In most cases, there was a significant difference in error proportion between model ranked easy and hard justifications. In the cases where error proportions revealed no difference better than chance, further small scale follow-up studies in the form of a more expensive talk-aloud study was used to gain an insight into the problems. These inspections highlighted an area for model improvement, namely in the area of superfluity. It is unclear how to rectify this in the model, as there could be justifications with superfluous parts that are trivial to understand, but the location and shape of superfluity seem an important factor.

The refinement and validation of our model is an ongoing task and will require considerably more experimental cycles. We plan to conduct a series of experiments with different cohorts as well as with an expanded corpus. We also plan to continue the analysis of our corpus with an eye to performing experiments to validate the model over the whole (for some given population).

References

1. M. Horridge, B. Parsia, and U. Sattler. Lemmas for justifications in OWL. In *DL 2009*.
2. P. N. Johnson-Laird and R. M. J. Byrne. *Deduction*. Psychology Press, 1991.
3. A. Kalyanpur, B. Parsia, E. Sirin, and B. Grau. Repairing unsatisfiable concepts in OWL ontologies. In *Proc. of ESWC-06*, pages 170–184, 2006.
4. A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, 3(4), 2005.
5. Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *Proc. KR 2008*. AAAI Press, 2008.
6. S. C. J. Lam. *Methods for Resolving Inconsistencies In Ontologies*. PhD thesis, Department of Computer Science, Aberdeen, 2007.
7. S. Newstead, P. Brandon, S. Handley, I. Dennis, and J. S. B. Evans. Predicting the difficulty of complex logical reasoning problems. *Psychology Press*, 12, 2006.
8. L. J. Rips. *The Psychology of Proof.* MIT Press, Cambridge, MA, 1994.
9. C. Roussey, O. Corcho, and L. Vilches-Blázquez. A catalogue of OWL ontology antipatterns. In *Proc. of K-CAP-09*, pages 205–206, 2009.
10. G. Strube. The role of cognitive science in knowledge engineering. In *Contemporary Knowledge Engineering and Cognition*, 1992.

Relaxed Abduction: Robust Information Interpretation for Incomplete Models

Thomas M. Hubauer[1,2], Steffen Lamparter[2], and Michael Pirker[2]

[1] Software, Technology, and Systems (STS)
Hamburg University of Technology, Hamburg, Germany
[2] Intelligent Systems and Control
Siemens Corporate Technology, Munich, Germany

Abstract. This paper introduces relaxed abduction, a novel non-standard reasoning task for description logics. Although abductive reasoning over description logic knowledge bases has been applied successfully to various information interpretation tasks, it typically fails to provide adequate (or even any) results when confronted with spurious information or incomplete models. Relaxed abduction addresses this flaw by ignoring such pieces of information automatically based on a joint optimization of the sets of explained observations and required assumptions. We present a method to solve relaxed abduction over $\mathcal{EL}^+$ TBoxes based on the notion of multi-criterion shortest hyperpaths.

Keywords: abduction, interpretation, non-standard reasoning

1 Introduction

Abduction was introduced in the late 19th century by Charles Sanders Pierce as an inference scheme aimed at deriving potential explanations for some observation [7]. It is conveniently expressed by the derivation rule

$$\frac{\phi \supset \omega \qquad \omega}{\phi}$$

which can be understood as an inversion of the modus ponens rule that permits to derive ϕ as a hypothetical explanation for the occurrence of ω, given that the presence of ϕ in some sense justifies ω. Note that this general formulation does not presuppose any causality between ϕ and ω; various notions of how ϕ sanctions the presence of ω give rise to different notions of abductive inference such as the set-cover-based approach, logic-based approaches, and the knowledge-level approach (see [12] for a survey). This paper focuses on logic-based abduction over $\mathcal{EL}^+$ TBoxes, however all results except the algorithm presented in Sect. 3 carry over to other logic-based representation schemes straightforwardly.

Due to its hypothetical nature, an abduction problem typically does not have a single solution but a collection of alternative answers $A_1, A_2, \ldots, A_k$ among which optimal solutions are selected by means of a preference order $\preceq$. We denote

A_i being not worse than A_j by $A_i \preceq A_j$, indifference ($A_i \preceq A_j \land A_j \preceq A_i$) is abbreviated by $A_i \simeq A_j$, and strict preference ($A_i \preceq A_j \land A_i \not\simeq A_j$) by $A_i \prec A_j$. Then a (normal) preferential abduction problem can be defined as follows:

Definition 1 (Preferential abduction problem $\mathcal{PAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}})$). *Given a set of axioms $\mathcal{T}$ called the theory, a set of abducible axioms $\mathcal{A}$, a set $\mathcal{O}$ of axioms representing observations such that $\mathcal{T} \not\models \mathcal{O}$, and a (not necessarily total) order relation $\preceq_{\mathcal{A}} \subseteq \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$, determine all $\preceq_{\mathcal{A}}$-minimal sets $A \subseteq \mathcal{A}$ such that $\mathcal{T} \cup A$ is consistent and $\mathcal{T} \cup A \models \mathcal{O}$.*

Typical preference orders over sets include subset-minimality ($A_i \preceq^{\mathrm{s}} A_j \leftrightarrow A_i \subseteq A_j$), minimum cardinality ($A_i \preceq^{\mathrm{c}} A_j \leftrightarrow |A_i| \leq |A_j|$), and weighting-based orders defined by a function w that assigns numerical weights to subsets of $\mathcal{A}$ ($A_i \preceq^{\mathrm{w}} A_j \leftrightarrow w(A_i) \leq w(A_j)$). The first two orders prefer a set A over any of its supersets, this monotonicity property is formalized in Def. 2.

Definition 2 (Monotone and anti-monotone order). *An order $\preceq$ ($\prec$) over sets is monotone (strictly monotone) for set inclusion if and only if $S' \subseteq S$ implies $S' \preceq S$ ($S' \subset S$ implies $S' \prec S$). Conversely, $\preceq$ ($\prec$) is anti-monotone (strictly anti-monotone) for set inclusion if and only if $S' \supseteq S$ implies $S' \preceq S$ ($S' \supset S$ implies $S' \prec S$).*

Applications of abductive information interpretation using a formal domain model include media interpretation [4] and diagnostics for complex technical systems such as production machinery [9]. These domains are characterized by an abundance of low-level observations due to a large number of sensors whereas the model is often unelaborate or incomplete. The next example illustrates how the classical definition of abduction may fail to handle such situations adequately.

Example 1 (Sensitivity to spurious information). Consider the diagnostic unit of a production system whose model states that a fluctuating power supply manifests by intermittent outages of the main control unit while the communication links remain functional and the mechanical gripper of the production system is unaffected (the observations entailed by the diagnosis). Assume a new vibration sensor additionally observes low-frequency vibrations of the system. If the diagnostic model has not been extended yet to encompass these observations, the additional data will in fact distract the diagnostic process and invalidate the diagnosis concerning the power supply, although it might be completely unrelated.

This flaw rests on the requirement that every single observation $o_i \in \mathcal{O}$ be entailed by an admissible solution. It severely restricts the practical applicability of logic-based abduction to real-world industrial applications where an ever-growing amount of sensor data almost inevitably generates pieces of information that the model cannot account for. We therefore extend logic-based abduction in Sect. 2 to handle such cases in a more flexible yet formally sound way, and propose a method to solve such extended abduction problems expressed in the description logic $\mathcal{EL}^+$ in Sect. 3. Section 4 contrasts our proposal with relevant related work on logics and abduction, and we conclude in Sect. 5.

2 Relaxed Abduction

While for very simple models it is possible to identify and remove spurious information in a preprocessing step, this is not feasible for reasonably complex models since the (ir-)relevance of a piece of information depends on the interpretation and is thusly not known beforehand. We therefore propose a general approach based on the intuition that spurious and missing information are two complementary facets of information imperfection and should thus be treated similarly: In addition to assuming information as needed based on the set of abducibles $\mathcal{A}$, relaxed abduction ignores observations from $\mathcal{O}$ during hypotheses generation if required. This intuition is formalized in the next definition.

Definition 3 (Relaxed abduction problem $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$). *Given a set of axioms $\mathcal{T}$ called the theory, a set of abducible axioms $\mathcal{A}$, a set $\mathcal{O}$ of axioms representing observations such that $\mathcal{T} \not\models \mathcal{O}$, and two (not necessarily total) order relations $\preceq_{\mathcal{A}} \subseteq \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$ and $\preceq_{\mathcal{O}} \subseteq \mathcal{P}(\mathcal{O}) \times \mathcal{P}(\mathcal{O})$, determine all $\preceq$-minimal tuples $(A, O) \in \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{O})$ such that $\mathcal{T} \cup A$ is consistent and $\mathcal{T} \cup A \models O$. The order $\preceq$ is defined based on $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{O}}$ as follows:*

- $(A, O) \simeq (A', O') \leftrightarrow A \simeq_{\mathcal{A}} A' \wedge O \simeq_{\mathcal{O}} O'$
- $(A, O) \prec (A', O') \leftrightarrow (A \preceq_{\mathcal{A}} A' \wedge O \prec_{\mathcal{O}} O') \vee (A \prec_{\mathcal{A}} A' \wedge O \preceq_{\mathcal{O}} O')$
- $(A, O) \preceq (A', O') \leftrightarrow ((A, O) \prec (A', O')) \vee ((A, O) \simeq (A', O'))$

Intuitively, a good solution will have high expressive power regarding the observations while being as non-assumptive as possible, which suggests to chose $\preceq_{\mathcal{A}}$ monotone and $\preceq_{\mathcal{O}}$ anti-monotone for set inclusion, respectively. The following example uses one such combination to solve the problem presented in Ex. 1.

Example 2 (Sensitivity to irrelevant data (cont.)). Using inclusion as order criterion over sets, we let $A \preceq_{\mathcal{A}} A' \leftrightarrow A \subseteq A'$ and $O \preceq_{\mathcal{O}} O' \leftrightarrow O \supseteq O'$. As intended, the resulting order $\preceq$ gives rise to the minimal solution which explains all observations but the vibrations and only requires to assume the diagnosis, namely a fluctuating power supply.

Proposition 1 (Conservativeness). *$A \subseteq \mathcal{A}$ is a solution to the preferential abduction problem $\mathcal{PAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}})$ if and only if $(A, \mathcal{O})$ is a solution to the relaxed abduction problem $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$ for an arbitrary order $\preceq_{\mathcal{O}}$ that is anti-monotone for set inclusion.*

Proof. Assume A solves $\mathcal{PAP}$. Then $\mathcal{T} \cup A$ is consistent, $\mathcal{T} \cup A \models \mathcal{O}$, and A is $\preceq_{\mathcal{A}}$-minimal. As $\preceq_{\mathcal{O}}$ is anti-monotone for set inclusion $\mathcal{O}$ is naturally $\preceq_{\mathcal{O}}$-minimal; $(A, \mathcal{O})$ is therefore $\preceq$-minimal and thus solves $\mathcal{RAP}$.
Conversely if $(A, \mathcal{O})$ solves $\mathcal{RAP}$ then $\mathcal{T} \cup A$ is consistent, $\mathcal{T} \cup A \models \mathcal{O}$, and $(A, \mathcal{O})$ is $\preceq$-minimal. Assume $A' \prec_{\mathcal{A}} A$ s. t. $A' \subseteq \mathcal{A}$, $\mathcal{T} \cup A'$ is consistent, $\mathcal{T} \cup A' \models \mathcal{O}$. Then $(A', \mathcal{O}) \prec (A, \mathcal{O})$, contradicting $\preceq$-minimality of $(A, \mathcal{O})$. □

Conservativeness states that, under natural conditions, relaxed abduction is guaranteed to reproduce all (if any) solutions of the corresponding standard abduction problem. Since $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{O}}$ will typically represent competing optimization objectives, it is convenient to treat relaxed abduction as a bi-criterion optimization problem. $\preceq$-minimal solutions then correspond to Pareto-optimal points in the space of all combinations (A, O) meeting the logical requirements of a solution (consistency and entailment) as shown next.

Proposition 2 (Pareto-optimality of $\mathcal{RAP}$). *Let $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$ be a relaxed abduction problem. (A^*, O^*) is a solution to $\mathcal{RAP}$ if and only if it is a Pareto-optimal element (subject to $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{O}}$) of the candidate space $\{(A, O) \in \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{O}) \mid \mathcal{T} \cup A \models O \wedge \mathcal{T} \cup A \not\models \bot\}$.*

Proof. If (A^*, O^*) solves $\mathcal{RAP}$, then $\mathcal{T} \cup A^*$ is consistent and $\mathcal{T} \cup A^* \models O^*$ holds. (A^*, O^*) is thus an element of the explanation space (ES), furthermore (A^*, O^*) must be $\preceq$-minimal. Now assume (A^*, O^*) is not Pareto-optimal for ES, and let $(A', O') \in ES$ such that (w. l. o. g.) $A' \prec_{\mathcal{A}} A^*$ and $O' \preceq_{\mathcal{O}} O^*$. Then $(A', O') \prec (A^*, O^*)$, contradicting $\preceq$-minimality of (A^*, O^*). Thus, (A^*, O^*) is a Pareto-optimal element of the explanation space.
Analogously, let (A', O') be a Pareto-optimal element of ES. To show that the tuple is $\preceq$-minimal, let (A^*, O^*) be a solution to $\mathcal{RAP}$ such that $(A^*, O^*) \prec (A', O')$. Then w. l. o. g. $A^* \prec_{\mathcal{A}} A'$ and $O^* \preceq_{\mathcal{O}} O'$, contradicting Pareto-optimality of (A', O'). Conclusively, (A', O') must be $\preceq$-minimal and therefore solves $\mathcal{RAP}$. $\square$

The next section presents an approach to solving relaxed abduction for $\mathcal{EL}^+$ that explicitly addresses the bi-criterial nature of the problem.

3 Solving Relaxed Abduction for $\mathcal{EL}^+$

The description logic $\mathcal{EL}^+$ is a member of the $\mathcal{EL}$ family of lightweight DLs for which subsumption can be tested in PTime [1]. $\mathcal{EL}^+$ concept descriptions are defined by $C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$ (for $A \in N_{\mathrm{C}}$, $r \in N_{\mathrm{R}}$ a basic concept / role name); $\mathcal{EL}^+$ axioms are either concept inclusion axioms $C \sqcap D$ or role inclusion axioms $r_1 \circ \cdots \circ r_k \sqsubseteq r$ (C, D concept descriptions, $r, r_1, \ldots, r_k \in N_{\mathrm{R}}, k \geq 1$). Since any $\mathcal{EL}^+$ TBox can be normalized with only a linear increase in size, we can assume w. l. o. g. that all axioms are of one of the following forms **(NF1)** $A_1 \sqsubseteq B$, **(NF2)** $A_1 \sqcap A_2 \sqsubseteq B$, **(NF3)** $A_1 \sqsubseteq \exists r.B$, **(NF4)** $\exists r.A_2 \sqsubseteq B$, **(NF5)** $r_1 \sqsubseteq s$, and **(NF6)** $r_1 \circ r_2 \sqsubseteq s$ (for $A_1, A_2, B \in N_{\mathrm{C}}^{\top} = N_{\mathrm{C}} \cup \{\top\}$ and $r_1, r_2, s \in N_{\mathrm{R}}$). In addition to standard refutation-based tableau reasoning, the $\mathcal{EL}$ family allows for a completion-based reasoning scheme that explicitly derives valid subsumptions using a set of rules in the style of Gentzen's sequent calculus. The rules are depicted in Fig. 1, the graph-structure created by applying them to derive subsumptions provides the basis for our approach as shown in the next subsection.

In contrast to other work such as [3, 5] where observations and abducibles are represented by means of named concepts, we assume that both $\mathcal{A}$ and $\mathcal{O}$ are

$$\textsf{(CR1)}\ \frac{A \sqsubseteq A_1}{A \sqsubseteq B}\ [A_1 \sqsubseteq B \in \mathcal{T}]$$

$$\textsf{(CR2)}\ \frac{A \sqsubseteq A_1 \qquad A \sqsubseteq A_2}{A \sqsubseteq B}\ [A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}]$$

$$\textsf{(CR3)}\ \frac{A \sqsubseteq A_1}{A \sqsubseteq \exists r.B}\ [A_1 \sqsubseteq \exists r.B \in \mathcal{T}]$$

$$\textsf{(CR4)}\ \frac{A \sqsubseteq \exists r.A_1 \qquad A_1 \sqsubseteq A_2}{A \sqsubseteq B}\ [\exists r.A_2 \sqsubseteq B \in \mathcal{T}]$$

$$\textsf{(CR5)}\ \frac{A \sqsubseteq \exists r_1.B}{A \sqsubseteq \exists s.B}\ [r_1 \sqsubseteq s \in \mathcal{T}]$$

$$\textsf{(CR6)}\ \frac{A \sqsubseteq \exists r_1.A_1 \qquad A_1 \sqsubseteq \exists r_2.B}{A \sqsubseteq \exists s.B}\ [r_1 \circ r_2 \sqsubseteq s \in \mathcal{T}]$$

$$\textsf{(IR1)}\ \frac{}{A \sqsubseteq A} \qquad \textsf{(IR2)}\ \frac{}{A \sqsubseteq \top}$$

Fig. 1. Completion rules for $\mathcal{EL}^+$

sets of DL axioms just like $\mathcal{T}$. In our experience the axiom-oriented representation provides greater flexibility and information reuse as well as being easier to understand for non-expert users; we furthermore conjecture without formal proof that the concept-based definition is subsumed by the axiom-based one.[3]

3.1 From Completion Rules to Hypergraphs

Since the rules shown in Fig. 1 constitute a sound and complete proof system for $\mathcal{EL}^+$, any normalized axiom set can be represented equivalently as a hypergraph whose vertices are all axioms of type **(NF1)** and **(NF3)** over the concept and role names used in the axiom set (corresponding to all statements admissible as premise or conclusion in a derivation step). The hyperedges are induced by instantiations of the rules **(CR1)**-**(CR6)**; for example an instantiation of **(CR4)** that derives $C \sqsubseteq F$ from $C \sqsubseteq \exists r.D$ and $D \sqsubseteq E$ using the axiom $\exists r.E \sqsubseteq F$ induces a hyperedge $e = (T(e), h(e), w(e))$ with $T(e) = \{C \sqsubseteq \exists r.D, D \sqsubseteq E\}$, $h(e) = C \sqsubseteq F$, and $w(e) = \exists r.E \sqsubseteq F$.

This correspondence can be extended to relaxed abduction problems as follows: Both $\mathcal{T}$ and $\mathcal{A}$ contain arbitrary $\mathcal{EL}^+$ normal form axioms that can justify

[3] First observe that $\mathcal{T} \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq O$ as required in [3] straightforwardly implies $\{\top \sqsubseteq A_1, \ldots, \top \sqsubseteq A_k\} \cup \mathcal{T} \models \top \sqsubseteq O$, i.e. a special case of our definition. Concept abduction and contraction introduced in [5] can conceptually be seen as abduction problems in the line of [3] with additional limitations on the solution A (namely $A = \{C, H\}$ in the former and $A = \{K, D\}$ in the latter case).

single derivation steps represented by a hyperedge (to simplify presentation we assume w. l. o. g. that $\mathcal{A} \cap \mathcal{O} = \emptyset$). Elements from $\mathcal{O}$ on the other hand represent information to be justified (i. e. derived), they therefore correspond to vertices of the hypergraph. This leads to the requirement that axioms in $\mathcal{O}$ may be of type **(NF1)** and **(NF3)** only – this restriction is however negligible in practice since **(NF2)**- and **(NF4)**-axioms can be translated into a **(NF1)**-axiom by introducing a new concept name, and role inclusion axioms are not required for expressing observations about domain objects. To keep track of required assumptions and explained observations, the hyperedges are labelled according to these criteria. This intuition is formalized in the next definition.

Definition 4 (Induced hypergraph $H_{\mathcal{RAP}}$). *Let $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$ be a relaxed abduction problem. The weighted hypergraph $H_{\mathcal{RAP}} = (V, E)$ induced by $\mathcal{RAP}$ is defined by $V = \{(A \sqsubseteq B), (A \sqsubseteq \exists r.B) \mid A, B \in N_{\mathrm{C}}^{\top}, r \in N_{\mathrm{R}}\}$ where $V_{\top} = \{(A, A), (A, \top) \mid A \in N_{\mathrm{C}}^{\top}\} \subseteq V$ denotes the set of terminal states, and E the set of all hyperedges $e = (T(e), h(e), w(e))$ s. t. there is an axiom $ax \in \mathcal{T} \cup \mathcal{A}$ justifying the derivation of $h(e) \in V$ from $T(e) \subseteq V$ due to one of* **(CR1)**-**(CR6)**. *The edge weight $w(e) = (A, O)$ is defined by*

$$A = \begin{cases} \{ax\} & \textit{if } ax \in \mathcal{A}, \\ \emptyset & \textit{otherwise} \end{cases}, \; O = \begin{cases} \{h(e)\} & \textit{if } h(e) \in \mathcal{O}, \\ \emptyset & \textit{otherwise} \end{cases}.$$

Note that the size of $H_{\mathcal{RAP}}$ is bounded polynomially in $|N_{\mathrm{C}}|$ and $|N_{\mathrm{R}}|$. Checking whether a concept inclusion $D \sqsubseteq E$ ($C \sqsubseteq \exists r.D$) is derivable corresponds to checking if in the graph there exists a hyperpath from $V_{\top}$ to the vertex $D \sqsubseteq E$ ($C \sqsubseteq \exists r.D$). Intuitively, there is a hyperpath from X to t if there is a hyperedge connecting some set of nodes Y to t, and each $y_i \in Y$ is reachable from X via a hyperpath; Def. 5 formalizes this intuitive picture.

Definition 5 (Hyperpath). *$p_{X,t} = (V_{X,t}, E_{X,t})$ is a hyperpath in $H = (V, E)$ from X to t if and only if (i) $t \in X$ and $p_{X,t} = (\{t\}, \emptyset)$, or (ii) there is an edge $e \in E$ such that $h(e) = t, T(e) = \{y_1, \ldots, y_k\}$, p_{X,y_i} are hyperpaths from X to y_i, $V \supseteq V_{X,t} = \{t\} \cup \bigcup_{y_i \in T(e)} V_{X,y_i}$, and $E \supseteq E_{X,t} = \{e\} \cup \bigcup_{y_i \in T(e)} E_{X,y_i}$.*

3.2 Hyperpath Search for Relaxed Abduction

This section presents an algorithm for solving a relaxed abduction problem $\mathcal{RAP}$ by determining bi-criterion shortest hyperpaths. The graph algorithm extends a label-correcting algorithm for finding bi-criterion shortest paths in graphs, which is one of the most efficient algorithms known for this problem [14]. It compactly represents the graph using two lists S and R as proposed in [1], the entries are however extended with labels encoding the Pareto-optimal paths to the vertex found so far, and changes are propagated along the weighted edges using two operators called `meet` ($\otimes$) and `join` ($\oplus$). When saturation has terminated, the labels of all $\preceq$-minimal paths in $H_{\mathcal{RAP}}$ are collected in the set $MP(H_{\mathcal{RAP}}) := \bigcup_{v \in V} label(v)$. Algorithm 1 depicts the label propagation algorithm restricted to rule **(CR4)** only due to space limitations. Note that while the order of propagations is irrelevant for correctness, it may have a significant effect on the number of candidates generated: Finding near-optimal solutions early leads to many suboptimal solutions being dominated and therefore not propagated further. As a heuristic to improve performance, we therefore suggest to exhaustively apply $\mathcal{T}$-propagations first, and introduce assumptions only if no other propagation is possible.

Algorithm 1: Label correcting construction of $H_{\mathcal{RAP}}$

```
Data   : $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$, a relaxed abduction problem over $N_C^\top$ and
         $N_R$.
Result : $H_{\mathcal{RAP}}$, the induced hypergraph.
// initialization
foreach $r \in N_R$ do
    R $(r) \leftarrow \emptyset$;
foreach $C \in N_C^\top$ do
    S $(C) \leftarrow \{\ \top\colon \{(\emptyset,\emptyset)\}, C : \{(\emptyset,\emptyset)\}\}$;
// propagation
repeat
    changed $\leftarrow$ false;
    foreach $ax \in \mathcal{T} \cup \mathcal{A}$ do
        else if $ax = \exists r.A_2 \sqsubseteq B$ then // CR4
            foreach $A_1 \in N_C^\top$ s. t. S$(A_1) \ni A_2 : L_{A_1,A_2}$ do
                foreach $A \in N_C^\top$ s. t. R$(r) \ni (A, A_1) : L_{A,r,A_1}$ do
                    $L \leftarrow \emptyset$;
                    if S$(A) \ni B : L_{A,B}$ then $L \leftarrow L_{A,B}$;
                    $L^* \leftarrow$ join($L$, meet($L_{A_1,A_2}$, $L_{A,r,A_1}$, $ax$, $A \sqsubseteq B$));
                    if $L^* \neq L$ then
                        S$(A) \leftarrow ($S$(A) \setminus \{B : L_{A,B}\}) \cup \{B : L^*\}$;
                        changed $\leftarrow$ true;
until changed = false;
```

Proposition 3 (Correctness). *The set of all solutions to a relaxed abduction problem $\mathcal{RAP} = (\mathcal{T}, \mathcal{A}, \mathcal{O}, \preceq_{\mathcal{A}}, \preceq_{\mathcal{O}})$ is given by the $\preceq$-minimal closure of MP($H_{\mathcal{RAP}}$) under component-wise union $(A, O) \uplus (A', O') := (A \cup A', O \cup O')$.*

Proof. Due to space limitations we can only present an outline of the proof here. Following the argumentation in [13, 8], it is clear that hyperpaths in $H_{\mathcal{RAP}}$ starting in $V_\top$ do indeed represent derivations, and that labels constructed from the hyperpaths can be used to encode relevant pieces of information used during that derivation. By Prop. 2, it then suffices to show that the proposed algorithm correctly determines the labels of all Pareto-optimal paths in $H_{\mathcal{RAP}}$ starting in

```
Function meet($L_1$, $L_2$, $just$, $concl$)
    Input  : $L_1$, $L_2$, two label sets; $just$, $concl$, two normal form axioms.
    Output : The label set produced by the meet-operator $\otimes$.
1 result $\leftarrow \{(A_1 \cup A_2, O_1 \cup O_2) \mid (A_1, O_1) \in L_1, (A_2, O_2) \in L_2\}$;
2 if $just \in \mathcal{A}$ then result $\leftarrow \{(A \cup \{just\}, O) \mid (A, O) \in$ result$\}$;
3 if $concl \in \mathcal{O}$ then result $\leftarrow \{(A, O \cup \{concl\}) \mid (A, O) \in$ result$\}$;
4 return result;
```

```
Function join($L_1$, $L_2$)
    Input  : $L_1$, $L_2$, two label sets.
    Output : The label set produced by the join-operator $\oplus$.
1 result $\leftarrow L_1 \cup L_2$;
2 result $\leftarrow$ remove-dominated(result, $\preceq_{\mathcal{A}}$, $\preceq_{\mathcal{O}}$);
3 return result;
```

$V_\top$. This can be proven inductively based on the correctness of the operators $\oplus$ and $\otimes$, which can easily be established in a case-by-case analysis. The terminal closure of $\bigcup_{v \in V} label(v)$ under component-wise union is based on the intuition that, having proved two statements a and b, we can obviously prove $a \wedge b$ by joining the two proofs (corresponding to the $\otimes$ operator). Graphically, this can be seen as adding a dedicated vertex $\top$ such that any other $v \in V$ is connected to $\top$ by a hyperedge $(\{v\}, \top, \{\emptyset, \emptyset\})$, and determining the label of this node that intuitively represents anything that can be derived at all. $\square$

Since the node labels may grow exponentially in the size of $\mathcal{A}$ and $\mathcal{O}$ for general preference orders such as set inclusion, it is worthwhile investigating the benefit of our method as compared to the following simple brute-force approach: Iterating over all pairs $(A, O) \in \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{O})$, collect all (A, O) such that $\mathcal{T} \cup A \models O$ holds and finally drop all $\preceq$-dominated tuples among them. This approach obviously requires $2^{|\mathcal{A}|+|\mathcal{O}|}$ entailment tests, each set passing this test is consequently tested for $\preceq$-minimality. We argue that the our approach is superior to the brute-force method due to three aspects:

1. In contrast to the uninformed search outlined above, the approach proposed in this paper realizes an informed search as it does not generate all possible (A, O)-pairs haphazardly but only those for which the property $\mathcal{T} \cup A \models O$ actually holds, without requiring any additional entailment tests. The net effect of this property depends on the model $\mathcal{T}$ as well as on $\mathcal{A}$ and $\mathcal{O}$; problems having only few solutions at all will obviously benefit most.
2. Dropping $\preceq$-dominated labels for $\preceq_{\mathcal{O}}$ and $\preceq_{\mathcal{A}}$ being (anti-)monotone for set inclusion reduces the worst-case size of node labels from by at least a factor of $O(\sqrt{|\mathcal{A}| \cdot |\mathcal{O}|})$. This can be justified as follows: Fixing a set $A^* \subseteq \mathcal{A}$, the sets $O_i \subseteq \mathcal{O}$ that constitute the (non-dominated) label entries (A^*, O_i) must form an antichain w. r. t. set inclusion. The maximum size of such an antichain is

given by $\binom{|\mathcal{O}|}{\lfloor|\mathcal{O}|/2\rfloor}$ according to Sperner's theorem [15], and can be bounded by $2^{|\mathcal{O}|}/\sqrt{\pi/2 \cdot |\mathcal{O}|}$ using Stirling's approximation.[4] An analogous argument holds for fixed O^*; the size of the cross product can therefore be bounded by $O((2^{|\mathcal{A}|}/\sqrt{|\mathcal{A}|}) \cdot (2^{|\mathcal{O}|}/\sqrt{|\mathcal{O}|}))$, resulting in the factor stated above.

3. In addition to the strict upper bound to the size of labels provided by the preceding line of argumentation, we can also determine the expected number of non-dominated paths to a state as follows: We assume two arbitrary orders over the elements of $\mathcal{A}$ and $\mathcal{O}$ such that any subset can be encoded straightforwardly as a binary vector of length $|\mathcal{A}|$ (resp. $|\mathcal{O}|$). Fixing $A^* \subseteq \mathcal{A}$, an estimate of the expected number of label entries (A^*, O_i) is given by the expected number $A(n,l)$ of maximal $(0,1)$-vectors of length $l = |\mathcal{O}|$ among a set of k distinct such vectors chosen uniformly at random. For our estimation, we let $k := 2^{|\mathcal{O}|}$ to get an upper bound though the actual number is expected to be less (c.f. aspect 1). Adapting the technique used in [2], $A(n,l)$ can be expressed by the recurrence $A(n,l) \leq \lceil\frac{n}{2}\rceil \cdot \frac{A(n,l-1)}{n} + \lceil\frac{n}{2}\rceil \cdot \frac{A(\lceil n/2\rceil,l-1)}{\lceil n/2\rceil} \approx \frac{1}{2} \cdot A(n,l-1) + A(n/2,l-1)$.[5] Assuming $n \geq 2^{l-1}$, the recursion is limited only by l and terminates with the terms $A(n,1) = A(n^{1/(l-1)},1) = 1$ at depth $l-1$. An upper bound is thus given by $A(n,l) \leq A'(l) = A'(l-1) + \frac{1}{2} \cdot A(l-1) = \frac{3}{2} \cdot A(l-1) = (\frac{3}{2})^{l-1}$; the expected label size is thus $O(1.5^{|\mathcal{A}|+|\mathcal{O}|})$.

Other choices for $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{O}}$ can lead to more substantial savings; since the preference orders are used as a pruning criterion during solution generation this may however turn the approach into an approximate one. For instance if the assumption and observation sets are not compared by set inclusion but by cardinality, the maximum label size is reduced to $|\mathcal{A}| \cdot |\mathcal{O}|$ – dependent on the order of rule application the algorithm may however fail to find the optimal solutions. In a more complex setting, assigning numerical weights to observations and abducibles allows to drop only solutions that are significantly worse than others, or to compute bounds on the maximum score a partial solution may still achieve, and use this value as a pruning criterion.

[4] For $m \to \infty$ it holds that $\binom{2m}{m} \sim \frac{4^m}{\sqrt{\pi \cdot m}}$. Letting $m := \lfloor\frac{n}{2}\rfloor$, this yields the estimate $\binom{n}{\lfloor\frac{n}{2}\rfloor} \sim \frac{4^{\lfloor\frac{n}{2}\rfloor}}{\sqrt{\pi \cdot \lfloor\frac{n}{2}\rfloor}} \approx \frac{4^{\frac{n}{2}}}{\sqrt{\pi \cdot \frac{n}{2}}} = \frac{2^n}{\sqrt{\frac{\pi}{2} \cdot n}}$.

[5] This recurrence can be understood as follows: Assume the vectors are arranged in a $(n \times l)$-matrix, sorted by the first component. A randomly chosen vector $\boldsymbol{v}$ starts with 1 or 0 with probability 0.5 each. In the former case, $\boldsymbol{v}$ cannot be dominated by any vector starting with a 0, i. e. the "lower half" of the table is ruled out instantly, and its probability of being dominated by another vector starting with 1 is given by the expected number of maxima among the remaining $\lceil n/2\rceil$ vectors divided by their number, taken together $\boldsymbol{v}$ is maximal with probability $A(\lceil n/2\rceil, l-1)/\lceil n/2\rceil$. If $\boldsymbol{v}$ starts with 0, we can similarly determine its probability of being maximal to be $A(n, l-1)/n$. Summing up these probabilities and and multiplying the result by the number n of original vectors yields the expected number of maxima given above.

4 Related Work

While abductive reasoning naturally addresses the problem of missing observations, there are to the authors' best knowledge no other approaches providing a formally sound solution to logic-based abduction with incomplete models.

The idea of considering abduction as a multi-criteria optimization problem is also central to [10], where multi-criteria decision making techniques are employed to red-cell antibody identification in blood samples. The task is solved using domain-specific operators for combining entries in tables representing the hypotheses. Being an instance of the set-cover approach to abduction, the proposed method does however not address the problem of hypotheses generation, and requires a simple tabular mapping from hypotheses to effects. In the context of abductive (or diagnostic) inference in Bayesian networks, [11] distinguishes between most informative and most simple explanations which correspond to the $\preceq_{\mathcal{O}}$-minimal and the $\preceq_{\mathcal{A}}$-minimal solution in our approach, respectively. However, intermediary Pareto-optimal combinations are not considered in their approach which is furthermore limited to propositional Bayes nets. The algorithm presented in [4] for ABox abduction resembles our approach as it determines alternative explanation sets with varying expressive power, keeping track of the assumptions required for each of them. Unlike the approach presented in this paper, the work by Castano et al. requires special handcrafted models combining forward- and backward-chaining rules, and uses an iterative approach to handle models expressed in the more expressive description logic $\mathcal{ALCQ}$.

[13, 8] use an automaton which is structurally similar to the hypergraph $H_{\mathcal{RAP}}$ introduced in Def. 4 to generate a formula encoding all solutions to a pinpointing respectively a (standard) abduction problem. In contrast to our approach these works guarantee polynomial runtime for solution generation, they do however impose strong restrictions on the combination function, and are inherently limited to uni-criterion problems. Assumption-based Truth Maintenance Systems (ATMSs) [6] impose fewer restrictions on edge weights as compared to the previously mentioned approaches, and similarly to our approach labels containing information on required assumptions are propagated between vertices in a hypergraph structure. We are however not aware of any extension to ATMSs allowing for a tradeoff between assumptions and explanatory power, nor do ATMSs consider any order over labels other than implication.

5 Conclusions and Outlook

We have introduced relaxed abduction, a novel non-standard reasoning task for description logics. Relaxed abduction extends logic-based abduction to a general and formally sound framework for interpreting spurious information w. r. t. incomplete models. We have presented an algorithm for relaxed abduction over $\mathcal{EL}^+$ knowledge bases based on the notion of Pareto-optimal hyperpaths in the derivation graph, and motivated its superiority to a straightforward enumeration approach despite the inherent exponential growth of node labels. The proposed algorithm is straightforwardly extensible to other DLs for which subsumption can be decided by completion such as $\mathcal{EL}^{++}$ which supports nominals and thus ABox abduction. The very general notion of relaxed abduction allows for several interesting specializations resulting from different choices for $\preceq_{\mathcal{A}}$ and $\preceq_{\mathcal{O}}$: Approximate solutions can for example be generated very efficiently (i. e. with linear label size) if we use set cardinality as a dominance criterion. More elaborate schemes based on weights assigned to the axioms allow for early and even lossless pruning of suboptimal partial solutions while also reducing label sizes.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. pp. 364–369 (2005)
2. Bentley, J.L., Kung, H.T., Schkolnick, M., Thompson, C.D.: On the average number of maxima in a set of vectors and applications. Journal of the ACM 25(4) (1978)
3. Bienvenu, M.: Complexity of abduction in the EL family of lightweight description logics. In: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning. pp. 220–230 (2008)
4. Castano, S., Espinosa Peraldi, I.S., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., Wessel, M.: Multimedia interpretation for dynamic ontology evolution. Journal of Logic and Computation 19(5), 859–897 (2009)
5. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In: Proceedings of the 16th International Workshop on Description Logics (2003)
6. De Kleer, J.: An assumption-based TMS. Artificial Intelligence 28(2), 127–162 (1986)
7. Hartshorne, C., Weiss, P. (eds.): Collected Papers of Charles Sanders Peirce. Harvard University Press, 1st edn. (1931)
8. Hubauer, T.M., Lamparter, S., Pirker, M.: Automata-based abduction for tractable diagnosis. In: Proceedings of the DL Home 23rd International Workshop on Description Logics. pp. 360–371 (2010)
9. Hubauer, T.M., Legat, C., Seitz, C.: Empowering adaptive manufacturing with interactive diagnostics: A multi-agent approach. In: Proceedings of the 9th International Conference on Practical Applications of Agents and Multi-Agent Systems. pp. 47–56 (2011)
10. Iyer, N.S., Josephson, J.R.: Multicriterially best explanations. In: Proceedings of the 4th International Conference on Discovery Science. pp. 128–140 (2001)
11. Kwisthout, J.: Two new notions of abduction in bayesian networks. In: Proceedings of the 22nd Benelux Conference on Artificial Intelligence (2010)
12. Paul, G.: Approaches to abductive reasoning: An overview. Artificial Intelligence Review 7(2), 109–152 (1993)
13. Peñaloza, R.: Using tableaux and automata for pinpointing in EL. In: Proceedings of the TABLEAUX 2009 Wokshop on Tableaux versus Automata as Logical Decision Methods (2009)
14. Skriver, A.J.V.: A classification of bicriterion shortest path (bsp) algorithms. Asia-Pacific Journal of Operational Research 17, 199–212 (2000)
15. Sperner, E.: Ein Satz über Untermengen einer endlichen Menge. Mathematische Zeitschrift 27, 544–548 (1928)

The Complexity of Probabilistic $\mathcal{EL}$

Víctor Gutiérrez Basulto[1], Jean Christoph Jung[1], Carsten Lutz[1], and Lutz Schröder[1,2]

[1] University Bremen, Germany, {victor,jeanjung,clu}@informatik.uni-bremen.de
[2] DFKI Bremen, Germany, lutz.schroeder@dfki.de

Abstract. We analyze the complexity of subsumption in probabilistic variants of the description logic $\mathcal{EL}$. In the case where probabilities apply only to concepts, we map out the borderline between tractability and EXPTIME-completeness. One outcome is that *any* probability value except zero and one leads to intractability in the presence of general TBoxes, while this is not the case for classical TBoxes. In the case where probabilities can also be applied to roles, we show PSPACE-completeness. This result is (positively) surprising as the best previously known upper bound was 2-EXPTIME and there were reasons to believe in completeness for this class.

1 Introduction

The fact that traditional description logics (DLs) do not provide any built-in means for representing uncertainty has led to various proposals for probabilistic extensions, see for example [14, 10, 5, 13, 12] and references therein. Recently, a new family of probabilistic DLs was introduced in [15], with the distinguishing feature that its members relate to the well-established probabilistic first-order logic (FOL) of Halpern and Bacchus [7, 4] in the same way as classical DLs relate to traditional FOL. The main purpose of DLs from the new family, from now on called *Prob-DLs*, is to enable concept definitions that require reference to (degrees of) possibility, likelihood, and certainty. To this effect, Prob-DLs provide a probabilistic constructor $P_{\sim p}$ with $\sim \in \{<, \leq, =, \geq, >\}$ and $p \in [0, 1]$ that can be applied to concepts and sometimes also to roles. For example,

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Disease} \sqcap P_{>0.25}\mathsf{Infectious})$$

describes patients having a disease that is infectious with probability at least $.25$. It was argued in [15] that Prob-DLs are well-suited to capture aspects of uncertainty in biomedical ontologies such as SNOMED CT. Since such ontologies are often formulated in DLs from the $\mathcal{EL}$ family for which subsumption can be solved in polynomial time [2, 16], probabilistic extensions of $\mathcal{EL}$ in the style of Prob-DLs is particularly relevant in this context. Some initial results have already been obtained in [15].

In this paper, we establish a rather complete picture of the complexity of subsumption in Prob-DLs based on $\mathcal{EL}$. In the first part, we consider *Prob-$\mathcal{EL}$* in which probabilities can only be applied to concepts, but not to roles. In [15], it was shown that some concrete combinations of probability constructors such as $P_{>0}$ and $P_{>0.4}$ lead to intractability (in fact, EXPTIME-completeness) of subsumption while a restriction to the probability values zero and one does not. Here, we prove the much more general result that the extension of $\mathcal{EL}$ with *any* single concept constructor $P_{\sim p}$, where

$\sim \in \{<, \leq, =, \geq, >\}$ and $p \in (0, 1)$, results in ExpTime-completeness. More specifically, this result applies to *general TBoxes*, i.e., to sets of concept inclusions $C \sqsubseteq D$ when $\sim \in \{=, \geq, >\}$ and even to the empty TBox when $\sim \in \{<, \leq, \}$. Inspired by the observation that many biomedical ontologies such as Snomed CT are *classical TBoxes*, i.e., sets of concept definitions $A \equiv D$ with atomic and unique left-hand sides, we then show that probabilities other than zero and one *can* be used without losing tractability in (possibly cyclic) classical TBoxes for the cases $\sim \in \{>, \geq\}$. More precisely, subsumption in Prob-$\mathcal{EL}$ is tractable when only the constructors $P_{\sim p}$ and $P_{=1}$ are admitted, for any (single!) choice of $\sim \in \{\geq , >\}$ and $p \in (0, 1)$. The resulting logic actually 'coincides' for all possible choices. We also show that when a second probability value from the range $(0, 1)$ sufficiently 'far away' from p is added, the complexity of subsumption snaps back to ExpTime-completeness.

In the second part of the paper, we consider *Prob-$\mathcal{EL}_r$*, where probabilities can be applied to both concepts and roles, concentrating on general TBoxes. While decidability is an open problem for full Prob-$\mathcal{EL}_r$, it was known that subsumption is in 2-ExpTime and PSpace-hard in *Prob-$\mathcal{EL}_r^{>0;=1}$*, where probability values are restricted to zero and one. Since subsumption in the $\mathcal{ALC}$-version of Prob-$\mathcal{EL}_r^{>0;=1}$ is 2-ExpTime-complete and the complexity of the $\mathcal{EL}$-version and the $\mathcal{ALC}$-version of many-dimensional DLs (such as Prob-DLs) coincides in all known cases, it was thus tempting to conjecture 2-ExpTime-completeness also of subsumption in Prob-$\mathcal{EL}_r^{>0;=1}$. We show that this is not the case by establishing a tight PSpace upper bound for subsumption in Prob-$\mathcal{EL}_r^{>0;=1}$. This also implies PSpace-completeness for the two-dimensional DL $\mathsf{S5}_{\mathcal{EL}}$, in sharp contrast with the 2-ExpTime-completeness of $\mathsf{S5}_{\mathcal{ALC}}$.

This paper is a workshop version of [6]. Proofs can be found in the long version of that paper, to be found at http://www.informatik.uni-bremen.de/˜clu/papers/.

2 Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of concept names and role names. *Prob-$\mathcal{EL}$* is the extension of $\mathcal{EL}$ that allows the application of probabilities to concepts, i.e., Prob-$\mathcal{EL}$ concepts are built according to the rule

$$C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.C \mid P_{\sim p}C$$

where A ranges over $\mathsf{N_C}$, r over $\mathsf{N_R}$, $\sim$ over $\{<, \leq, =, \geq, >\}$, and $p \in [0, 1]$. The concept $P_{\sim p}C$ denotes the class of objects that are an instance of C with probability $\sim p$. For example, the Snomed CT concept 'animal bite by potentially rabid animal' can be expressed as

$$\mathsf{Bite} \sqcap \exists \mathsf{by}.(\mathsf{Animal} \sqcap P_{>0.5}\exists \mathsf{has.Rabies}).$$

When we admit only a few values for $\sim$ and n, we put them in superscript; for example, Prob-$\mathcal{EL}^{>0.4,<0.1}$ denotes the extension of $\mathcal{EL}$ with $P_{>0.4}C$ and $P_{<0.1}C$. Probabilities can be applied to roles using the concept constructors $\exists P_{\sim p}r.C$ where $\sim$ and p range over the same values as above, expressing that there is an element satisfying C that is related to the current element by the role name r with probability $\sim p$. For example, the SNOMED CT concept 'disease of possible viral origin' can be modeled as

$$\mathsf{Disease} \sqcap \exists P_{>0}\mathsf{origin}.\mathsf{Viral}.$$

We denote the extension of Prob-$\mathcal{EL}$ with the constructor $\exists P_{\sim p} r.C$ with *Prob-*$\mathcal{EL}_r$. We also consider the restriction *Prob-*$\mathcal{EL}_r^{>0;=1}$ of Prob-$\mathcal{EL}_r$ to probabilities 0 and 1 (both on concepts and roles).

The semantics of the probabilistic DLs is given in terms of a *probabilistic interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu)$, where $\Delta^{\mathcal{I}}$ is the (non-empty) domain, W a non-empty set of *possible worlds*, μ a discrete probability distribution on W, and for each $w \in W$, $\mathcal{I}_w$ is a classical DL interpretation with domain $\Delta^{\mathcal{I}}$. We usually write $C^{\mathcal{I},w}$ for $C^{\mathcal{I}_w}$, and likewise for $r^{\mathcal{I},w}$. For concept names A and role names r, we define the probability

– $p_d^{\mathcal{I}}(A)$ that $d \in \Delta^{\mathcal{I}}$ is an A as $\mu(\{w \in W \mid d \in A^{\mathcal{I},w}\})$;
– $p_{d,e}^{\mathcal{I}}(r)$ that $d, e \in \Delta^{\mathcal{I}}$ are related by r as $\mu(\{w \in W \mid (d,e) \in r^{\mathcal{I},w}\})$.

Next, we extend $p_d^{\mathcal{I}}(A)$ to compound concepts C and define the extension $C^{\mathcal{I},w}$ of compound concepts by mutual recursion on C. The definition of $p_d^{\mathcal{I}}(C)$ is exactly as in the base case, with A replaced by C. The extension of compound concepts is defined as follows:

$$\begin{aligned}
\top^{\mathcal{I},w} &= \Delta^{\mathcal{I}} \quad (C \sqcap D)^{\mathcal{I},w} = C^{\mathcal{I},w} \cap D^{\mathcal{I},w} \\
(\exists r.C)^{\mathcal{I},w} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in r^{\mathcal{I},w} \wedge e \in C^{\mathcal{I},w}\} \\
(P_{\sim p} C)^{\mathcal{I},w} &= \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) \sim p\} \\
(\exists P_{\sim p} r.C)^{\mathcal{I},w} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I},w} : p_{d,e}^{\mathcal{I}}(r) \sim p\}
\end{aligned}$$

A *general TBox* is a finite set of *concept inclusions* $C \sqsubseteq D$, where C, D are concepts. A *classical TBox* is a set of *concept definitions* $A \equiv C$, where A is a concept name and the left-hand sides of concept definitions are unique. Note that cyclic definitions are allowed.

A probabilistic interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I},w} \subseteq D^{\mathcal{I},w}$ and a concept definition $A \equiv C$ if $A^{\mathcal{I},w} = C^{\mathcal{I},w}$, for all $w \in W$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all inclusions/definitions in $\mathcal{T}$. A concept C is *subsumed by a concept D relative to a TBox $\mathcal{T}$* (written $\mathcal{T} \models C \sqsubseteq D$) if every model $\mathcal{I}$ of $\mathcal{T}$ satisfies $C \sqsubseteq D$.

The above definition is the result of transferring the notion of subsumption from standard DLs to probabilistic DLs in a straightforward way. However, there is an alternative variant of subsumption that is natural for probabilistic DLs: a concept C is *positively subsumed by a concept D relative to a TBox $\mathcal{T}$* (written $\mathcal{T} \models^+ C \sqsubseteq D$) if $C^{\mathcal{I},w} \subseteq D^{\mathcal{I},w}$ for every probabilistic model $\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu)$ and every $w \in W$ with $\mu(w) > 0$. Intuitively, classical subsumption is about subsumptions that are *logically implied* whereas positive subsumption is about subsumptions that are *certain*. For example, when $\mathcal{T}_\emptyset$ is the empty TBox, then $\mathcal{T}_\emptyset \not\models P_{=1}A \sqsubseteq A$, but we can only have $d \in (P_{=1}A)^{\mathcal{I},v} \setminus A^{\mathcal{I},v}$ when $\mu(v) = 0$, thus non-subsumption is only witnessed by worlds that we are certain to not be the actual world. Consequently, $\mathcal{T}_\emptyset \models^+ P_{=1}A \sqsubseteq A$.

In the extension Prob-$\mathcal{ALC}$ of Prob-$\mathcal{EL}$ with negation studied in [15], positive subsumption can easily be reduced to subsumption. This does not seem easily possible in Prob-$\mathcal{EL}$. In fact, we will sometimes use (Turing) reductions in the opposite direction.

3 Probabilistic Concepts

In [15], it was shown that subsumption in Prob-$\mathcal{EL}^{>0;=1}$ with general TBoxes is in PTIME, whereas the same problem is EXPTIME-complete in Prob-$\mathcal{EL}^{>0;>0.4}$ (both in the positive and in the unrestricted case). This raises the question whether *any* probability except 0,1 can be admitted in Prob-$\mathcal{EL}$ without losing tractability. The following theorem provides a strong negative result.

Theorem 1. *For all $p \in (0,1)$, (positive) subsumption in Prob-$\mathcal{EL}^{\sim p}$ relative to*

1. *general TBoxes is* EXPTIME-*hard when* $\sim \in \{=,>,\geq\}$
2. *the empty TBox is* EXPTIME-*hard when* $\sim \in \{\leq, <\}$

Matching upper bounds are an immediate consequence of the fact that each logic Prob-$\mathcal{EL}^{\sim p}$ is a fragment of the DL Prob-$\mathcal{ALC}_c$ for which subsumption was proved EXPTIME-complete in [15]. To prove the lower bounds, it suffices to show that each logic Prob-$\mathcal{EL}^{\sim p}$ is *non-convex*, i.e., that there are a general TBox $\mathcal{T}$ and concepts $C, D_1, \dots, D_n$, $n \geq 2$, such that $\mathcal{T} \models C \sqsubseteq D_1 \sqcup \dots \sqcup D_n$, but $\mathcal{T} \not\models C \sqsubseteq D_i$ for all i. Once that this is established, standard proof techniques from [2] can be used to reduce satisfiability in $\mathcal{ALC}$ relative to general TBoxes, which is EXPTIME-complete, to subsumption in Prob-$\mathcal{EL}^{\sim p}$. The following constructions work for standard subsumption and positive subsumption alike.

First consider $\sim = \geq$ and assume $p \leq 0.5$. Fix a $k > 0$ such that $k \cdot p > 1$ and set

$$\begin{aligned} \mathcal{T} &= \{A_i \sqcap A_j \sqsubseteq P_{\geq p} B_{ij} \mid 1 \leq i < j \leq k\} \\ C &= P_{\geq p} A_1 \sqcap \dots \sqcap P_{\geq p} A_k \\ D_{ij} &= P_{\geq p} B_{ij} \end{aligned}$$

Intuitively, the probabilities stipulated by C sum up to > 1, thus some of the A_i have to overlap, but there is a choice as to which ones these are. Formally, we can show non-convexity by proving that $\mathcal{T} \models C \sqsubseteq \bigsqcup_{1 \leq i < j \leq k} D_{ij}$, but $\mathcal{T} \not\models C \sqsubseteq D_{ij}$ for any i, j. The comparisons $\sim \in \{=,>\}$ can be handled similarly.

Now assume that $p > 0.5$. We start with the case $\sim = >$ and use a variation of the above. The main idea is to employ $P_{>p}C$ to simulate $P_{>q}C$, for some $q \leq 0.5$, which brings us back to a case already dealt with. More precisely, let $n > 0$ be smallest such that $n > \frac{1}{2(1-p)}$ and set $q = pn - n + 1$. An easy computation shows that $0 \leq q < 0.5$. Moreover, it can be shown that

$$P_{>p} X_1 \sqcap \dots \sqcap P_{>p} X_n \sqsubseteq P_{>q}(X_1 \sqcap \dots \sqcap X_n)$$

which allows us to redo the above reduction with probability $q < 0.5$. Details are given in the long version of [6]. The comparisons $\sim \in \{=,\geq\}$ can be handled similarly.

For the remaining cases $\sim \in \{<, \leq\}$, there is a very simple argument for non-convexity even w.r.t. the empty TBox: we have $\top \sqsubseteq P_{<p}A \sqcup P_{<p}P_{<p}A$, but neither $\top \sqsubseteq P_{<p}A$ nor $\top \sqsubseteq P_{<p}P_{<p}A$, and likewise when $\sim$ is $\leq$.

When $\sim \in \{=, >, \geq\}$, the proof of Theorem 1 relies on general TBoxes in a crucial way. It turns out that when we restrict ourselves to classical TBoxes, tractability can be attained even with probabilities other than 0 and 1.

Theorem 2. *For all $\sim \in \{>, \geq\}$ and $p \in [0,1]$, (positive) subsumption in Prob-$\mathcal{EL}^{\sim p;=1}$ relative to classical TBoxes is in* PTIME.

To prove Theorem 2, we start with positive subsumption. We can assume $p > 0$ since subsumption in Prob-$\mathcal{EL}^{>0;=1}$ is in PTIME even with general TBoxes. To prove a PTIME upper bound, we use a 'consequence-driven' procedure similar to the ones in [2, 11]. A concept name A is *defined* in a classical TBox $\mathcal{T}$ if there is a concept definition $A \equiv C \in \mathcal{T}$, and *primitive* otherwise. We can w.l.o.g. restrict our attention to the subsumption of *defined concept names* relative to TBoxes. We also assume that the input TBox is normalized to a set of concept definitions of the form

$$A \equiv P_1 \sqcap \cdots \sqcap P_n \sqcap C_1 \sqcap \cdots \sqcap C_m$$

$n, m \geq 0$, and where $P_1, \ldots, P_n$ are primitive concept names and $C_1, \ldots, C_m$ are of the form $P_{\sim p}A$, $P_{=1}A$, and $\exists r.A$ with A a defined concept name (note that the top concept is completely normalized away). It is well-known that such a normalization can be achieved in polytime, see [1] for details. For a given TBox $\mathcal{T}$ and a defined concept name A in $\mathcal{T}$, we write C_A to denote the *defining concept* for A in $\mathcal{T}$, i.e., $A \equiv C_A \in \mathcal{T}$. Moreover, we deliberately confuse the concept $C_A = D_1 \sqcap \cdots \sqcap D_k$ with the set $\{D_1, \ldots, D_k\}$. We define a set of concepts 'certain for C_A' as

$$\mathsf{cert}(C_A) = \{P_*B \mid P_*B \in C_A\} \cup \bigcup_{P_{=1}B \in C_A} \{C_B\}$$

where, here and in what follows, P_* ranges over $P_{=1}$ and $P_{>p}$. Intuitively, $\mathsf{cert}(C_A)$ contains concepts that hold with probability 1 whenever A is satisfied in some world. The algorithm starts with the normalized input TBox and then exhaustively applies the completion rules displayed in Figure 1. As a general proviso, each rule can be applied only if it adds a concept that occurs in $\mathcal{T}$ and actually changes the TBox, e.g., **R1** can only be applied when $\exists r.B'$ occurs in $\mathcal{T}$ and $\exists r.B' \notin C_A$. Exemplarily, we explain rule **R5** in more detail. If all defining concepts C_B of B are certain for A, then $A \sqsubseteq P_{=1}B$, thus we can add $P_{=1}B$ to C_A. Let $\mathcal{T}^*$ be the result of exhaustive rule application and let C^*_A be the defining concept for A in $\mathcal{T}^*$, for all concept names A. The 'only if' direction requires a careful and surprisingly subtle model construction.

Lemma 1. *For all defined concept names A, B, we have $\mathcal{T} \models^+ A \sqsubseteq B$ iff $C^*_B \subseteq C^*_A$.*

It is easy to see that TBox completion requires only polytime: every rule application extends the TBox, but both the number of concept definitions and of conjuncts in each concept definition is bounded by the size of the original TBox.

To prove Theorem 2 for unrestricted subsumption, we provide a Turing reduction from unrestricted subsumption to positive subsumption. We again assume that the input

R1 If $\exists r.B \in C_A$, and $C_{B'} \subseteq C_B$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{\, \exists r.B'\}$

R2 If $P_{=1}B \in C_A$
then replace $A \equiv C_A$ with $A \equiv C_A \cup C_B$

R3 If $P_{=1}B \in C_A$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{P_{\sim p}B\}$

R4 If $P_{\sim p}B \in C_A$, and $D \in \mathsf{cert}(C_B)$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{D\}$

R5 If $C_B \subseteq \mathsf{cert}(C_A)$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{P_{=1}B\}$

R6 If $P_{\sim p}B \in C_A$ and $C_{B'} \subseteq \mathsf{cert}(C_A) \cup C_B$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{P_{\sim p}B'\}$

Fig. 1. TBox completion rules for positive subsumption

TBox is in the described normal form and then exhaustively apply the rules shown in Figure 2, calling the result $\mathcal{T}^*$ with defining concept of the form C_A^*.

Lemma 2. *For all defined concept names A, B, we have $\mathcal{T} \models A \sqsubseteq B$ iff $C_B^* \subseteq C_A^*$.*

Clearly, the Turing reduction and thus the overall algorithm runs in polytime.

It is interesting to note that the proof of Theorem 2 is based on exactly the same algorithm, for all $\sim\, \in \{\geq\,, >\}$ and $p \in (0, 1]$. It follows that there is in fact only *a single logic* Prob-$\mathcal{EL}^{\sim p}$, for all such $\sim$ and p. Formally, given a Prob-$\mathcal{EL}^{\sim p}$-concept C, $\approx\, \in \{\geq\,, >\}$ and $q \in (0, 1]$, let $C_{\approx q}$ denote the result of replacing each subconcept $P_{\sim p}D$ in C with $P_{\approx q}D$ in C and similarly for Prob-$\mathcal{EL}^{\sim p}$-TBoxes $\mathcal{T}$.

Theorem 3. *For any $p, q > 0$, $\sim, \approx\, \in \{>, \geq\}$, $\mathcal{EL}^{\sim p}$-concepts C, D and -TBox $\mathcal{T}$, we have $\mathcal{T} \models^+ C \sqsubseteq D$ iff $\mathcal{T}_{\approx q} \models^+ C_{\approx q} \sqsubseteq D_{\approx q}$, and likewise for unrestricted subsumption.*

Consequently, the (potentially difficult!) choice of a concrete $\sim\, \in \{\geq\,, >\}$ and $p \in (0, 1]$ is moot. In fact, it might be more intuitive to replace the constructor $P_{\sim p}C$ with a constructor $\mathcal{L}\,C$ that describes elements which 'are likely to be a C', and to replace $P_{=1}C$ with the constructor $\mathcal{C}\,C$ to describe elements that 'are certain to be a C', see e.g. [8, 9] for other approaches to logics of likelihood. Note that the case $p = 0$ is different from the cases considered above: for example, we have $\mathcal{T}_\emptyset \models^+ \exists r.A \sqsubseteq \exists r.P_{>p}A$ iff $p = 0$, and likewise $\mathcal{T}_\emptyset \models P_{>p}\exists r.A \sqsubseteq P_{>p}\exists r.P_{>p}A$ iff $p = 0$. In the spirit of the constructors $\mathcal{C}$ and $\mathcal{L}$, $P_{>0}C$ can be replaced with a constructor $\mathcal{P}C$ that describes elements for which 'it is possible that they are a C'. For example, the SNOMED CT concepts 'definite thrombus' and 'possible thrombus' can then be written as $\mathcal{C}\,\mathsf{Thrombus}$ and $\mathcal{P}\,\mathsf{Thrombus}$ (although we speculate that the SNOMED CT designers mean 'likely' rather than 'possible').

S1 If $\exists r.B \in C_A$, and $C_{B'} \subseteq C_B$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{ \exists r.B' \}$

S2 If $\mathcal{T} \models^+ \mathsf{cert}(C_A) \sqsubseteq P_* B$
then replace $A \equiv C_A$ with $A \equiv C_A \cup \{P_* B\}$

Fig. 2. TBox completion rules for Turing reduction

It is a natural question whether the PTime upper bound for classical TBoxes extends to the case of multiple probability values (except one, which is apparently always uncritical). The following result shows that many combinations of two probability values lead to (non-convexity, thus) intractability, even without any TBox.

Theorem 4. *Let $\sim \in \{>, \geq\}$, and $p, q \in [0, 1)$. Then (positive) subsumption in Prob-$\mathcal{EL}^{\sim p;\sim q}$ relative to the empty TBox is* ExpTime*-hard if (i) $q = 0$, (ii) $p \leq 1/2$ and $q < p^2$, or more generally (iii) $2p - 1 < q < p^2$.*

In particular, we cannot combine the constructors $\mathcal{P}$ and $\mathcal{L}$ mentioned above without losing tractability. The above formulation of Theorem 4 is actually only a consequence of a more general (but also more complicated to state) result established in the long version of [6]. We conjecture that (positive) subsumption in Prob-$\mathcal{EL}^{\sim p;\sim q}$ relative to classical TBoxes is in PTime whenever $p \geq q \geq p^2$ and that, otherwise, it is ExpTime-hard.

4 Probabilistic Roles

Adding probabilistic roles to Prob-$\mathcal{EL}$ tends to increase the complexity of subsumption. While for full Prob-$\mathcal{EL}_r$ even decidability is open, it was shown in [15] that subsumption is in 2-ExpTime and PSpace-hard in Prob-$\mathcal{EL}_r^{>0;=1}$. As discussed in the introduction, there were reasons to believe that this problem is actually 2-ExpTime-complete. We show that this is not the case by proving a PSpace upper bound, thus establishing PSpace-completeness. This result holds both for positive and unrestricted subsumption, we start with the positive case.

We again concentrate on subsumption between concept *names* and assume that the input TBox is in a certain normal form, defined as follows. A *basic* concept is a concept of the form $\top$, A, $P_{>0}A$, $P_{=1}A$, or $\exists\alpha.A$, where A is a concept name and, here and in what follows, α is a *role*, i.e., of the form r, $P_{>0}r$, or $P_{=1}r$ with r a role name. Now, every concept inclusion in the input TBox is required to be of the form

$$X_1 \sqcap \cdots \sqcap X_n \sqsubseteq X$$

with $X_1, \ldots, X_n, X$ basic concepts. It is not hard to show that every TBox can be transformed into this normal form in polynomial time such that (non-)subsumption between the concept names that occur in the original TBox is preserved.

Let $\mathcal{T}$ be the input TBox in normal form, CN the set of concept names that occur in $\mathcal{T}$, BC the set of basic concepts in $\mathcal{T}$, and ROL the set of roles in $\mathcal{T}$. Call a role *probabilistic* if it is of the form $P_{=1}r$ or $P_{>0}r$. Our algorithm maintains the following data structures:

- a mapping Q that associates with each $A \in \mathsf{CN}$ a subset $Q(A) \subseteq \mathsf{BC}$ such that $\mathcal{T} \models A \sqsubseteq X$ for all $X \in Q(A)$;
- a mapping Q_{cert} that associates with each $A \in \mathsf{CN}$ a subset $Q_{\mathsf{cert}}(A) \subseteq \mathsf{BC}$ such that $\mathcal{T} \models A \sqsubseteq P_{=1}X$ for all $X \in Q_{\mathsf{cert}}(A)$;
- a mapping R that associates with each probabilistic role $\alpha \in \mathsf{ROL}$ a binary relation $R(\alpha)$ on CN such that $\mathcal{T} \models A \sqsubseteq P_{>0}(\exists\alpha.B)$ for all $(A, B) \in R(\alpha)$.

Some intuition about the data structures is already provided above; e.g., $X \in Q(A)$ means that $\mathcal{T} \models A \sqsubseteq X$. However, there is also another view on these structures that will be important in what follows: they represent an abstract view of a model of $\mathcal{T}$, where each set $Q(A)$ describes the concept memberships of a domain element d in a world w with $d \in A^{\mathcal{I},w}$ and R describes role memberships, i.e., when $(A, B) \in R(\alpha)$, then $d \in A^{\mathcal{I},w}$ implies that in some world v with positive probability, d has an element described by $Q(B)$ as an α-successor. In this context, $Q_{\mathsf{cert}}(A)$ contains all concepts that must be true with probability 1 for any domain element that satisfies A in *some* world. Note that non-probabilistic roles r and probabilistic roles $P_{=1}r$ are not represented in the $R(\cdot)$ data structure; we will treat them in a more implicit way later on.

The data structures are initialized as follows, for all $A \in \mathsf{CN}$ and relevant roles α:

$$Q(A) = \{\top, A\} \qquad Q_{\mathsf{cert}}(A) = \{\top\} \qquad R(\alpha) = \emptyset.$$

The sets $Q(\cdot)$, $Q_{\mathsf{cert}}(\cdot)$, and $R(\cdot)$ are then repeatedly extended by the application of various rules. Before we can introduce these rules, we need some preliminaries. As the first step, Figure 3 presents a (different!) set of rules that serves the purpose of saturating a set of concepts Γ. We use $\mathsf{cl}(\Gamma)$ to denote the set of concepts that is the result of exhaustively applying the displayed rules to Γ, where any rule can only be applied if the added concept is in BC, but not yet in Γ. The rules access the data structure $Q(\cdot)$ introduced above and shall later be applied to the sets $Q(A)$ and $Q_{\mathsf{cert}}(A)$, but they will also serve other purposes as described below. It is not hard to see that rule application terminates after polynomially many steps.

R1 If $X_1 \sqcap \ldots \sqcap X_n \sqsubseteq X \in \mathcal{T}$ and $X_1, \ldots, X_n \in \Gamma$ then add X to Γ

R2 If $P_{=1}A \in \Gamma$ then add A to Γ

R3 If $\exists P_{=1}r.A \in \Gamma$ then add $\exists r.A$ to Γ

R4 If $A \in \Gamma$ then add $P_{>0}A$ to Γ

R5 If $\exists r.A \in \Gamma$ then add $\exists P_{>0}r.A$ to Γ

R6 If $\exists\alpha.A \in \Gamma$ and $B \in Q(A)$ then add $\exists\alpha.B$ to Γ

Fig. 3. Saturation rules for $\mathsf{cl}(\Gamma)$

The rules that are used for completing the data structures $Q(\cdot)$, $Q_{\mathsf{cert}}(\cdot)$, and $R(\cdot)$ are more complex and refer to 'traces' through these data structures.

Definition 1. *Let $B \in \mathsf{CN}$. A* trace to B *is a sequence $S, A_1, \alpha_2, A_2, \ldots, \alpha_n, A_n$ where*

1. *$S = A$ for some $P_{>0}A \in Q(A_1)$ or $S = (r, B)$ for some $(A_1, B) \in R(P_{>0}r)$;*
2. *each $A_i \in \mathsf{CN}$ and each $\alpha_i \in \mathsf{ROL}$ is a probabilistic role, such that $A_n = B$;*
3. *$(A_i, A_{i-1}) \in R(\alpha_i)$ for $1 < i \leq n$.*

If t is a trace of length n, we use t_k, $k \leq n$, to denote the trace $S, A_1, \alpha_2, \ldots, \alpha_k, A_k$. Intuitively, the purpose of a trace is to deal with worlds that are generated by concepts $P_{>0}A$ and $\exists P_{>0}r.A$; there can be infinitely many such worlds as Prob-$\mathcal{EL}_r^{>0;=1}$ lacks the finite model property, see [15]. The trace starts at some domain element represented by a set $Q(A_1)$ in the world generated by the first element S of the trace, then repeatedly follows role edges represented by $R(\cdot)$ backwards until it reaches the final domain element represented by $Q(B)$. The importance of traces stems from the fact that information can be propagated along them, as captured by the following notion.

Definition 2. *Let $t = S, A_1, \alpha_2, \ldots, \alpha_n, A_n$ be a trace of length n. Then the* type *$\Gamma(t) \subseteq \mathsf{BC}$ of t is*

- $\mathsf{cl}(\{A\} \cup Q_{\mathsf{cert}}(A_1))$ *if $n = 1$ and $S = A$;*
- $\mathsf{cl}(Q_{\mathsf{cert}}(A_1) \cup \{\exists\, r.B' \in \mathsf{BC} \mid B' \in Q_{\mathsf{cert}}(B)\})$ *if $n = 1$ and $S = (r, B)$;*
- $\mathsf{cl}(Q_{\mathsf{cert}}(A_n) \cup \{\exists\, \alpha_n.B' \in \mathsf{BC} \mid B' \in \Gamma(t_{n-1})\})$ *if $n > 1$.*

Note that the rules $\mathbf{R1}$ to $\mathbf{R6}$ are used in every step of this inductive definition. The mentioned propagation of information along traces is now as follows: if there is a trace t to B, then any domain element that satisfies B in *some* world must satisfy the concepts in $\Gamma(t)$ in some other world. So if for example $P_{>0}A \in \Gamma(t)$, we need to add $P_{>0}A$ also to $Q_{\mathsf{cert}}(B)$ and to $Q(B)$.

Figure 4 shows the rules used for completing the data structures $Q(\cdot)$, $Q_{\mathsf{cert}}(\cdot)$, and $R(\cdot)$. Note that **S6** and **S7** implement the propagation of information along traces, as discussed above. Our algorithm for deciding (positive) subsumption starts with the initial data structures defined above and then exhaustively applies the rules shown in Figure 4. To decide whether $\mathcal{T} \models^+ A \sqsubseteq B$, it then simply checks whether $B \in Q(A)$.

Lemma 3. *Let $\mathcal{T}$ be a Prob-$\mathcal{EL}_r^{>0;=1}$-TBox in normal form and A, B be concept names. Then $\mathcal{T} \models^+ A \sqsubseteq B$ iff, after exhaustive rule application, $B \in Q(A)$.*

We now argue that the algorithm can be implemented using only polynomial space. First, it is easy to see that there can be only polynomially many rule applications: every rule application extends the data structures $Q(\cdot)$, $Q_{\mathsf{cert}}(\cdot)$, and $R(\cdot)$, but these structures consist of polynomially many sets, each with at most polynomially many elements. It thus remains to verify that each rule application can be executed using only polyspace, which is obvious for all rules except those involving traces, i.e., **S6** and **S7**. For these rules, we first note that it is not necessary to consider all (infinitely many!) traces. In fact, a straightforward 'pumping argument' can be used to show that there is a trace t to B with some relevant concept $C \in \Gamma(t)$ iff there is a *non-repeating* such trace, i.e., a trace t' of length n such that for all distinct $k,\ell \leq n$, we have $\Gamma(t'_k) \neq \Gamma(t'_\ell)$.

S1 apply **R1-R6** to $Q(A)$ and $Q_{\mathsf{cert}}(A)$

S2 if $P_* B \in Q(A)$
then add $P_* B$ to $Q_{\mathsf{cert}}(A)$

S3 if $C \in Q_{\mathsf{cert}}(A)$
then add $P_{=1}C$ and C to $Q(A)$

S4 If $\exists\alpha.B \in Q(A)$ with α a probabilistic role
then add (A, B) to $R(\alpha)$.

S5 If $P_{>0}B_1 \in Q(A)$, $(B_1, B_2) \in R(\alpha)$, $B_3 \in Q_{\mathsf{cert}}(B_2)$
then add $\exists\alpha.B_3$ to $Q_{\mathsf{cert}}(A)$

S6 if t is a trace to B and $P_* A \in \Gamma(t)$
then add $P_* A$ to $Q_{\mathsf{cert}}(B)$

S7 if t is a trace to B and $\exists\alpha.A \in \Gamma(t)$ with α a probabilistic role

Fig. 4. The rules for completing the data structures.

Clearly, the length of non-repeating traces is bounded by 2^m, m the size of $\mathcal{T}$. To get to polyspace, we use a non-deterministic approach, enabled by Savitch's theorem: to check whether there is a trace t to B with $C \in \Gamma(t)$, we guess t step-by-step, at each time keeping only a single A_i, α_i and $\Gamma(t_i)$ in memory. When we reach a situation where $A_i = B$ and $C \in \Gamma(t_i)$, our guessing was successful and we apply the rule. We also maintain a binary counter of the number of steps that have been guessed so far. As soon as this counter exceeds 2^m, the maximum length of non-repeating traces, we stop the guessing and do not apply the rule. Clearly, this yields a polyspace algorithm.

Theorem 5. *Deciding positive subsumption in Prob-$\mathcal{EL}_r^{>0;=1}$ with respect to general TBoxes is* PSPACE*-complete.*

As a byproduct, the proof of Lemma 3 yields a unique least model (in the sense of Horn logic), thus proving convexity of Prob-$\mathcal{EL}_r^{>0;=1}$. Note that positive subsumption in Prob-$\mathcal{EL}_r^{>0;=1}$ is actually the same as subsumption in the two-dimensional description logic S5$_{\mathcal{EL}}$, which is thus also PSPACE-complete. Using a Turing reduction similar to that shown in Figure 2, we can 'lift' the result from positive subsumption to unrestricted subsumption.

Theorem 6. *Subsumption in Prob-$\mathcal{EL}_r^{>0;=1}$ relative to general TBoxes is* PSPACE*-complete.*

5 Conclusion

We have established a fairly complete picture of the complexity of subsumption in Prob-$\mathcal{EL}$, although some questions remain open. We speculate that Theorem 2 can be proved also when $\sim$ is $=$ with only minor changes (e.g. rule **R3** becomes unsound). It would be interesting to verify the conjecture made below Theorem 4 that (positive) subsumption in Prob-$\mathcal{EL}^{\sim p;\sim q}$ relative to classical TBoxes is in PTIME whenever $p \geq q \geq p^2$ and that, otherwise, it is EXPTIME-hard relative to the empty TBox. It is even conceivable that the conjectured PTIME result can be further generalized to any set of probability values $\mathcal{P} \subseteq [0, 1]$ as long as $q \geq p^2$ whenever $p, q \in \mathcal{P}$ and $p \geq q$. Moreover, variants of Theorem 4 that involve, additionally or exclusively, the case where $\sim$ is $=$ would also be of interest.

Acknowledgements The present work was supported by the DFG project on probabilistic description logics (LU1417/1-1) and DAAD-CONACYT grant 206550.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. of the 18th Int. Joint Conf. on Artif. Intell. (IJCAI03). pp. 325–330. Morgan Kaufmann (2003)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artif. Intell. (IJCAI05). pp. 364–369. Morgan Kaufmann (2005)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
4. Bacchus, F.: Representing and Reasoning with Probabilistic Knowledge. MIT Press, Cambridge, MA (1990)
5. da Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Formal Ontology in Information Systems (FOIS06). Frontiers in Artif. Intell. and Applic., vol. 150, pp. 237–249. IOS Press (2006)
6. Gutierrez-Basulto, V., Jung, J.C., Lutz, C., Schröder, L.: A closer look at the probabilistic description logic Prob-$\mathcal{EL}$. In: Proc. of the 25th AAAI Conf. on Artif. Intell. (AAAI11). AAAI Press (2011)
7. Halpern, J.Y.: Reasoning About Uncertainty. MIT Press (2003)
8. Halpern, J.Y., Rabin, M.O.: A logic to reason about likelihood. Artif. Intell. 32, 379–405 (1987)
9. Herzig, A.: Modal probability, belief, and actions. Fund. Inf. 57, 323–344 (2003)
10. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Proc. of the 4th Int. Conf. on Princ. of Knowledge Repres. and Reas. (KR94), pp. 305–316. Morgan Kaufmann (1994)
11. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: Proc. of the 21st Int. Joint Conf. on Artif. Intell. (IJCAI09). pp. 2040–2045 (2009)
12. Lukasiewicz, T.: Tractable probabilistic description logic programs. In: Proc. of the 1st Int. Conf. in Scalable Uncertainty Management (SUM07), number 4772 in LNCS, 143–156. Springer Verlag.
13. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. 172, 852–883 (2008)
14. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
15. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In: Proc. of the 12th Int. Conf. on Princ. of Knowledge Repres. and Reas. (KR10). AAAI Press (2010)
16. Schulz, S., Suntisrivaraporn, B., Baader, F.: SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In: Proc. of The Medinfo 2007 Congress. Studies in Health Technology and Informatics (SHTI-series), IOS Press (2007)

Unchain My $\mathcal{EL}$ Reasoner

Yevgeny Kazakov, Markus Krötzsch, and František Simančík

Department of Computer Science, University of Oxford, UK

Abstract. We study a restriction of the classification procedure for $\mathcal{EL}^{++}$ where the inference rule for complex role inclusion axioms (RIAs) is applied in a "left-linear" way in analogy with the well-known procedure for computing the transitive closure of a binary relation. We introduce a notion of left-admissibility for a set of RIAs, which specifies when a subset of RIAs can be used in a left-linear way without loosing consequences, prove a criterion which can be used to effectively check this property, and describe some preliminary experimental results analyzing when the restricted procedure can give practical improvements.

1 Introduction

The description logic (DL) $\mathcal{EL}$ and its extension $\mathcal{EL}^{++}$ [1] provide the bases of the OWL EL profile of the Web Ontology Language [6] and are distinguished by having tractable worst-case complexity for the standard DL reasoning problems. The nice computational properties of $\mathcal{EL}$-style reasoning procedures such as optimal (polynomial) worst-case complexity and "pay-as-you-go" behavior, are commonly mentioned as main reasons for the improved practical performance of reasoners based on such procedures for large ontologies such as SNOMED CT [2,5,3,8].

Although $\mathcal{EL}^{++}$ admits a polynomial reasoning procedure, different features of $\mathcal{EL}^{++}$ contribute differently to the degree of this polynomial [4]. In particular, the $\mathcal{EL}^{++}$ rule for dealing with complex role inclusion axioms (RIAs) has $O(n^4)$ time complexity, which is higher than for other rules. Even for a single transitivity axiom, the rule can result in $O(n^3)$ inferences. Although complex role inclusion axioms are not used as commonly as other constructors in existing ontologies such as SNOMED CT, this might change in the future as more OWL EL ontologies emerge.

Inspired by an $O(n^2)$ algorithm for computing the transitive closure of a binary relation, in this paper we propose a refinement of the $\mathcal{EL}^{++}$ rule for dealing with complex RIAs. Our main idea is to restrict the rule so that inferences are applied in a *left-linear way*, that is, only a restricted number of the "initial" axioms can be used in all premises of the rule except for the left-most. To this end, we (i) formulate a notion of *left-admissibility* describing subsets of complex RIAs that can be used in a left-linear way without losing consequences, (ii) prove a criterion for left-admissibility that can be checked in polynomial time, and (iii) provide an experimental evaluation measuring the proportion of left-admissible RIAs and reduction in the number of inferences for a selection of commonly-used ontologies.

Table 1. The inference rules for $\mathcal{ELR}$, with N_C the set of all concept names

$$\mathsf{C_0}\ \frac{}{C \sqsubseteq C} : C \in N_C$$

$$\mathsf{C_1}\ \frac{C \sqsubseteq C'}{C \sqsubseteq D} : C' \sqsubseteq D \in \mathrm{KB}$$

$$\mathsf{C_2}\ \frac{(C \sqsubseteq C_i)_{i=1}^n}{C \sqsubseteq D} : \begin{array}{l} n > 1 \\ \bigsqcap_{i=1}^n C_i \sqsubseteq D \in \mathrm{KB} \end{array}$$

$$\mathsf{C_3}\ \frac{C \sqsubseteq C'}{C \sqsubseteq \exists R.D} : C' \sqsubseteq \exists R.D \in \mathrm{KB}$$

$$\mathsf{C_4}\ \frac{C \sqsubseteq \exists R.D \quad D \sqsubseteq D'}{C \sqsubseteq E} : \exists R.D' \sqsubseteq E \in \mathrm{KB}$$

$$\mathsf{C_5}\ \frac{C \sqsubseteq \exists S.D}{C \sqsubseteq \exists R.D} : S \sqsubseteq R \in \mathrm{KB}$$

$$\mathsf{C_6}\ \frac{(C_{i-1} \sqsubseteq \exists R_i.C_i)_{i=1}^n}{C_0 \sqsubseteq \exists R.C_n} : \begin{array}{l} n > 1 \\ R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathrm{KB} \end{array}$$

2 Reasoning in $\mathcal{ELR}$

We first introduce a basic classification calculus for the description logic $\mathcal{ELR}$ that will serve as a baseline for our study. $\mathcal{ELR}$ is the DL that supports only conjunction, existential role restrictions, role hierarchies, and role inclusion axioms, each of which can be used in arbitrary general concept inclusions and role inclusion axioms. We do not require regularity of RBoxes, and $\mathcal{ELR}$ can thus be viewed as a fragment of $\mathcal{EL}^{++}$ without top, bottom, nominals, and concrete domains. We use the following notation for role inclusion axioms.

Definition 1. *A* role chain *ρ is an expression of the form $R_1 \cdot \ldots \cdot R_n$, $n \geq 0$; when $n = 0$ then $\rho = \epsilon$ is the* empty role chain *and when $n \geq 2$ then ρ is a* complex role chain. *We denote by $\rho_1 \cdot \rho_2$ the* concatenation *of two role chains ρ_1 and ρ_2. A* (complex) role inclusion axiom *(short RIA) is an expression of the form $\rho \sqsubseteq R$ where ρ is a non-empty (complex) role chain and R a role. An RBox $\mathcal{R}$ is a finite set of RIAs.*

Table 1 shows the rules of a classification calculus for $\mathcal{ELR}$, obtained by restricting the calculus for $\mathcal{EL}^{++}$ [1]. The input to the rules are axioms from an $\mathcal{ELR}$ knowledge base that have been normalized as in [1]. The main difference is that we treat n-ary conjunctions/role chains in a single application of $\mathsf{C_2}/\mathsf{C_6}$, corresponding to the implementation we used for experiments. Each rule of inference consists of a premise, a conclusion, and possible side conditions. The calculus derives axioms of the form $C \sqsubseteq D$ and $C \sqsubseteq \exists R.D$ based on an input knowledge base KB, and it is sound and complete for classification in the sense that an axiom $C \sqsubseteq D$ is entailed by KB if and only if the exhaustive application of the inference rules can be used to derive $C \sqsubseteq D$. This follows immediately from the according result in [1] since it is easy to see that our rules correspond to the inference rules in that paper: $\mathsf{C_0}$ corresponds to the initialization, $\mathsf{C_1}$ to $\mathsf{CR_1}$, $\mathsf{C_2}$ to $\mathsf{CR_2}$, $\mathsf{C_3}$ to $\mathsf{CR_3}$, $\mathsf{C_4}$ to $\mathsf{CR_4}$, $\mathsf{C_5}$ to $\mathsf{CR_{10}}$, and $\mathsf{C_6}$ to $\mathsf{CR_{11}}$.

3 Linear Use of Role Inclusion Axioms

One of the simplest examples of complex RIAs is a transitivity axiom:

$$R \cdot R \sqsubseteq R. \tag{1}$$

Transitivity axioms occur in many ontologies where they are used to express hierarchical relations between concepts, such as "part-of" or "child-of" hierarchies. Let us consider an ontology containing axioms expressing a simple R-hierarchy:

$$A_i \sqsubseteq \exists R.A_{i+1}, \quad 1 \leq i < n. \tag{2}$$

If we apply rule $\mathsf{C_6}$ to these axioms, we derive exactly axioms of the form:

$$A_i \sqsubseteq \exists R.A_j, \quad 1 \leq i < j \leq n, \tag{3}$$

using the following instances of rule $\mathsf{C_6}$:

$$\frac{A_i \sqsubseteq \exists R.A_j \quad A_j \sqsubseteq \exists R.A_k}{A_i \sqsubseteq \exists R.A_k} : \quad 1 \leq i < j < k \leq n. \tag{4}$$

There are exactly $n \cdot (n-1)/2$ possible axioms of the form (3) and there are exactly $n \cdot (n-1) \cdot (n-2)/6$ rule applications in (4). In particular, every axiom in (3) is derived $(n-2)/3$ times in average. Clearly, this demonstrates that rule $\mathsf{C_6}$ can be a source of inefficiency, especially for large n.

The inferences (4) look like the computation of the transitive closure for a binary relation, if we read $C \sqsubseteq \exists R.D$ as $\langle C, D\rangle \in R$. Using this correspondence, we can apply a more efficient algorithm for computing the transitive closure by restricting the second premise in $\mathsf{C_6}$ to the initial axioms only. Specifically, let us use $\sqsubseteq^0$ to distinguish the initial (told) axioms $C \sqsubseteq^0 \exists R.D$ from the axioms $C \sqsubseteq \exists R.D$ that are derived using inference rules. Then one can restrict rule $\mathsf{C_6}$ for transitivity axioms as follows:

$$\frac{C_1 \sqsubseteq \exists R.C_2 \quad C_2 \sqsubseteq^0 \exists R.C_3}{C_1 \sqsubseteq \exists R.C_3} : R \cdot R \sqsubseteq R \in \mathrm{KB}. \tag{5}$$

We will call the rule (5) a *left-linear rule* in analogy with left-linear production rules in context-free grammars because the conclusions of other inferences can be used here only in the left premise. By applying (5) to the input axioms (2) (written using $\sqsubseteq^0$), we obtain inferences of the following form:

$$\frac{A_i \sqsubseteq \exists R.A_j \quad A_j \sqsubseteq^0 \exists R.A_{j+1}}{A_i \sqsubseteq \exists R.A_{j+1}} : \quad 1 \leq i < j < n. \tag{6}$$

Table 2. Left-linear inference rules for $\mathcal{ELR}$

$$\mathbf{L_0}\ \frac{}{C \sqsubseteq_{\mathcal{L}} C} : C \in N_C$$

$$\mathbf{L_1}\ \frac{C \sqsubseteq_{\mathcal{L}} C'}{C \sqsubseteq_{\mathcal{L}} D} : C' \sqsubseteq D \in \mathrm{KB}$$

$$\mathbf{L_2}\ \frac{(C \sqsubseteq_{\mathcal{L}} C_i)_{i=1}^n}{C \sqsubseteq_{\mathcal{L}} D} : \begin{array}{l} n > 1 \\ \bigsqcap_{i=1}^n C_i \sqsubseteq D \in \mathrm{KB} \end{array}$$

$$\mathbf{L_3}\ \frac{C \sqsubseteq_{\mathcal{L}} C'}{C \sqsubseteq_{\mathcal{L}}^0 \exists R.D} : C' \sqsubseteq \exists R.D \in \mathrm{KB}$$

$$\mathbf{L_4}\ \frac{C \sqsubseteq_{\mathcal{L}} \exists R.D \quad D \sqsubseteq_{\mathcal{L}} D'}{C \sqsubseteq_{\mathcal{L}} E} : \exists R.D' \sqsubseteq E \in \mathrm{KB}$$

$$\mathbf{L_5}\ \frac{C \sqsubseteq_{\mathcal{L}} \exists S.D}{C \sqsubseteq_{\mathcal{L}} \exists R.D} : S \sqsubseteq R \in \mathcal{R}$$

$$\mathbf{L_5'}\ \frac{C \sqsubseteq_{\mathcal{L}}^0 \exists S.D}{C \sqsubseteq_{\mathcal{L}}^0 \exists R.D} : S \sqsubseteq R \in \mathcal{R}$$

$$\mathbf{L_6}\ \frac{(C_{i-1} \sqsubseteq_{\mathcal{L}} \exists R_i.C_i)_{i=1}^n}{C_0 \sqsubseteq_{\mathcal{L}} \exists R.C_n} : \begin{array}{l} n > 1 \\ R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R} \setminus \mathcal{L} \end{array}$$

$$\mathbf{L_6'}\ \frac{C_0 \sqsubseteq_{\mathcal{L}} \exists R_1.C_1 \quad (C_{i-1} \sqsubseteq_{\mathcal{L}}^0 \exists R_i.C_i)_{i=2}^n}{C_0 \sqsubseteq_{\mathcal{L}} \exists R.C_n} : \begin{array}{l} n > 1 \\ R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{L} \end{array}$$

It is easy to see that there are exactly $(n-1) \cdot (n-2)/2$ rule applications in (6) producing exactly those axioms in (3) that are not in (2), and that every such axiom is derived exactly once. Clearly, this strategy represents an improvement over the application of the (unrestricted) rule $\mathbf{C_6}$.

We use the idea above to formulate a calculus for $\mathcal{ELR}$ with a restricted version of rule $\mathbf{C_6}$. In order to do that, we need to specify where the "initial" axioms $C \sqsubseteq^0 \exists R.D$ come from. Clearly, we cannot take such axioms just from the knowledge base, since otherwise, e.g., we would not be able to derive $A \sqsubseteq \exists R.D$ for KB consisting of $A \sqsubseteq \exists R.B$, $B \sqsubseteq C$, $C \sqsubseteq \exists R.D$, and $R \cdot R \sqsubseteq R$, as we cannot avoid using $B \sqsubseteq \exists R.D$ in the second premise of $\mathbf{C_6}$. Similarly, we need to allow initial axioms to be produced by $\mathbf{C_5}$, since otherwise $A \sqsubseteq \exists R.C$ cannot be derived for KB consisting of $A \sqsubseteq \exists R.B$, $B \sqsubseteq \exists S.C$, $S \sqsubseteq R$, and $R \cdot R \sqsubseteq R$.

The new calculus for $\mathcal{ELR}$ is formulated in Table 2. The calculus is parametrized with a distinguished subset $\mathcal{L} \subseteq \mathcal{R}$ of complex RIAs. The RIAs in $\mathcal{L}$ can, similar to the transitivity axiom in the example above, only be used in a left-linear version $\mathbf{L_6'}$ of rule $\mathbf{C_6}$. The remaining axioms from $\mathcal{R} \setminus \mathcal{L}$ can be used without restrictions in rule $\mathbf{L_6}$. The *initial axioms* of the form $C \sqsubseteq_{\mathcal{L}}^0 \exists R.D$ are produced by rules $\mathbf{L_3}$ and $\mathbf{L_5'}$. We use $\mathcal{L}$ in the subscripts of $\sqsubseteq_{\mathcal{L}}$ and $\sqsubseteq_{\mathcal{L}}^0$ to emphasize that these relations depend on $\mathcal{L}$. We implicitly assume that $\sqsubseteq_{\mathcal{L}}^0 \subseteq \sqsubseteq_{\mathcal{L}}$; in particular, axioms of the form $C \sqsubseteq_{\mathcal{L}}^0 \exists R.D$ can also be used as premises of rules $\mathbf{L_4}$ and $\mathbf{L_6}$ and as the first premise of rule $\mathbf{L_6'}$. Note that if $\mathcal{L} = \emptyset$, our new calculus coincides with the original calculus for $\mathcal{ELR}$ (ignoring the distinction

Table 3. Left-linear composition of roles

$$\textsf{E}_0 \quad \frac{}{R \sqsubseteq^0_{\mathcal{L}} R} : R \in N_R$$

$$\textsf{E}_1 \quad \frac{\rho \sqsubseteq_{\mathcal{L}} S}{\rho \sqsubseteq_{\mathcal{L}} R} : S \sqsubseteq R \in \mathcal{R}$$

$$\textsf{E}'_1 \quad \frac{T \sqsubseteq^0_{\mathcal{L}} S}{T \sqsubseteq^0_{\mathcal{L}} R} : S \sqsubseteq R \in \mathcal{R}$$

$$\textsf{E}_2 \quad \frac{(\rho_i \sqsubseteq_{\mathcal{L}} R_i)_{i=1}^n}{\rho_1 \cdot \ldots \cdot \rho_n \sqsubseteq_{\mathcal{L}} R} : \begin{array}{l} n > 1 \\ R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R} \setminus \mathcal{L} \end{array}$$

$$\textsf{E}'_2 \quad \frac{\rho_1 \sqsubseteq_{\mathcal{L}} R \quad (T_i \sqsubseteq^0_{\mathcal{L}} R_i)_{i=2}^n}{\rho_1 \cdot T_2 \cdot \ldots \cdot T_n \sqsubseteq_{\mathcal{L}} R} : \begin{array}{l} n > 1 \\ R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{L} \end{array}$$

between $\sqsubseteq^0_{\mathcal{L}}$ and $\sqsubseteq_{\mathcal{L}}$). Clearly, the larger $\mathcal{L}$ is, the more restricted our rules are, and so the less inferences are possible. Thus, in the remainder of the paper we are concerned with the problem of finding subsets $\mathcal{L}$ of a given $\mathcal{R}$ which do not result in lost consequences relative to the original calculus in Table 1.

4 Left-Admissible Role Inclusion Axioms

In this section we are concerned with the problem of how to determine, given a set of complex RIAs $\mathcal{L} \subseteq \mathcal{R}$, whether the calculus in Table 2 produces the same consequences as the original calculus in Table 1. In order to study the properties of the calculus in Table 2 for different subsets $\mathcal{L}$ of $\mathcal{R}$, consider the smallest relations $\sqsubseteq^0_{\mathcal{L}} \subseteq \sqsubseteq_{\mathcal{L}}$ on role chains satisfying the properties in Table 3. Note the similarities between the rules in Table 3 and rules in Table 2. Also note that unlike the derivation relation $\sqsubseteq^0_{\mathcal{L}}$ in Table 2, the relation $\sqsubseteq^0_{\mathcal{L}}$ in Table 3 does not depend on $\mathcal{L}$ and coincides with the closure of the role hierarchy. The following lemma can be easily proved using the correspondence between the rules $\textsf{L}_3$, $\textsf{L}_5$, $\textsf{L}'_5$, $\textsf{L}_6$ and $\textsf{L}'_6$ and the rules $\textsf{E}_0$, $\textsf{E}_1$, $\textsf{E}'_1$, $\textsf{E}_2$ and $\textsf{E}'_2$.

Lemma 1. *For every subset $\mathcal{L} \subseteq \mathcal{R}$ of complex RIAs, all concepts A and B, and every role R, the following two conditions are equivalent:*

(i) $A \sqsubseteq_{\mathcal{L}} \exists R.B$.
(ii) *There exist $C_0, \ldots, C_n$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq_{\mathcal{L}} R$ such that $A = C_0$, $B = C_n$, and $C_{i-1} \sqsubseteq^0_{\mathcal{L}} \exists R_i.C_i$ $(1 \leq i \leq n)$.*

The following properties can be proved by induction on $S \sqsubseteq^0_{\mathcal{L}} T$:

$$\text{if } R \sqsubseteq^0_{\mathcal{L}} S \text{ and } S \sqsubseteq^0_{\mathcal{L}} T, \text{ then } R \sqsubseteq^0_{\mathcal{L}} T, \tag{7}$$

$$\text{if } \rho \sqsubseteq_{\mathcal{L}} S \text{ and } S \sqsubseteq^0_{\mathcal{L}} T, \text{ then } \rho \sqsubseteq_{\mathcal{L}} T. \tag{8}$$

The necessary and sufficient condition on $\mathcal{L} \subseteq \mathcal{R}$ that guarantees that our new calculus for $\mathcal{ELR}$ derives the same consequences as the original calculus, can now be defined as follows:

Definition 2. *A set of complex RIAs $\mathcal{L} \subseteq \mathcal{R}$ is* left-admissible *for $\mathcal{R}$ if the following condition holds:*

$$\textit{if } (\rho_i \sqsubseteq_{\mathcal{L}} R_i)_{i=1}^n \textit{ and } R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R}, \textit{ then } \rho_1 \cdot \ldots \cdot \rho_n \sqsubseteq_{\mathcal{L}} R. \quad (9)$$

Intuitively, $\mathcal{L} \subseteq \mathcal{R}$ is *left-admissible* if the relation $\sqsubseteq_{\mathcal{L}}$ is closed under the unrestricted version of the rule $\mathbf{E_2}$ when RIAs $R_1 \cdot \ldots \cdot R_n \sqsubseteq R$ can be taken not only from $\mathcal{R} \setminus \mathcal{L}$, but from the whole $\mathcal{R}$ (note the similarity of (9) and $\mathbf{E_2}$). Left-admissibility thus ensures that the relation $\sqsubseteq_{\mathcal{L}}$ coincides with the unrestricted relation $\sqsubseteq_{\emptyset}$, i.e., the relation for $\mathcal{L} = \emptyset$.

Example 1. Consider $\mathcal{R}$ consisting of the axiom

$$\mathsf{isPartOf} \cdot \mathsf{isProperPartOf} \sqsubseteq \mathsf{isProperPartOf}. \quad (10)$$

It is easy to show that the following relations hold for any (of the two) $\mathcal{L} \subseteq \mathcal{R}$:

$$\mathsf{isPartOf} \sqsubseteq_{\mathcal{L}}^{o} \mathsf{isPartOf} \quad (\text{by } \mathbf{E_0}), \quad (11)$$

$$\mathsf{isProperPartOf} \sqsubseteq_{\mathcal{L}}^{o} \mathsf{isProperPartOf} \quad (\text{by } \mathbf{E_0}), \quad (12)$$

$$\mathsf{isPartOf} \cdot \mathsf{isProperPartOf} \sqsubseteq_{\mathcal{L}} \mathsf{isProperPartOf} \quad (\text{by } \mathbf{E_2} \text{ or } \mathbf{E_2'}). \quad (13)$$

The following relation, however, holds only for $\mathcal{L} = \emptyset$:

$$\mathsf{isPartOf} \cdot \mathsf{isPartOf} \cdot \mathsf{isProperPartOf} \sqsubseteq_{\mathcal{L}} \mathsf{isProperPartOf} \quad (\text{by } \mathbf{E_2}). \quad (14)$$

When $\mathcal{L} = \mathcal{R}$, one cannot use $\mathbf{E_2'}$ to produce (14) from (11) and (13) using (10). Therefore $\mathcal{L} = \mathcal{R}$ is not left-admissible for $\mathcal{R}$ according to Definition 2.

Now suppose $\mathcal{R}$ is extended with the transitivity axiom

$$\mathsf{isPartOf} \cdot \mathsf{isPartOf} \sqsubseteq \mathsf{isPartOf}. \quad (15)$$

Then, similarly, for any $\mathcal{L} \subseteq \mathcal{R}$ we have

$$\mathsf{isPartOf} \cdot \mathsf{isPartOf} \sqsubseteq_{\mathcal{L}} \mathsf{isPartOf} \quad (\text{by } \mathbf{E_2} \text{ or } \mathbf{E_2'}). \quad (16)$$

And now (14) can be produced from (16) and (12) using (10) for any $\mathcal{L} \subseteq \mathcal{R}$. In fact, one can show that any $\mathcal{L} \subseteq \mathcal{R}$ will be left-admissible for the extended $\mathcal{R}$.

Theorem 1. *Let* KB *be a knowledge base with RBox $\mathcal{R}$, and $\mathcal{L} \subseteq \mathcal{R}$ be left-admissible for $\mathcal{R}$. Then $C \sqsubseteq D$ is derivable by rules in Table 1 using* KB *iff $C \sqsubseteq_{\mathcal{L}} D$ is derivable by rules in Table 2 using* KB.

Proof. The "if" direction of the theorem is straightforward since each rule in Table 2 is a restriction of a corresponding rule in Table 1.

To prove the "only if" direction, assume to the contrary that there exists $C \sqsubseteq D$ derivable by rules in Table 1 such that $C \not\sqsubseteq_{\mathcal{L}} D$. Without loss of generality, $C \sqsubseteq D$ is produced by some rule in Table 1 from some premises $C_i \sqsubseteq D_i$, $0 \leq i < n$, $n \geq 0$, such that $C_i \sqsubseteq_{\mathcal{L}} D_i$. We obtain contradiction by considering all possible cases for such a rule.

The only non-trivial case is an application of the rule $\mathsf{C_6}$ since all other rules have a direct counterpart in Table 2. In this case $D_{i-1} = \exists R_i.C_i$ $(1 \leq i \leq n)$, $C = C_0$, $D = \exists R.C_n$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R}$. By case $(i) \Rightarrow (ii)$ of Lemma 1 applied to each $C_{i-1} \sqsubseteq_\mathcal{L} \exists R_i.C_i$, there exist $C_i^{j-1} \sqsubseteq_\mathcal{L}^0 \exists R_i^j.C_i^j$ $(1 \leq j \leq m_i)$ such that $C_i^0 = C_{i-1}$, $C_i^{m_i} = C_i$ and $R_i^1 \cdot \ldots \cdot R_i^{m_i} \sqsubseteq_\mathcal{L} R_i$. Define $\rho_i := R_i^1 \cdot \ldots \cdot R_i^{m_i}$. Since $\rho_i \sqsubseteq_\mathcal{L} R_i$ $(1 \leq i \leq n)$, $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R}$, and $\mathcal{L}$ is left-admissible, we have $\rho_i \cdot \ldots \cdot \rho_n \sqsubseteq_\mathcal{L} R$. By case $(i) \Leftarrow (ii)$ of Lemma 1 for $((C_i^{j-1} \sqsubseteq_\mathcal{L}^0 \exists R_i^j.C_i^j)_{j=1}^{m_i})_{i=1}^n$ using $\rho_i \cdot \ldots \cdot \rho_n \sqsubseteq_\mathcal{L} R$ we obtain $C = C_0 = C_1^0 \sqsubseteq_\mathcal{L} \exists R.C_n^{m_n} = \exists R.C_n = D$, which contradicts $C \not\sqsubseteq_\mathcal{L} D$. This proves the theorem. □

5 Recognizing Left-Admissibility

It is difficult in general to verify the conditions of left-admissibility formulated in Definition 2 since this requires checking property (9) for a potentially infinite number of role chains ρ_i. In this section we give an equivalent formulation for left-admissibility, which can be checked in polynomial time.

We start with the following sufficient condition for left-admissibility:

Lemma 2. *Let $\mathcal{L} \subseteq \mathcal{R}$ be sets of complex RIAs that satisfy the property:*

$$\text{if } \rho \sqsubseteq S \in \mathcal{R} \text{ and } \rho_1 \cdot S \cdot \rho_2 \sqsubseteq_\mathcal{L} R, \text{ then } \rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R. \tag{17}$$

Then $\mathcal{L}$ is left-admissible for $\mathcal{R}$.

Proof. We first show that (17) implies the following stronger property:

$$\text{if } \rho \sqsubseteq_\mathcal{L} S \text{ and } \rho_1 \cdot S \cdot \rho_2 \sqsubseteq_\mathcal{L} R, \text{ then } \rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R. \tag{18}$$

The proof of (18) is by induction on the derivation of $\rho \sqsubseteq_\mathcal{L} S$. In the base case $\mathsf{E_0}$ we have $\rho = S$ and the claim $\rho_1 \cdot S \cdot \rho_2 \sqsubseteq_\mathcal{L} R$ is part of the precondition. In all other cases $\mathsf{E_1}$–$\mathsf{E_2'}$ there exist $(\sigma_i \sqsubseteq_\mathcal{L} S_i)_{i=1}^n$, $n \geq 1$ such that $\rho = \sigma_1 \cdot \ldots \cdot \sigma_n$ and $S_1 \cdot \ldots \cdot S_n \sqsubseteq S \in \mathcal{R}$. By (17) $\rho_1 \cdot S_1 \cdot \ldots \cdot S_n \cdot \rho_2 \sqsubseteq_\mathcal{L} R$. Now use the induction hypothesis (18) for each $\sigma_i \sqsubseteq_\mathcal{L} S_i$ to iteratively expand the left-hand side of $\rho_1 \cdot S_1 \cdot \ldots \cdot S_n \cdot \rho_2 \sqsubseteq_\mathcal{L} R$ to obtain the claim $\rho_1 \cdot \rho \cdot \rho_2 = \rho_1 \cdot \sigma_1 \cdot \ldots \cdot \sigma_n \cdot \rho_2 \sqsubseteq_\mathcal{L} R$.

To finish the proof of the lemma, we now show that (18) implies (9). To this end, consider any $(\rho_i \sqsubseteq_\mathcal{L} R_i)_{i=1}^n$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R}$. We must prove that $\rho_1 \cdot \ldots \cdot \rho_n \sqsubseteq_\mathcal{L} R$. For this, note that $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R}$ implies $R_1 \cdot \ldots \cdot R_n \sqsubseteq_\mathcal{L} R$, and use (18) for each $\rho_i \sqsubseteq_\mathcal{L} R_i$ to iteratively expand the left-hand side of $R_1 \cdot \ldots \cdot R_n \sqsubseteq_\mathcal{L} R$ to obtain the desired $\rho_1 \cdot \ldots \cdot \rho_n \sqsubseteq_\mathcal{L} R$. □

The following lemma formulates some useful closure properties of the relation $\sqsubseteq_\mathcal{L}$, which hold for arbitrary $\mathcal{L}$:

Lemma 3. *Let $\mathcal{L} \subseteq \mathcal{R}$ be sets of RIAs. If $\rho_1 \sqsubseteq_\mathcal{L} S_1$, $(T_i \sqsubseteq_\mathcal{L}^0 S_i)_{i=2}^n$, $n \geq 1$, and $S_1 \cdot \ldots \cdot S_n \sqsubseteq_\mathcal{L} R$, then $\rho_1 \cdot T_2 \cdot \ldots \cdot T_n \sqsubseteq_\mathcal{L} R$.*

Proof. Let $\rho = \rho_1 \cdot T_2 \cdot \ldots \cdot T_n$. We will show $\rho \sqsubseteq_\mathcal{L} R$ by induction on the derivation of $S_1 \cdot \ldots \cdot S_n \sqsubseteq_\mathcal{L} R$.

E₀: $n = 1$ and $S_1 = R$. The claim $\rho_1 \sqsubseteq_\mathcal{L} R$ is part of the precondition.

E₁: $S_1 \cdot \ldots \cdot S_n \sqsubseteq_\mathcal{L} S$ and $S \sqsubseteq R \in \mathcal{R}$. By the induction hypothesis $\rho \sqsubseteq_\mathcal{L} S$ from which $\rho \sqsubseteq_\mathcal{L} R$ follows by **E₁**.

E′₁: Analogous to the case of **E₁**.

E₂: $S_1 \cdot \ldots \cdot S_n = \sigma_1 \cdot \ldots \cdot \sigma_m$, $m > 1$, $(\sigma_i \sqsubseteq_\mathcal{L} R_i)_{i=1}^m$ and $R_1 \cdot \ldots \cdot R_m \sqsubseteq R \in \mathcal{R} \setminus \mathcal{L}$. Let s_i and e_i be the start and the end indices of σ_i in $S_1 \cdot \ldots \cdot S_n$. By the induction hypothesis $\rho_1 \cdot T_2 \cdot \ldots \cdot T_{e_1} \sqsubseteq_\mathcal{L} R_1$ and $(T_{s_i} \cdot \ldots \cdot T_{e_i} \sqsubseteq_\mathcal{L} R_i)_{i=2}^m$, from which $\rho \sqsubseteq_\mathcal{L} R$ follows by **E₂**.

E′₂: $S_1 \cdot \ldots \cdot S_k \sqsubseteq_\mathcal{L} R_1$, $k \geq 1$, $(S_{i+k-1} \sqsubseteq^0_\mathcal{L} R_i)_{i=2}^m$, $m > 1$, $k + m = n + 1$ and $R_1 \cdot \ldots \cdot R_m \sqsubseteq R \in \mathcal{L}$. By the induction hypothesis $\rho_1 \cdot T_2 \cdot \ldots \cdot T_k \sqsubseteq_\mathcal{L} R_1$. By (7) we have $(T_{i+k-1} \sqsubseteq^0_\mathcal{L} R_i)_{i=2}^m$. Then $\rho \sqsubseteq_\mathcal{L} R$ follows by **E′₂**. □

We are now ready to formulate our main criterion for left-admissibility:

Theorem 2. *A subset $\mathcal{L} \subseteq \mathcal{R}$ of complex RIAs is left-admissible for an RBox $\mathcal{R}$ if and only if the following property holds:*

$$\textit{if } \rho \sqsubseteq S_1 \in \mathcal{R},\ S_1 \sqsubseteq^0_\mathcal{L} S_2 \textit{ and } \rho_1 \cdot S_2 \cdot \rho_2 \sqsubseteq R \in \mathcal{L}, \textit{ then } \rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R. \quad (19)$$

Proof. The "only if" direction of the theorem can be easily shown using Definition 2 since $\rho \sqsubseteq S_1 \in \mathcal{R}$ and $S_1 \sqsubseteq^0_\mathcal{L} S_2$ imply $\rho \sqsubseteq_\mathcal{L} S_2$.

To show the "if" direction, we first prove the following strengthening of (19):

$$\text{if } \rho \sqsubseteq S_1 \in \mathcal{R},\ S_1 \sqsubseteq^0_\mathcal{L} S_2 \text{ and } \rho_1 \cdot S_2 \cdot \rho_2 \sqsubseteq_\mathcal{L} R, \text{ then } \rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R. \quad (20)$$

The proof of (20) is by induction on the derivation of $\rho_1 \cdot S_2 \cdot \rho_2 \sqsubseteq_\mathcal{L} R$.

E₀: $S_2 = R$ and $\rho_1 = \rho_2 = \epsilon$. Then (20) follows from (8).

E₁: $\rho_1 \cdot S_2 \cdot \rho_2 \sqsubseteq_\mathcal{L} S$ and $S \sqsubseteq R \in \mathcal{R}$. By the induction hypothesis $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} S$, from which $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R$ follows by **E₁**.

E′₁: Analogous to the case of **E₁**.

E₂: $\rho_1 \cdot S_2 \cdot \rho_2 = \sigma_1 \cdot \ldots \cdot \sigma_n$, $(\sigma_i \sqsubseteq_\mathcal{L} R_i)_{i=1}^n$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{R} \setminus \mathcal{L}$. Let k be such that S_2 occurs in σ_k, that is $\sigma_1 \cdot \ldots \cdot \sigma_{k-1} = \rho'_1$, $\sigma_k = \rho''_1 \cdot S_2 \cdot \rho''_2$, $\sigma_{k+1} \cdot \ldots \cdot \sigma_n = \rho'_2$ and $\rho_1 = \rho'_1 \cdot \rho''_1$, $\rho_2 = \rho''_2 \cdot \rho'_2$. By the induction hypothesis $\rho''_1 \cdot \rho \cdot \rho''_2 \sqsubseteq_\mathcal{L} R_k$, from which $\rho_1 \cdot \rho \cdot \rho_2 = \sigma_1 \cdot \ldots \cdot \sigma_{k-1} \cdot \rho''_1 \cdot \rho \cdot \rho''_2 \cdot \sigma_{k+1} \cdot \ldots \cdot \sigma_n \sqsubseteq_\mathcal{L} R$ follows by **E₂**.

E′₂: $\rho_1 \cdot S_2 \cdot \rho_2 = \sigma_1 \cdot T_2 \cdot \ldots \cdot T_n$, $\sigma_1 \sqsubseteq_\mathcal{L} R_1$, $(T_i \sqsubseteq^0_\mathcal{L} R_i)_{i=2}^n$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq R \in \mathcal{L}$. If S_2 occurs in σ_1, then this is analogous to the case of **E₂**. Otherwise, let k be such that $\sigma_1 \cdot T_2 \cdot \ldots \cdot T_{k-1} = \rho_1$, $T_k = S_2$ and $T_{k+1} \cdot \ldots \cdot T_n = \rho_2$. By (7) $S_1 \sqsubseteq^0_\mathcal{L} R_k$. By (19) applied to $\rho \sqsubseteq S_1 \in \mathcal{R}$, $S_1 \sqsubseteq^0_\mathcal{L} R_k$ and $R_1 \cdot \ldots \cdot R_n \sqsubseteq_\mathcal{L} R$ we obtain $R_1 \cdot \ldots \cdot R_{k-1} \cdot \rho \cdot R_{k+1} \cdot \ldots \cdot R_n \sqsubseteq_\mathcal{L} R$, from which $\rho_1 \cdot \rho \cdot \rho_2 = \sigma_1 \cdot T_2 \cdot \ldots \cdot T_{k-1} \cdot \rho \cdot T_{k+1} \cdot \ldots \cdot T_n \sqsubseteq_\mathcal{L} R$ follows by Lemma 3.

Having proved (20), condition (17) now follows by taking $S_1 = S_2 = S$ in (20) and using rule **E₀** to derive $S \sqsubseteq^0_\mathcal{L} S$. Therefore $\mathcal{L}$ is left-admissible for $\mathcal{R}$. □

Condition (19) in Theorem 2 can be checked in polynomial time in the size of $\mathcal{R}$. Indeed, there are only polynomially many possible instances of the precondition in (19). For every such precondition, the property $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_\mathcal{L} R$ can be checked in polynomial time by, e.g., applying Lemma 1: $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_{\mathcal{L}} R$ holds iff $C_0 \sqsubseteq_{\mathcal{L}} \exists R.C_n$ is derivable from $(C_{i-1} \sqsubseteq^{\text{o}}_{\mathcal{L}} \exists R_i.C_i)_{i=1}^n$, where $R_1 \cdot \ldots \cdot R_n = \rho_1 \cdot \rho \cdot \rho_2$.

Theorem 2 can help checking if a given set $\mathcal{L}$ is left-admissible, but does not explain how to find such a set without exhaustively checking al possible subsets of $\mathcal{R}$. The following sufficient condition will help us quickly find a suitable left-admissible set of RIAs in practice:

Theorem 3. *For a set of RIAs $\mathcal{R}$ let $\mathcal{L}(\mathcal{R})$ be the set of exactly those complex RIAs $\sigma \sqsubseteq R \in \mathcal{R}$ that satisfy the following condition for all ρ, ρ_1, ρ_2, S_1, S_2:*

$$\text{if } \rho \sqsubseteq S_1 \in \mathcal{R},\ S_1 \sqsubseteq^{\text{o}}_{\mathcal{R}} S_2 \text{ and } \rho_1 \cdot S_2 \cdot \rho_2 = \sigma, \text{ then } \rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_{\mathcal{R}} R. \qquad (21)$$

Then $\mathcal{L}(\mathcal{R})$ is left-admissible for $\mathcal{R}$.

Proof. Note that the relations $\sqsubseteq_{\mathcal{L}}$ are anti-monotonic in $\mathcal{L}$, that is for $\mathcal{L}_1 \subseteq \mathcal{L}_2$ we have $\sqsubseteq_{\mathcal{L}_1} \supseteq \sqsubseteq_{\mathcal{L}_2}$. Let $\mathcal{L} = \mathcal{L}(\mathcal{R})$. Since $\mathcal{L} \subseteq \mathcal{R}$, we have $\sqsubseteq_{\mathcal{L}} \supseteq \sqsubseteq_{\mathcal{R}}$, and, since $\sqsubseteq^{\text{o}}_{\mathcal{L}}$ does not depend on $\mathcal{L}$, we have $\sqsubseteq^{\text{o}}_{\mathcal{L}} = \sqsubseteq^{\text{o}}_{\mathcal{R}}$. Now it is easy to show that $\mathcal{L}$ satisfies (19): Suppose $\rho \sqsubseteq S_1 \in \mathcal{R}$, $S_1 \sqsubseteq^{\text{o}}_{\mathcal{L}} S_2$ and $\rho_1 \cdot S_2 \cdot \rho_2 \sqsubseteq R \in \mathcal{L}$. Then $\sqsubseteq^{\text{o}}_{\mathcal{L}} = \sqsubseteq^{\text{o}}_{\mathcal{R}}$ implies $S_1 \sqsubseteq^{\text{o}}_{\mathcal{R}} S_2$, so $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_{\mathcal{R}} R$ by (21). Then $\sqsubseteq_{\mathcal{L}} \supseteq \sqsubseteq_{\mathcal{R}}$ implies $\rho_1 \cdot \rho \cdot \rho_2 \sqsubseteq_{\mathcal{L}} R$, so (19) holds. Therefore $\mathcal{L} = \mathcal{L}(\mathcal{R})$ is left-admissible for $\mathcal{R}$ by Theorem 2. □

6 Experimental Evaluation

In this section we present the results of an experimental comparison of applying the calculi in Sections 2 and 3 to several commonly considered $\mathcal{EL}$ ontologies that contain complex RIAs, and discuss whether and to which extent our optimized treatment of RIAs can improve the performance of reasoning in practice.

To evaluate the proposed algorithms, we have implemented the calculi described in Sections 2 and 3 in a prototype Java-based reasoner ELK.[1] All experiments were conducted using Java 1.6 on a 2.5 GHz quad core CPU with 4GB RAM running Fedora 13 Linux.

Our test ontology suite includes GO,[2] FMA-lite,[3] and an OWL EL version of GALEN.[4] These ontologies contain only (left-admissible) RIAs of the form $R \sqsubseteq S$ and $R \cdot R \sqsubseteq R$. In order to test which proportion of complex RIAs in realistic ontologies is left-admissible, we additionally considered the two latest versions of GALEN,[5] namely GALEN7 and GALEN8, which contain RIAs of the form $R \sqsubseteq S$, $R \cdot S \sqsubseteq R$, and $S \cdot R \sqsubseteq R$. We reduced these ontologies to $\mathcal{ELR}$ by removing all axioms for role functionalities and role inverses and replacing all datatypes by fresh atomic concepts. It is worth noting that the RIAs in GALEN7 and GALEN8 do not satisfy the regularity restrictions of OWL 2 [7]

[1] http://code.google.com/p/elk-reasoner/
[2] obtained from http://lat.inf.tu-dresden.de/~meng/toyont.html
[3] obtained from http://www.bioontology.org/wiki/index.php/FMAInOwl
[4] obtained from http://condor-reasoner.googlecode.com/
[5] obtained from http://www.opengalen.org/sources/sources.html

Table 4. Ontology metrics and experimental results

	GO	FMA-lite	OWL GALEN	GALEN7	GALEN8
Number of normalized input axioms					
$A \sqsubseteq B$	28,896	121,708	71,366	92,749	588,806
$\bigsqcap_{i=1}^{n} A_i \sqsubseteq B, n \geq 2$	0	0	11,561	12,097	122,527
$A \sqsubseteq \exists R.B$	1,796	12,355	14,115	15,105	106,065
$\exists R.A \sqsubseteq B$	0	0	7,549	7,973	93,241
$R \sqsubseteq S$	0	3	958	972	996
$R_1 \cdot R_2 \sqsubseteq R_3 \in \mathcal{R}$	1	1	58	385	385
$R_1 \cdot R_2 \sqsubseteq R_3 \in \mathcal{L}$	1	1	58	183	183
Number of derived axioms					
$A \sqsubseteq B$	206,205	1,035,527	1,119,636	1,770,895	11,462,383
$A \sqsubseteq \exists R.B$	33,985	867,209	2,282,471	3,299,376	24,998,147
Number of rule applications					
$\mathbf{C_0}/\mathbf{L_0}$	19,468	78,977	25,963	30,534	202,664
$\mathbf{C_1}/\mathbf{L_1}$	241,834	958,754	1,396,379	2,917,625	15,900,712
$\mathbf{C_2}/\mathbf{L_2}$	0	0	259,654	372,780	2,639,688
$\mathbf{C_3}/\mathbf{L_3}$	21,994	114,724	339,880	446,980	3,780,076
$\mathbf{C_4}/\mathbf{L_4}$	0	0	949,148	1,316,768	17,217,292
$\mathbf{C_5}/\mathbf{L_5}+\mathbf{L'_5}$	0	96,891	2,023,828	2,903,821	21,988,137
$\mathbf{C_6}$	34,753	5,807,992	275,248	1,264,208	11,867,857
$\mathbf{L_6}+\mathbf{L'_6}$	19,756	1,186,733	216,982	1,087,328	8,728,711

and, for this reason, no OWL reasoner can handle them in the unreduced form. We have excluded SNOMED CT from our experiments for the reason that the only complex RIA it contains is redundant for classification in the sense that rule $\mathbf{C_6}$ is never applied on this ontology.

In our experiments, we first normalized all test ontologies using structural transformation, and applied Theorem 3 to identify left-admissible sets of RIAs. Table 4 presents statistics on the number of axioms of each type and the number of left-admissible RIAs for each of the tested ontologies. For GALEN7 and GALEN8, which contain identical complex RIAs, we found a left-admissible subset containing 183 out of the total 385 complex RIAs. In this case, we additionally checked that adding any one of the remaining complex RIAs to the previously found 183 violates the conditions of Theorem 2, showing that the left-admissible subset of RIAs we found is maximal. For the remaining ontologies, the full set of RIAs is left-admissible since transitivity axioms are the only kind of complex RIA. In general, the computation of left-admissible RIAs had no relevant impact on overall performance, running in less than 0.5 seconds in all cases.

For each of the tested ontologies, we computed the saturation under the inference rules of Table 1 and Table 2. Table 4 presents the total number of different conclusions of each type, and the total number of inferences for each rule. In accordance with Theorem 1, both approaches produce the same conclusions. For this reason, the number of applications of each rule $\mathbf{C_0}$–$\mathbf{C_5}$ coincides with the corresponding number for rules $\mathsf{L_0}$–$\mathsf{L'_5}$. Differences between the two approaches are found in the number of applications of $\mathsf{C_6}$ on the one hand, and the combined number of applications of $\mathsf{L_6}$ and $\mathsf{L'_6}$ on the other hand. As can be seen from the results, the effect of our optimization strongly depends on the input ontology, with the largest relative reductions obtained for FMA-lite and GO, and less significant reductions for all versions of GALEN.

Although the reduction in the number of rule application is significant for FMA-lite and GO, this, surprisingly, did not translate to a significant reduction in the running time for our prototype implementation. For FMA-lite, for example, the running time is reduced just from 7.2 to 6.1 seconds (15.3%), which is less than expected for more than 65% reduction in the number of inferences. For other ontologies the reduction in the running time was even less measurable.

One possible explanation for this effect is that a rule application producing a new consequence costs more than a rule application producing a previously derived consequence because the first requires a (relatively expensive) memory allocation. Since our optimized procedure derives exactly the same conclusions, it reduces only the number of inferences of the second kind. Nevertheless, our optimization can give improvement in some cases and should not be difficult to implement (at least for transitivity) in any reasoner based on the original $\mathcal{EL}$ calculus [1], such as in CEL/jCEL [2] or Snorocket [5].

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05). pp. 364–369. Professional Book Center (2005)
2. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL—a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06). LNCS, vol. 4130, pp. 287–291. Springer (2006)
3. Delaitre, V., Kazakov, Y.: Classifying $\mathcal{ELH}$ ontologies in SQL databases. In: Patel-Schneider, P.F., Hoekstra, R. (eds.) Proc. OWLED 2009 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org (2009)
4. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh [9], to appear
5. Lawley, M.J., Bousquet, C.: Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In: Taylor, K., Meyer, T., Orgun, M. (eds.) Proc. 6th Australasian Ontology Workshop (IAOA'10). Conferences in Research and Practice in Information Technology, vol. 122, pp. 45–49. Australian Computer Society Inc. (2010)
6. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-profiles/
7. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-syntax/
8. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Walsh [9], to appear
9. Walsh, T. (ed.): Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'11). IJCAI (2011)

On Prototypes for Winslett's Semantics of DL-Lite ABox Evolution

Evgeny Kharlamov, and Dmitriy Zheleznyakov

KRDB Research Centre, Free University of Bozen-Bolzano, Italy
`last_name@inf.unibz.it`

Abstract. Evolution of Knowledge Bases expressed in Description Logics (DLs) proved its importance. Most studies on evolution in DLs have focused on model-based approaches to evolution semantics and in particular on Winslett's semantics (WS). It was understood that evolution under WS even in tractable DLs, such as *DL-Lite*, suffers from inexpressibility, i.e., the result of evolution cannot be expressed in the same logics. In this work we show which combination of *DL-Lite* logical constructs is responsible for the inexpressibility and explain reasons for such a behaviour. We present novel techniques, based on what we called prototypes, to capture Winslett's evolution in FO[2] for *DL-Lite*$_\mathcal{R}$. We also discuss which fragments of *DL-Lite*$_\mathcal{R}$ are closed under evolution.

1 Introduction

Description Logics (DLs) provide excellent mechanisms for representing structured knowledge by means of Knowledge Bases (KBs) $\mathcal{K}$ that are composed of two components: TBox (describes intensional or general knowledge about an application domain) and ABox (describes facts about individual objects). DLs constitute the foundations for various dialects of OWL, the Semantic Web ontology language.

Traditionally DLs have been used for modeling *static* and structural aspects of application domains [1]. Recently, the scope of KBs has broadened, and they are now used also for providing support in the maintenance and *evolution* phase of information systems. This makes it necessary to study *evolution of Knowledge Bases* [2], where the goal is to incorporate a new knowledge $\mathcal{N}$ into an existing KB $\mathcal{K}$ so as to take into account changes that occur in the underlying application domain. In general, $\mathcal{N}$ is represented by a set of formulas denoting those properties that should be true after $\mathcal{K}$ has evolved, and the result of evolution, denoted $\mathcal{K} \diamond \mathcal{N}$, is also intended to be a set of formulas. In the case where $\mathcal{N}$ interacts with $\mathcal{K}$ in an undesirable way, e.g., by causing the KB or relevant parts of it to become unsatisfiable, $\mathcal{N}$ cannot simply be added to the KB. Instead, suitable changes need to be made in $\mathcal{K}$ so as to avoid this undesirable interaction, e.g., by deleting parts of $\mathcal{K}$ conflicting with $\mathcal{N}$. Different choices for changes are possible, corresponding to different approaches to *semantics* for KB evolution [3,4,5].

One approach to evolution semantics that proved its importance is *Winslett's semantics* (WS) [6], which is an *update* semantics in terms of Katsumo and Mendelzon [4], and was originally proposed for propositional theories. Under this semantics the result of evolution $\mathcal{K} \diamond \mathcal{N}$ is a *set of models* of $\mathcal{N}$ that are minimally distanced from models of $\mathcal{K}$, where the distance is based on symmetric difference between models (see Section 3 for details). Since the result of evolution $\mathcal{K} \diamond \mathcal{N}$ is a set of models, while $\mathcal{K}$ and $\mathcal{N}$ are logical theories, it is desirable to represent $\mathcal{K} \diamond \mathcal{N}$ as a logical theory using the same language as for $\mathcal{K}$ and $\mathcal{N}$. Thus, looking for representations of $\mathcal{K} \diamond \mathcal{N}$ is the main challenge in a study of evolution under WS. When $\mathcal{K}$ and $\mathcal{N}$ are propositional theories, representing $\mathcal{K} \diamond \mathcal{N}$ is well understood [5], while it becomes dramatically more complicated as soon as $\mathcal{K}$ and $\mathcal{N}$ are first-order, e.g., DL KBs [7].

In this work we study how WS can be applied to evolution of KBs under the following two assumptions. First, we assume that both $\mathcal{K}$ and $\mathcal{N}$ are written in a language of the *DL-Lite* family [8]. The focus on *DL-Lite* is not surprising since *DL-Lite* is tightly connected with conceptual data models and it is the basis of OWL 2 QL, a tractable OWL 2 profile.Second, we assume that $\mathcal{N}$ is a new ABox and the TBox of $\mathcal{K}$ should remain the same after the evolution. That is, we study a so-called *ABox evolution*. ABox evolution is important for areas, e.g., bioinformatics, where the structural knowledge TBox is well crafted and stable, while ABox facts about specific individuals may get changed, or/and new facts can be inserted in the ABox. These ABox changes should be reflected in KBs in a way that the TBox is not affected.

There are several works on WS for both *DL-Lite* and more expressive DLs. Liu, Lutz, Milicic, and Wolter studied Winslett's evolution in expressive DLs [7], for KBs with empty TBoxes. Most of DLs they considered are not closed under WS and in order to close these logics they used "@" operator. Poggi, Lembo, De Giacomo, Lenzerini, and Rosati applied WS to *DL-Lite* [9] and proposed an algorithm to compute the result of evolution. It turned out that their algorithm is wrong, i.e. it is neither sound, nor complete [10]. Actually, such an algorithm cannot exist since Calvanese, Kharlamov, Nutt, and Zheleznyakov showed that, e.g., *DL-Lite*$_{\mathcal{FR}}$ is not closed under WS of evolution [11], that is, there are $\mathcal{K}$ and $\mathcal{N}$ such that $\mathcal{K} \diamond \mathcal{N}$ is not axiomatizable in this family. Recently [12] we introduced *prototypes*, which are in a way generalization of the notion of canonical model, and proposed a way to capture some fragments of *DL-Lite* in $\mathsf{FO}[2]$, a fragment of first-order logic that uses two variables only.

Current work extends the preliminary results of [12]. Our goals here are

(i) to clarify our prototype-based techniques which was only sketched in [12],
(ii) to extend the techniques to wider *DL-Lite* fragments,
(iii) to gain a better understanding on which fragments of *DL-Lite* are closed under WS and how to approximate evolution results in *DL-Lite*.

We would also like to promote prototypes since we believe they are an useful tool to study evolution of ontologies and might be not only of *DL-Lite* ones.

In Sections 2 and 3 we define *DL-Lite*$_{\mathcal{R}}$ and ABox evolution under WS. In Section 4 we give an intuition of our approach to capture WS of evolution for *DL-Lite*$_{\mathcal{R}}$ KBs using prototypes and $\mathsf{FO}[2]$ theories. In Sections 5 and 6 we formalize the approach. Finally, we discuss properties and approximation of these theories.

2 DL-Lite$_{\mathcal{R}}$

We introduce some basic notions of DLs (see [1] for more details). We consider a logic *DL-Lite*$_{\mathcal{R}}$ of *DL-Lite* family of DLs [8,13]. *DL-Lite*$_{\mathcal{R}}$ has the following constructs for (complex) *concepts* and *roles*: *(i)* $B ::= A \mid \exists R$, *(ii)* $C ::= B \mid \neg B$, *(iii)* $R ::= P \mid P^{-}$, where A and P stand for an *atomic concept* and *role*, respectively, which are just names. A DL *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is compound of two sets of *assertions*: *TBox* $\mathcal{T}$, and *ABox* $\mathcal{A}$. *DL-Lite*$_\mathcal{R}$ TBox assertions are *concept inclusion assertions* of the form $B \sqsubseteq C$ and *role inclusion assertions* $R_1 \sqsubseteq R_2$, while ABox assertions are *membership assertions* of the form $A(a)$, $\neg A(a)$, and $R(a, b)$. The *active domain* of $\mathcal{K}$, denoted *adom*$(\mathcal{K})$, is the set of all constants occurring in $\mathcal{K}$. The *DL-Lite* family has nice computational properties, for example, KB satisfiability has polynomial-time complexity in the size of the TBox and logarithmic-space in the size of the ABox [14,15].

The semantics of DL-Lite KBs is given in the standard way: using first order *interpretations* $\mathcal{I}$, all over the same countable domain Δ. We assume that Δ contains the constants and $c^\mathcal{I} = c$, i.e., we adopt *standard names*. Alternatively, we view interpretations as sets of atoms: $A(a) \in \mathcal{I}$ iff $a \in A^\mathcal{I}$ and $P(a, b) \in \mathcal{I}$ iff $(a, b) \in P^\mathcal{I}$.

Definitions of $\mathcal{I}$ being a *model* of an ABox or a TBox assertion F, denoted $\mathcal{I} \models F$, and a KB $\mathcal{K}$, denoted $\mathcal{I} \models K$, are standard, as well as the notion of *satisfiability*. We use *Mod*$(\mathcal{K})$ to denote the set of all models of $\mathcal{K}$. We use entailment on KBs $\mathcal{K} \models \mathcal{K}'$ in the standard sense. An ABox $\mathcal{A}$ $\mathcal{T}$*-entails* an ABox $\mathcal{A}'$, denoted $\mathcal{A} \models_\mathcal{T} \mathcal{A}'$, if $\mathcal{T} \cup \mathcal{A} \models \mathcal{A}'$, and $\mathcal{A}$ is $\mathcal{T}$*-equivalent* to $\mathcal{A}'$, denoted $\mathcal{A} \equiv_\mathcal{T} \mathcal{A}'$, if $\mathcal{A} \models_\mathcal{T} \mathcal{A}'$ and $\mathcal{A}' \models_\mathcal{T} \mathcal{A}$.

The deductive *closure of a TBox* $\mathcal{T}$, denoted *cl*$(\mathcal{T})$, is the set of all TBox assertions F such that $\mathcal{T} \models F$. For satisfiable KBs $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a *full closure of* $\mathcal{A}$ (wrt $\mathcal{T}$), *fcl*$_\mathcal{T}(\mathcal{A})$, is the set of all membership assertions f (both positive and negative) over *adom*$(\mathcal{K})$ such that $\mathcal{A} \models_\mathcal{T} f$. Clearly, in *DL-Lite*$_\mathcal{R}$ both *cl*$(\mathcal{T})$ and *cl*$_\mathcal{T}(\mathcal{A})$ are computable in time quadratic in, respectively, $|\mathcal{T}|$, i.e., the number of assertions of $\mathcal{T}$, and $|\mathcal{T} \cup \mathcal{A}|$. For the ease of exhibition and wlg we assume that all TBoxes and ABoxes are closed.

A *homomorphism* h from a model $\mathcal{I}$ to a model $\mathcal{J}$ is a structure-preserving mapping from Δ to Δ satisfying: *(i)* $h(a) = a$ for every constant a; *(ii)* if $\alpha \in A^\mathcal{I}$ (resp., $(\alpha,\beta) \in P^\mathcal{I}$), then $h(\alpha) \in A^\mathcal{J}$ (resp., $(h(\alpha), h(\beta)) \in P^\mathcal{J}$) for every A (resp., P). We write $\mathcal{I} \hookrightarrow \mathcal{J}$ if there is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$. A canonical model $\mathcal{I}$ of $\mathcal{K}$, denoted as $\mathcal{I}^{can}_\mathcal{K}$ or just $\mathcal{I}^{can}$ when $\mathcal{K}$ is clear from the context, is a model of $\mathcal{K}$ which can be homomorphically embedded in every model of $\mathcal{K}$ [8].

3 Winslett's Semantics for Evolution of Knowledge Bases

We start with ABox evolution of single models under Winslett's semantics. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a *DL-Lite*$_\mathcal{R}$ KB, $\mathcal{I}$ a model of $\mathcal{K}$, and $\mathcal{N}$ a new ABox satisfiable with $\mathcal{T}$. Evolution of a model $\mathcal{I}$ of $\mathcal{K}$ is based on the symmetric difference $\ominus$: $S_1 \ominus S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$, and defined as follows. The (result of) *evolution of* $\mathcal{I}$ *with* $\mathcal{N}$ under Winslett's semantics (WS) [9], denoted $\mathcal{I} \diamond \mathcal{N}$, is the set of models $\mathcal{J}$ such that:

(i) $\mathcal{J} \in$ *Mod*$(\mathcal{T} \cup \mathcal{N})$, and

(ii) there is no model $\mathcal{J}' \in$ *Mod*$(\mathcal{T} \cup \mathcal{N})$ satisfying $\mathcal{I} \ominus \mathcal{J}' \subsetneq \mathcal{I} \ominus \mathcal{J}$

Note that in Case *(i)* we have *Mod* of both $\mathcal{T}$ and $\mathcal{N}$, which means that the evolution preserves both the old TBox and the new knowledge. Case *(ii)* guarantees the principle of minimal change [5]. We extend the definition to KBs:

The result of *evolution of* $\mathcal{K}$ *with* $\mathcal{N}$ under WS, denoted $\mathcal{K} \diamond \mathcal{N}$, is the following set of models:

$$\mathcal{K} \diamond \mathcal{N} = \cup_{\mathcal{I} \in Mod(\mathcal{K})} \mathcal{I} \diamond \mathcal{N}.$$

In terms of [10], WS corresponds to $\mathcal{L}^a_\subseteq$ semantics, i.e., local model-based semantics based on atoms and set inclusion.

The input for evolution is two finite syntactic objects: a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a new information $\mathcal{N}$, while the output $\mathcal{K} \diamond \mathcal{N}$ is a set of models, which is an infinite object for *DL-Lite*$_\mathcal{R}$. Indeed, $\mathcal{K} \diamond \mathcal{N}$ is in general infinite. One can easily come up with examples where $\mathcal{K} \diamond \mathcal{N}$ has an infinite number of infinite models. These observations imply that storing $\mathcal{K} \diamond \mathcal{N}$ is infeasible and in practice one would like to represent the evolution as a KB $\mathcal{K}'$. Moreover, one would like to stay within the same formalism and express $\mathcal{K}'$ in *DL-Lite*$_\mathcal{R}$. Formally, we say that a logic $\mathcal{L}$ is *closed* under Winslett's evolution if for every $\mathcal{K}$ and $\mathcal{N}$ in $\mathcal{L}$, the result of evolution $\mathcal{K} \diamond \mathcal{N}$ is expressible in $\mathcal{L}$, that is, there is a KB $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ in $\mathcal{L}$ such that $\mathit{Mod}(\mathcal{K}') = \mathcal{K} \diamond \mathcal{N}$.

Example 1. Consider the following *DL-Lite* KB $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$ and $\mathcal{N}_1 = \{C(b)\}$:

$$\mathcal{T}_1 = \{A \sqsubseteq \exists R, \exists R^- \sqsubseteq \neg C\}; \quad \mathcal{A}_1 = \{A(a), C(e), C(d), R(a, b)\}.$$

Consider the following model $\mathcal{I}$ of $\mathcal{K}_1$:

$$\mathcal{I}: \quad A^\mathcal{I} = \{a, x\}, \quad C^\mathcal{I} = \{d, e\}, \quad R^\mathcal{I} = \{(a, b), (x, b)\},$$

where $x \in \Delta \setminus \mathit{adom}$. The following models belong to $\mathcal{I} \diamond \mathcal{N}_1$:

$$\begin{array}{llll} \mathcal{J}_0: & A^\mathcal{I} = \emptyset, & C^\mathcal{I} = \{d, e, b\}, & R^\mathcal{I} = \emptyset, \\ \mathcal{J}_1: & A^\mathcal{I} = \{x\}, & C^\mathcal{I} = \{e, b\}, & R^\mathcal{I} = \{(x, d)\}, \\ \mathcal{J}_2: & A^\mathcal{I} = \{x\}, & C^\mathcal{I} = \{d, b\}, & R^\mathcal{I} = \{(x, e)\}. \end{array}$$

Indeed, all the models satisfy $\mathcal{N}_1$ and $\mathcal{T}_1$. To see that they are in $\mathcal{I} \diamond \mathcal{N}_1$ observe that every model $\mathcal{J}(I) \in (\mathcal{I} \diamond \mathcal{N}_1)$ can be obtained from $\mathcal{I}$ by making modifications that guarantee that $\mathcal{J}(I) \models (\mathcal{N}_1 \cup \mathcal{T}_1)$ and that the distance between $\mathcal{I}$ and $\mathcal{J}(I)$ is minimal. What are these modifications? Since in every $\mathcal{J}(I)$ the new assertion $C(b)$ holds and $(C \sqsubseteq \neg \exists R^-) \in \mathcal{T}_1$, there should be no R-atoms with b-fillers at the second coordinate in $\mathcal{J}(\mathcal{I})$. Hence, the necessary modifications of $\mathcal{I}$ are either to drop (some of) the R-atoms $R(a, b)$ and $R(x, b)$, or to modify (some of) them, by substituting the b-fillers with another ones, while keeping the elements a and x on the first position. The model $\mathcal{J}_0$ corresponds to the case when both R-atoms are dropped, while in $\mathcal{J}_1$ and $\mathcal{J}_2$ only $R(a, b)$ is dropped and $R(x, b)$ is modified to $R(x, d)$ and $R(x, e)$, respectively. Note that the modification in $R(x, b)$ leads to a further change in the interpretation of C in both $\mathcal{J}_1$ and $\mathcal{J}_2$, namely, $C(d)$ and $C(e)$ should be dropped, respectively. ■

4 Prototypes for Winslett's Semantics

We first present a general discussion on issues with capturing WS in *DL-Lite*, then give an intuition of our approach for capturing it in FO[2], and finally give an example of how the approach works. In the next section we formalize the approach.

ABox Evolution of a *DL-Lite* KB $\mathcal{K}$ with an ABox $\mathcal{N}$ is the set of models $\mathcal{K} \diamond \mathcal{N}$ that may not have a canonical one [12]. This immediately yields that $\mathcal{K} \diamond \mathcal{N}$ cannot be described (aka axiomatized) in any language of the *DL-Lite* family.

Example 2. We now illustrate the lack of canonical models in $\mathcal{K}_1 \diamond \mathcal{N}_1$ from Example 1. One can verify that any model $\mathcal{J}_{\mathit{can}}$ that can be homomorphically embedded into $\mathcal{J}_0$, $\mathcal{J}_1$, and $\mathcal{J}_2$ is such that $A^{\mathcal{J}_{\mathit{can}}} = R^{\mathcal{J}_{\mathit{can}}} = \emptyset$, and $e, d \notin C^{\mathcal{J}_{\mathit{can}}}$. It is easy to check that such a model does not belong to $\mathcal{K}_1 \diamond \mathcal{N}_1$. Hence, there is no canonical model in $\mathcal{K} \diamond \mathcal{N}$ and it is inexpressible in *DL-Lite*. ■

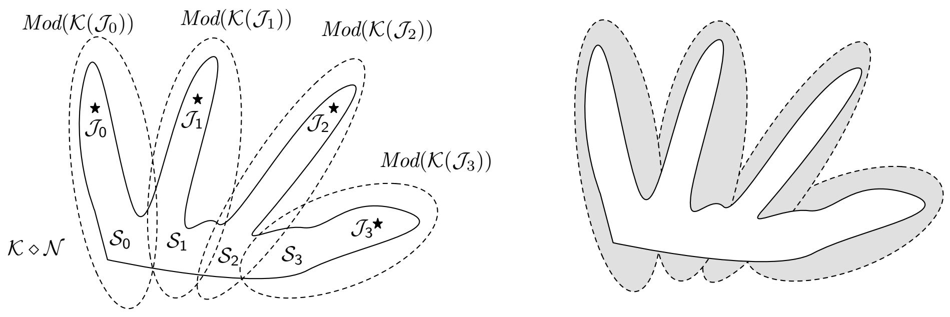

$Mod(\mathcal{K}(\mathcal{J}_0))$
$Mod(\mathcal{K}(\mathcal{J}_1))$
$Mod(\mathcal{K}(\mathcal{J}_2))$
$Mod(\mathcal{K}(\mathcal{J}_3))$
$\mathcal{J}_0$
$\mathcal{J}_1$
$\mathcal{J}_2$
$\mathcal{J}_3$
$\mathcal{K} \diamond \mathcal{N}$
$\mathcal{S}_0$
$\mathcal{S}_1$
$\mathcal{S}_2$
$\mathcal{S}_3$

Fig. 1. Graphical representation of our approach to capture the result of evolution under WS.

A closer look at sets $\mathcal{K} \diamond \mathcal{N}$ for different $\mathcal{K}$ and $\mathcal{N}$ gave a surprising outcome: all of them satisfy the following property.

Theorem 3. *$\mathcal{K} \diamond \mathcal{N}$ can be divided (but in general not partitioned) into finitely many subsets $\mathcal{S}_0, \dots, \mathcal{S}_n$ of models, where each $\mathcal{S}_i$ has a canonical model $\mathcal{J}_i$. Each of these canonical models is a minimal element in $\mathcal{K} \diamond \mathcal{N}$ wrt homomorphisms.*

We called these $\mathcal{J}_i$s *prototypes* [12]. Thus, capturing $\mathcal{K} \diamond \mathcal{N}$ in some logics boils down to *(i)* capturing each $\mathcal{S}_i$ with some theory $\mathcal{K}_{\mathcal{S}_i}$ and *(ii)* taking the disjunction across all $\mathcal{K}_{\mathcal{S}_i}$. This will give the desired theory $\mathcal{K}' = \mathcal{K}_{\mathcal{S}_1} \vee \cdots \vee \mathcal{K}_{\mathcal{S}_n}$ that captures $\mathcal{K} \diamond \mathcal{N}$. Unfortunately, some of $\mathcal{K}_{\mathcal{S}_i}$ are not *DL-Lite* theories (while they are $\mathsf{FO}[2]$ theories, see Section 5 for details).

We construct $\mathcal{K}'$ in two steps. First, we construct *DL-Lite*$_\mathcal{R}$ KBs $\mathcal{K}(\mathcal{J}_i)$ for each $\mathcal{J}_i$ such that $\mathcal{K}(\mathcal{J}_i)$ is a sound approximations of $\mathcal{S}_i$s, that is, $\mathcal{S}_i \subseteq Mod(\mathcal{K}(\mathcal{J}_i))$. Second, based on $\mathcal{K}$ and $\mathcal{N}$, we construct an $\mathsf{FO}[2]$ formula Ψ, which cancels out all the models in $Mod(\mathcal{K}(\mathcal{J}_i)) \setminus \mathcal{S}_i$, that is, $\mathcal{K}_{\mathcal{S}_0} \vee \cdots \vee \mathcal{K}_{\mathcal{S}_n} = \Psi \wedge (\mathcal{K}(\mathcal{J}_0) \vee \cdots \vee \mathcal{K}(\mathcal{J}_n))$.

To get a better intuition on our approach, consider Figure 1, where the result of evolution $\mathcal{K} \diamond \mathcal{N}$ is depicted as the figure with solid-line borders (each point within the figure is a model in $\mathcal{K} \diamond \mathcal{N}$). Assume that $\mathcal{K} \diamond \mathcal{N}$ can be divided in four subsets $\mathcal{S}_0, \dots, \mathcal{S}_3$. To emphasize this fact, $\mathcal{K} \diamond \mathcal{N}$ looks similar to a hand with four fingers, where each finger represents an $\mathcal{S}_i$. Consider the left part of Figure 1. Each of $\mathcal{S}_i$s has a canonical model depicted as a star. Using *DL-Lite*$_\mathcal{R}$, we can provide KBs $\mathcal{K}(\mathcal{J}_0), \dots, \mathcal{K}(\mathcal{J}_3)$ that are sound approximation of corresponding $\mathcal{S}_i$s. We depict the models $Mod(\mathcal{K}(\mathcal{J}_i))$ as ovals with dashed-line boarders. Consider the right part of Figure 1. In this figure we depict in grey the models $Mod(\mathcal{K}(\mathcal{J}_i)) \setminus \mathcal{S}_i$ that are cut off by Ψ.

Before proceeding to the next section where we formalize our approach, we introduce prototypes formally.

Definition 4. *Let $\mathcal{K}$ be a DL-Lite$_\mathcal{R}$ KB and $\mathcal{N}$ be an ABox. A* prototypal set for $\mathcal{K} \diamond \mathcal{N}$ *is a minimal subset $\mathfrak{J} = \{\mathcal{J}_0, \dots, \mathcal{J}_n\}$ of $\mathcal{K} \diamond \mathcal{N}$ satisfying the property:*

$$\textit{for every } \mathcal{J} \in \mathcal{K} \diamond \mathcal{N} \textit{ there is } \mathcal{J}_i \in \mathfrak{J} \textit{ such that } \mathcal{J}_i \hookrightarrow \mathcal{J}.$$

We call every $\mathcal{J}_i \in \mathfrak{J}$ a *prototype for $\mathcal{K} \diamond \mathcal{N}$*. Note that prototypes generalize canonical models in the sense that every set of models with a canonical one, say $Mod(\mathcal{K})$ for a *DL-Lite*$_\mathcal{R}$ KB $\mathcal{K}$, has a prototype, which is exactly the canonical model.

5 Computing Winslett's Semantics When No Roles Interact

We first discuss some of the reasons of WS inexpressibility in our examples and *DL-Lite*$_\mathcal{R}$.

```
BZP(K,N)
1. $\mathcal{J}_0 := Align(\mathcal{I}^{can}, \mathcal{N}) \cup \mathcal{N}$, where $\mathcal{I}^{can}$ is the canonical model of $\mathcal{K}$.
2. For each $R(a,b) \in AA(\mathcal{K},\mathcal{N})$, do $\mathcal{J}_0 := \mathcal{J}_0 \setminus \{R(a,b)\}$,
       if there is no $R(a,\beta) \in \mathcal{A} \setminus AA(\mathcal{K},\mathcal{N})$ do $\mathcal{J}_0 := \mathcal{J}_0 \setminus root^{at}_{\mathcal{T}}(\exists R(a))$.
3. Return $\mathcal{J}_0$.
```

Fig. 2. The procedure of building zero-prototype

Dual-Affection of Roles. As we discussed in the previous section and illustrated in Example 1, sets of models $\mathcal{K} \diamond \mathcal{N}$ that result from Winslett's evolution do not have canonical models. We now give an intuition *why* in $\mathcal{K} \diamond \mathcal{N}$ canonical models are missing. Observe that in Example 1 the role R is affected by the old TBox $\mathcal{T}_1$ as follows:

(i) $\mathcal{T}_1$ *places* (i.e., enforces the *existence* of) R-atoms in the evolution result, and on *one* of coordinates of these R-atoms, there are constants from specific sets, e.g., $A \sqsubseteq \exists R$ of $\mathcal{T}_1$ enforces R-atoms with constants from A on the first coordinate, and

(ii) $\mathcal{T}_1$ *forbids* R-atoms in $\mathcal{K}_1 \diamond \mathcal{N}_1$ with specific constants on the *other* coordinate, e.g., $\exists R^- \sqsubseteq \neg C$ forbids R-atoms with C-constants on the second coordinate.

Due to this *dual-affection* (both positive and negative) of the role R in $\mathcal{T}_1$, we were able to provide an ABox $\mathcal{A}_1$ and $\mathcal{N}_1$, which together triggered the case analyses of modifications on the model $\mathcal{I}$, that is, $\mathcal{A}_1$ and $\mathcal{N}_1$ were *triggers* for R. Existence of such an affected R and triggers $\mathcal{A}_1$ and $\mathcal{N}_1$ made $\mathcal{K}_1 \diamond \mathcal{N}_1$ inexpressible in *DL-Lite*$_\mathcal{R}$. Therefore, we now learn how to detect dually-affected roles in TBoxes and how to understand whether these roles are triggered by an ABox and a new (ABox) information.

Formally, let $\mathcal{T}$ be a TBox, a role R is *dually-affected* in $\mathcal{T}$ if for some concepts A and B it holds that $\mathcal{T} \models A \sqsubseteq \exists R$ and $\mathcal{T} \models \exists R^- \sqsubseteq \neg B$. Let $\mathcal{N}$ be an ABox satisfiable with $\mathcal{T}$, then a dually-affected role R is *triggered* by $\mathcal{N}$ if there is a concept B such that $\mathcal{T} \models \exists R^- \sqsubseteq \neg B$ and $\mathcal{N} \models_\mathcal{T} B(b)$ for some constant b. The set *TR*$(\mathcal{T},\mathcal{N})$ (or simply *TR*) is the set of all roles (dually-affected in $\mathcal{T}$) that are triggered by $\mathcal{N}$.

Description Logics DL-Lite$^I_\mathcal{R}$. We now show a restriction of *DL-Lite*$_\mathcal{R}$ for which we later present an algorithm to capture WS using prototypal set. *DL-Lite*$^I_\mathcal{R}$ (where I stands for (mutual) *independence* of roles) is a restriction of *DL-Lite*$_\mathcal{R}$ in which TBoxes $\mathcal{T}$ satisfy: for any two roles R and R', $\mathcal{T} \not\models \exists R \sqsubseteq \exists R'$ and $\mathcal{T} \not\models \exists R \sqsubseteq \neg \exists R'$. That is, we forbid direct role interaction (subsumption and disjointness) between role projections. Some interaction is still possible: role projections may contain the same concept. This restriction allows us to analyze evolution affecting roles independently for every role.

Components for Computation. We now introduce several notions and notations that we further use in the description of our algorithm. An *alignment* of a model I with N, denoted $Align(\mathcal{I},\mathcal{N})$, is the interpretation:

$$Align(\mathcal{I},\mathcal{N}) = \{f \mid f \in \mathcal{I} \text{ and } f \text{ is satisfiable with } \mathcal{N}\}.$$

An auxiliary set of atoms *AA* (*Auxiliary Atoms*) that, due to evolution, should be deleted from the original KB and have some extra condition on the first coordinate is:

$$AA(\mathcal{T},\mathcal{A},\mathcal{N}) = \{R(a,b) \in fcl_\mathcal{T}(\mathcal{A}) \mid \mathcal{T} \models A \sqsubseteq \exists R, \mathcal{A} \models_\mathcal{T} A(a), \mathcal{N} \models_\mathcal{T} \neg\exists R^-(b)\}.$$

For the set *TR* we define the set of *forbidden atoms* $\mathtt{FA}[\mathcal{T},\mathcal{A},\mathcal{N}](R_i)$ of the original ABox as:

$$\{D(c) \in fcl_\mathcal{T}(\mathcal{A}) \mid \exists R_i^-(c) \wedge D(c) \models_\mathcal{T} \bot, \mathcal{N} \not\models_\mathcal{T} D(c), \text{ and } \mathcal{N} \not\models_\mathcal{T} \neg D(c)\}.$$

$BP(\mathcal{K}, \mathcal{N}, \mathcal{J}_0)$

```
1. $\mathfrak{J} := \{\mathcal{J}_0\}$.
2. For each subset $\mathcal{D} = \{D_1(c_1), \ldots, D_k(c_k)\} \subseteq \mathtt{FA}$ do
   for each $\mathcal{R} = (R_{i_1}, \ldots, R_{i_k})$ such that $D_j(c_j) \in \mathtt{FA}(R_{i_j})$ for $j = 1, \ldots, k$ do
   for each $\mathcal{B} = (A_{i_1}, \ldots, A_{i_k})$ such that $A_j \in SC(R_j)$ do
   $\mathcal{J}[\mathcal{D}, \mathcal{R}, \mathcal{B}] := \left[\mathcal{J}_0 \setminus \bigcup_{i=1}^{k} root_{\mathcal{T}}(D_i(c_i))\right] \cup \bigcup_{i=1}^{k} \left[fcl_{\mathcal{T}}(R_i'(x_i, c_i)) \cup \{A_{R_i'}(x_i)\}\right]$,
   where all $x_i$'s are different constants from $\Delta \setminus adom(\mathcal{K})$, fresh for $\mathcal{I}^{can}$.
   $\mathfrak{J} := \mathfrak{J} \cup \{\mathcal{J}[\mathcal{D}, \mathcal{R}, \mathcal{B}]\}$.
3. Return $\mathfrak{J}$.
```

Fig. 3. The procedure of building prototypes in *DL-Lite*$_{\mathcal{R}}^{I}$ based on the zero prototype $\mathcal{J}_0$

Consequently, the set of forbidden atoms for the entire KB $(\mathcal{T}, \mathcal{A})$ and $\mathcal{N}$ is

$$\mathtt{FA}(\mathcal{T}, \mathcal{A}, \mathcal{N}) = \cup_{R_i \in TR} \mathtt{FA}(\mathcal{T}, \mathcal{A}, \mathcal{N})(R_i).$$

In the following we omit the arguments $(\mathcal{T}, \mathcal{A}, \mathcal{N})$ whenever they are clear from the context. For a role R, the set $SC(R)$, where *SC* stands for *sub-concepts*, is a set of those concepts which are *immediately* under $\exists R$ in the concept hierarchy generated by $\mathcal{T}$:

$$SC(R) = \{A \mid \mathcal{T} \models A \sqsubseteq \exists R \text{ and there is no } A' \text{ s.t. } \mathcal{T} \models A \sqsubseteq A' \text{ and } \mathcal{T} \models A' \sqsubseteq \exists R\}.$$

If f is an ABox assertion, then $root_{\mathcal{T}}^{at}(f)$ is a set of all the atoms that $\mathcal{T}$-entail f. For example, $A(x) \in root_{\mathcal{T}}^{at}(\exists R(x))$ if $\mathcal{T} \models A \sqsubseteq \exists R$.

We are ready to proceed to construction of prototypes.

Constructing Zero-Prototype. The procedure $BZP(\mathcal{K}, \mathcal{N})$ (*Build Zero Prototype*) in Figure 2 constructs the main prototype $\mathcal{J}_0$ for $\mathcal{K}$ and $\mathcal{N}$ from *DL-Lite*$_{\mathcal{R}}^{I}$, which we call *zero-prototype*. Based on $\mathcal{J}_0$ we will construct all the other prototypes. To build $\mathcal{J}_0$ one has to align the canonical model of $\mathcal{K}$ with $\mathcal{N}$, and then delete from the resulting set of atoms all the auxiliary atoms $R(a, b)$ (from $AA(\mathcal{K}, \mathcal{N})$). In the case when *no* $R(a,\beta)$ for some constant β such that $R(a,\beta) \in AA(\mathcal{K}, \mathcal{N})$ is in the canonical model, we also delete atoms $root_{\mathcal{T}}^{at}(\exists R(a))$, since their presence in the model and the absence of R-atoms with a at the first coordinate would contradict the TBox.

Constructing Other Prototypes. The procedure $BP(\mathcal{K}, \mathcal{N}, \mathcal{J}_0)$ (*Build Prototypes*) of constructing $\mathfrak{J}$ for the case of *DL-Lite*$_{\mathcal{R}}^{I}$, takes $\mathcal{J}_0$ and manipulates with it by first dropping atoms from FA and then adding atoms in order to compensate the dropped ones so that the result is an evolved *model* under WS. It can be found in Figure 3

We conclude the discussion on the algorithms with a theorem:

Theorem 5. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_{\mathcal{R}}^{I}$ KB, and $\mathcal{N}$ a DL-Lite$_{\mathcal{R}}$ ABox consistent with $\mathcal{T}$. Then the set $BP(\mathcal{K}, \mathcal{N}, BZP(\mathcal{K}, \mathcal{N}))$ is a prototypal set for $\mathcal{K} \diamond \mathcal{N}$.*

Continuing with Example 1, it is easy to check that the prototypal set for $\mathcal{K}_1$ and $\mathcal{N}_1$ is $\{\mathcal{J}_0, \mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3\}$, where $\mathcal{J}_0$, $\mathcal{J}_1$, and $\mathcal{J}_2$ are described in the example and

$$\mathcal{J}_3: \quad A^{\mathcal{I}} = \{x, y\}, \quad C^{\mathcal{I}} = \{b\}, \quad R^{\mathcal{I}} = \{(x, d), (y, e)\}.$$

We proceed to correctness of BP in capturing evolution in *DL-Lite*$_{\mathcal{R}}^{I}$, where we use the following set $\mathtt{FC}[\mathcal{T}, \mathcal{A}, \mathcal{N}](R_i) = \{c \mid D(c) \in \mathtt{FA}[\mathcal{T}, \mathcal{A}, \mathcal{N}](R_i)\}$, that collects all the constants that participate in the forbidden atoms.

Theorem 6. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$^{I}_{\mathcal{R}}$ KB, $\mathcal{N}$ a DL-Lite$_{\mathcal{R}}$ ABox consistent with $\mathcal{T}$, and $BP(\mathcal{K}, \mathcal{N}, BZP(\mathcal{K}, \mathcal{N})) = \{\mathcal{J}_0, \dots, \mathcal{J}_n\}$ is a prototypal set for $\mathcal{K} \diamond \mathcal{N}$. Then*

$$\mathcal{K} \diamond \mathcal{N} = \mathit{Mod}(\mathcal{T}) \cap \mathit{Mod}(\mathcal{A}_0 \vee \dots \vee \mathcal{A}_n) \cap \mathit{Mod}(\Phi \wedge \Psi),$$

where A_i is a DL-Lite$_{\mathcal{R}}$ ABox such that $\mathcal{J}_i$ is a canonical model for $(\mathcal{T}, \mathcal{A}_i)$, and

$$\Phi = \bigwedge_{R_i \in TR} \bigwedge_{c_j \in \mathtt{FC}[R_i]} \forall x. \left[\left(R_i(x, c_j) \rightarrow (\mathit{root}^{at}_{\mathcal{T}}(\exists R_i(x)) \neq \emptyset) \right) \wedge \right.$$
$$\left. \forall y. \left(R_i(x, c_j) \wedge R_i(x, y) \rightarrow y = c_j \right) \right],$$

$$\Psi = \bigwedge_{R(a,b) \in \mathcal{S}_{at}} \exists R(a) \rightarrow \mathit{root}^{at}_{\mathcal{T}}(\exists R(a)) \cap \mathit{fcl}_{\mathcal{T}}(\mathcal{A}).$$

The $\mathcal{A}_i$ mentioned in Theorem 6, can be constructed in the similar way that the corresponding prototypes $\mathcal{J}_i$, taking the original ABox $\mathcal{A}$ instead of $\mathcal{I}^{can}$. Note that an ABox may include a negative literals, like $\neg B(c)$. Those should be treated in the same way that the positive literal (atoms) are. We will denote such an ABox as $\mathcal{A}[\mathcal{J}_i]$.

Theorem 7. *A prototype $\mathcal{J}_i$ is a canonical model of the KB $(\mathcal{T}, \mathcal{A}[\mathcal{J}_i])$.*

Continuing with Example 1, the ABoxes $\mathcal{A}[\mathcal{J}_0]$ and $\mathcal{A}[\mathcal{J}_1]$ are as follows:

$$\mathcal{A}[\mathcal{J}_0] = \{C(d), C(e), C(b)\}; \quad \mathcal{A}[\mathcal{J}_1] = \{A(x), C(e), C(b), R(x, d)\}.$$

$\mathcal{A}[\mathcal{J}_2]$ and $\mathcal{A}[\mathcal{J}_3]$ can be built in the similar way. Note that only $\mathcal{A}[\mathcal{J}_0]$ is in *DL-Lite$_{\mathcal{R}}$*, while writing $\mathcal{A}[\mathcal{J}_1], \dots, \mathcal{A}[\mathcal{J}_3]$ requires variables in ABoxes. Variables, also known as *soft constants*, are not allowed in *DL-Lite$_{\mathcal{R}}$* ABoxes, while present in *DL-Lite$_{\mathcal{RS}}$* ABoxes. Soft constants x are constants not constrained by the Unique Name Assumption: it is not necessary that $x^{\mathcal{I}} = x$. Since *DL-Lite$_{\mathcal{RS}}$* is tractable and FO rewritable [13], expressing $\mathcal{A}[\mathcal{J}_1]$ in *DL-Lite$_{\mathcal{RS}}$* instead of *DL-Lite$_{\mathcal{R}}$* does not affect tractability.

6 Computing Winslett's Semantics with Roles Interaction

The algorithm BP for constructing prototypal set works only when roles do not interact. The following example illustrates that it does not work in a general case.

Example 8. Consider a KB $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A}_2)$ and a new ABox $\mathcal{N}_2 = \{C(b)\}$:
TBox $\mathcal{T}_2$: $\exists R^- \sqsubseteq \neg \exists P^-$, $\exists R^- \sqsubseteq \neg C$, $A \sqsubseteq \exists R$, $B \sqsubseteq \exists P$;
ABox $\mathcal{A}_2$: $R(a, b)$, $A(a)$, $R(f, g)$, $A(f)$, $P(c, d)$, $B(c)$, $C(e)$.
One can check that the following model $\mathcal{J}'$ is in $\mathcal{K}_2 \diamond \mathcal{N}_2$:

$$A^{\mathcal{J}'} = \{y\}, \quad B^{\mathcal{J}'} = \{z\}, \quad C^{\mathcal{J}'} = \{b, e\}, \quad R^{\mathcal{J}'} = \{(y, d)\}, \quad P^{\mathcal{J}'} = \{(z, g)\}.$$

At the same time, BP over $\mathcal{K}_2$ and $\mathcal{N}_2$ returns the following four prototypes only:

	$A^{\mathcal{J}_i}$	$B^{\mathcal{J}_i}$	$C^{\mathcal{J}_i}$	$R^{\mathcal{J}_i}$	$P^{\mathcal{J}_i}$
$i = 0$	$\{f\}$	$\{c\}$	$\{b, e\}$	$\{(f, g)\}$	$\{(c, d)\}$
$i = 1$	$\{f, x\}$	$\{c\}$	$\{b\}$	$\{(f, g), (x, e)\}$	$\{(c, d)\}$
$i = 2$	$\{f, y\}$	$\emptyset$	$\{b, e\}$	$\{(f, g), (y, d)\}$	$\emptyset$
$i = 3$	$\{f, x, y\}$	$\emptyset$	$\{\ \ b\}$	$\{(f, g), (x, e), (y, d)\}$	$\emptyset$

where x and y are fresh constants. It is easy to see that none of $\mathcal{J}_i$s is homomorphically embeddable in $\mathcal{J}'$. Thus, BP does not capture $\mathcal{J}'$ and it is incomplete. ■

```
                    $BP_{rec}(\mathcal{K},\mathcal{N})$
1. Compute $\mathfrak{J} := BP(\mathcal{K},\mathcal{N}, BZP(\mathcal{K},\mathcal{N}))$.
2. Repeat
     $\mathfrak{J}' := \mathfrak{J}$;
     for each $\mathcal{J} \in \mathfrak{J}'$ do $\mathfrak{J} := \mathfrak{J} \cup BP(\mathcal{K},\mathcal{N} \cup S, \mathcal{J})$,
       where $S = \{f \in \mathcal{J} \mid$ a fresh constant from $\Delta \setminus adom(\mathcal{K})$ appears in $f\}$;
   until $\mathfrak{J} = \mathfrak{J}'$.
3. Return $\mathfrak{J}$.
```

Fig. 4. The procedure of building prototypes in *DL-Lite*$_\mathcal{R}$

6.1 Recursive BP Algorithm.

For general *DL-Lite*$_\mathcal{R}$ KBs, BP algorithm does return prototypes but not all of them. The reason is: when, while constructing prototypes with BP, we delete a forbidden atom (an atom from FA), it may trigger another dually-affected role and such triggering may require further modifications, which are not accounted by BP. In order to compute all prototypes we should run BP recursively: considering the prototypes obtained at the previous step as zero ones. We present a recursive algorithm BP_{rec} for building prototypes for general *DL-Lite*$_\mathcal{R}$ KBs in Figure 4. The following theorem shows the correctness of the algorithm.

Theorem 9. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_\mathcal{R}$ KB and $\mathcal{N}$ a DL-Lite$_\mathcal{R}$ ABox consistent with $\mathcal{T}$. Then the algorithm $BP_{rec}(\mathcal{K},\mathcal{N})$ terminates and returns the finite set which is a prototypal set for $\mathcal{K} \diamond \mathcal{N}$.*

We illustrate BP_{rec} on the following example.

Example 10. Consider KB $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A}_2)$ and a new ABox $\mathcal{N}_2$ from Example 8. Let us compute $BP_{rec}(\mathcal{K}_2,\mathcal{N}_2)$. First we run $BP(\mathcal{K},\mathcal{N},\mathcal{J}_0)$ and it returns four prototypes: $\mathcal{J}_0$, $\mathcal{J}_1$, $\mathcal{J}_2$, and $\mathcal{J}_3$ (see Example 8). Now we apply the BP procedure to $\mathcal{J}_1$, $\mathcal{J}_2$, and $\mathcal{J}_3$. It is easy to see that $BP(\mathcal{K},\mathcal{N} \cup \{A(x), R(x,e)\}, \mathcal{J}_1) = \emptyset$, since no role atom except for $R(a,b)$ was affected. Consider $BP(\mathcal{K},\mathcal{N} \cup \{A(y), R(y,d)\}, \mathcal{J}_2)$: it consists of the only prototype $\mathcal{J}_4$:

$$A^{\mathcal{J}_4} = \{y\}, \quad B^{\mathcal{J}_4} = \{z\}, \quad C^{\mathcal{J}_4} = \{b,e\}, \quad R^{\mathcal{J}_4} = \{(y,d)\}, \quad P^{\mathcal{J}_4} = \{(z,g)\}.$$

The uniqueness of the prototype follows from the fact that the role atom that was affected in $\mathcal{J}_2$ is $P(c,d)$ and $\mathtt{FA}[\mathcal{T},\mathcal{A},\mathcal{N} \cup \{A(y), R(y,d)\}](P) = \{\exists R^-(g)\}$. Finally, running $BP(\mathcal{T},\mathcal{N} \cup \{A(y), R(y,d), B(z), P(z,g)\}, \mathcal{J}_4)$ we obtain a prototype $\mathcal{J}_5$:

$$A^{\mathcal{J}_5} = \{y,v\},\ B^{\mathcal{J}_5} = \{z\},\ C^{\mathcal{J}_5} = \{b\},\ R^{\mathcal{J}_5} = \{(y,d),(v,e)\},\ P^{\mathcal{J}_5} = \{(z,g)\}.$$

Note that $BP(\mathcal{T},\mathcal{N} \cup \{A(y), R(y,d), B(z), P(z,g), A(v), R(v,e)\}, \mathcal{J}_5) = \emptyset$. Analogously, $\mathcal{J}_6$ can be obtained by running $BP(\mathcal{K},\mathcal{N} \cup \{A(x), A(y), R(x,e), R(y,d)\}, \mathcal{J}_3)$:

$$A^{\mathcal{J}_6} = \{x,y\},\ B^{\mathcal{J}_6} = \{z\},\ C^{\mathcal{J}_6} = \{b\},\ R^{\mathcal{J}_6} = \{(x,e),(y,d)\},\ P^{\mathcal{J}_6} = \{(z,g)\}.$$

Thus, the prototypal set $\mathfrak{J}$ for $\mathcal{K} \diamond \mathcal{N}$ is $\{\mathcal{J}_i\}_{i=0}^{6}$. ∎

We conclude with the theorem that BP_{rec} gives a sound approximation for WS.

Theorem 11. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_{\mathcal{R}}$ KB, $\mathcal{N}$ a DL-Lite$_{\mathcal{R}}$ ABox consistent with $\mathcal{T}$, and $BP_{rec}(\mathcal{K}, \mathcal{N}) = \{\mathcal{J}_0, \dots, \mathcal{J}_n\}$ is a prototypal set for $\mathcal{K} \diamond \mathcal{N}$. Then*

$$\mathcal{K} \diamond \mathcal{N} \subseteq \mathit{Mod}(\mathcal{T}) \cap \mathit{Mod}(\mathcal{A}_0 \vee \dots \vee \mathcal{A}_n) \cap \mathit{Mod}(\Phi \wedge \Psi),$$

where A_i is a DL-Lite$_{\mathcal{R}}$ ABox such that $\mathcal{J}_i$ is a canonical model for $(\mathcal{T}, \mathcal{A}_i)$ and Φ and Ψ are as they defined in Theorem 6.

6.2 Closure Under Evolution and Approximation

Next theorem allows us to approximate results of evolution under WS, since $\mathsf{FO}[2]$ is decidable.

Theorem 12. *$\mathcal{K} \diamond \mathcal{N}$ under WS for KBs in DL-Lite$_{\mathcal{R}}$ can be captured in* $\mathsf{FO}[2]$.

As a future work we are going to study ways to approximate the resulted $\mathsf{FO}[2]$ theories in *DL-Lite*.

Finally, we discuss cases when the result of Winslett's evolution is expressible in *DL-Lite*$_{\mathcal{R}}$. The following formulas appearing in Theorem 6 are not expressible in *DL-Lite*$_{\mathcal{R}}$: *(i)* the disjunction of the ABoxes $\mathcal{A}_0 \vee \dots \vee \mathcal{A}_n$ and *(ii)* formula $\Phi \wedge \Psi$. The disjunction of ABoxes becomes expressible when it is of the length one, i.e., there is the only prototype: $\mathcal{J}_0$. The last statement yields that $\mathtt{FC} = \emptyset$ and therefore Φ is always true. The formula Ψ becomes trivially true when $AA = \emptyset$, i.e., for every atom $R(a,b) \in \mathit{fcl}_{\mathcal{T}}(\mathcal{A})$ either $\mathcal{N} \not\models_{\mathcal{T}} \neg\exists R^-(b)$ or $\mathit{root}^{at}_{\mathcal{T}}(\exists R_i(a_i)) \cap \mathit{fcl}_{\mathcal{T}}(\mathcal{A}) = \emptyset$. As one can see, the condition of expressibility of the result in *DL-Lite*$_{\mathcal{R}}$ (emptiness of $\mathtt{FA}$ and AA), depends on a TBox, an ABox, and a new information. Hence, if we do a chain of evolution, at some step the result may be not expressible in *DL-Lite*$_{\mathcal{R}}$. Since TBox stays unchangeable, to guarantee the expressibility we need to find TBoxes $\mathcal{T}$ such that $(\mathcal{T}, \mathcal{A}) \diamond \mathcal{N}$ is expressible in *DL-Lite*$_{\mathcal{R}}$ for every $\mathcal{A}$ and $\mathcal{N}$. A condition that guarantees the emptiness of $\mathtt{FA}$ and AA is: for every role $R \in \Sigma(K \cup \mathcal{N})$ at least one of the following items holds: *(1)* there is no concept C such that $\mathcal{T} \models \exists R^- \sqsubseteq \neg C$, or *(2)* there is no concept A such that $\mathcal{T} \models A \sqsubseteq \exists R$. The former conditions gives that $TR = \emptyset$ since $\mathcal{N} \not\models_{\mathcal{T}} \neg\exists R^-(b)$, which leads to $\mathtt{FA} = AA = \emptyset$. The latter one yields that $SC(R) = \emptyset$, therefore *TR* again is empty.

As a practical summary of this section, given a KB $\mathcal{K}$ and a new ABox $\mathcal{N}$, one can check (in polynomial time) whether any dually-affected role is "triggered" by $\mathcal{N}$. If it is *not* the case, one can compute (in polynomial time) an evolved KB $\mathcal{K}'$ that exactly captures $\mathcal{K} \diamond \mathcal{N}$. Otherwise, it is the case that $\mathcal{K} \diamond \mathcal{N}$ is inexpressible in *DL-Lite*$_{\mathcal{R}}$. Thus, one can compute an $\mathsf{FO}[2]$ theory that captures $\mathcal{K} \diamond \mathcal{N}$ and then approximate it in *DL-Lite*$_{\mathcal{R}}$, by, for example, dropping all the not *DL-Lite*$_{\mathcal{R}}$ formulas. We will not focus on approximation in this paper.

7 Conclusion

We studied how to capture ABox evolution for *DL-Lite*$_{\mathcal{R}}$ under WS. In general the result of evolution requires constructs that are not present in *DL-Lite*$_{\mathcal{R}}$, and even not in *DL-Lite*, such as disjunction. Moreover, in general the result of evolution, which is a set of models, does not even have a canonical model, which should always exist for any *DL-Lite* theory. It turned out that the inexpressibility is caused by a condition on the TBox level, which we called dual-interaction: by pairs of assertions of the form $A \sqsubseteq \exists R$ and $\exists R^- \sqsubseteq \neg B$. In order to capture evolution results in the presence of dual-interactions, we introduced prototypes. Our approach is based on the observation that evolution results can be divided into a finite number of subsets and each of them has a canonical model, i.e., a prototype. These subsets can be captured by theories guided by prototypes and the disjunction of these theories, compensated with two formulas, captures evolution results and is in $\mathsf{FO}[2]$. We proved that this technique works for *DL-Lite*$_{\mathcal{R}}$. We are currently working on efficient approximation of the obtained $\mathsf{FO}[2]$ theory in *DL-Lite* and on extending results to capture evolution for other *DL-Lite* languages.

Acknowledgements We are thankful to Diego Calvanese and Werner Nutt for insightful discussions. The authors are supported by EU projects ACSI (FP7-ICT-257593), Ontorule (FP7-ICT-231875). The first author was supported by the ERC FP7 grant Webdam (agreement n. 226513). Part of this work was carried out while the first author was visiting Department of Computer Science of Oxford University.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook. Cambridge University Press (2003)
2. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. Knowledge Engineering Review **23**(2) (2008) 117–152
3. Abiteboul, S., Grahne, G.: Update semantics for incomplete databases. (VLDB-85)
4. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. (KR-91)
5. Eiter, T., Gottlob, G.: On the complexity of propositional knowledge base revision, updates and counterfactuals. Artificial Intelligence **57** (1992) 227–270
6. Winslett, M.: Updating Logical Databases. Cambridge University Press (1990)
7. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. (KR-06)
8. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3) (2007) 385–429
9. Giacomo, G.D., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. of Logic and Computation, Special Issue on Ontology Dynamics **19**(5) (2009) 745–770
10. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of DL-Lite knowledge bases. In: International Semantic Web Conference (1). (2010) 112–128
11. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of DL-Lite knowledge bases (extended version). Technical Report KRDB11-3, KRDB Research Centre (2011)
12. Kharlamov, E., Zheleznyakov, D.: Understanding inexpressibility of model-based ABox evolution in DL-Lite. (AMW-11)
13. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR) **36** (2009) 1–69
14. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research **36** (2009) 1–69
15. Poggi, A., Lembo, D., C., D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics (2008) 133–173

On (In)Tractability of OBDA with *OWL 2 QL*

S. Kikot, R. Kontchakov, and M. Zakharyaschev

Department of Computer Science and Information Systems,
Birkbeck College, London. U.K.
{kikot,roman,michael}@dcs.bbk.ac.uk

Abstract. We show that, although conjunctive queries over *OWL 2 QL* ontologies are reducible to database queries, no algorithm can construct such a reduction in polynomial time without changing the data. On the other hand, we give a polynomial reduction for *OWL 2 QL* ontologies without role inclusions.

1 Introduction

Ontology-based data access (OBDA) [9, 13, 21] has recently emerged as a promising application area for description logic (DL) with a potential impact on the new generation of information systems. One of the profiles of the Web Ontology Language *OWL 2*, called *OWL 2 QL*, was tailored specifically aiming at OBDA. In DL terms, OBDA involves the following reasoning problem:

CQA$(\mathcal{A}, \mathcal{T}, q)$**:** given an ABox (data) $\mathcal{A}$,[1] a TBox (ontology describing the background knowledge) $\mathcal{T}$, a conjunctive query (CQ) $q(\boldsymbol{x})$, and a tuple $\boldsymbol{a}$ of ABox elements, decide whether $\boldsymbol{a}$ is a certain answer to $q(\boldsymbol{x})$ over $(\mathcal{T}, \mathcal{A})$.

In other words, the task is to check whether $\mathcal{I} \models q(\boldsymbol{a})$ for every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$. It is to be noted that reasoning problems of this kind are well known in logic and computer science (cf. Prolog or Datalog). A distinctive feature of *OWL 2 QL* is that 'in *OWL 2 QL*, conjunctive query answering can be implemented using conventional relational database systems. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in LogSpace with respect to the size of the data' (`www.w3.org/TR/owl2-profiles`).

There exists a number of reductions of OBDA with *OWL 2 QL* to answering queries in relational database management systems, which transform (or *rewrite*) the problem CQA$(\mathcal{A}, \mathcal{T}, q)$ to the database query evaluation problem QE$(\mathcal{A}, q')$, where the first-order (FO) query q' does not depend on $\mathcal{A}$. They have been implemented in the systems QuOnto [1], REQUIEM [20], Presto [23] and Nyaya [11]. In all of these approaches, the size of the query q', posed to the database system, can be $\mathcal{O}((|\mathcal{T}| \cdot |q|)^{|q|})$ in the worst case.

The aim of this paper is to try and understand whether the exponential blow-up in the size of the rewritten query is inevitable and whether polynomial rewritings are possible, at least for fragments of *OWL 2 QL*. In Section 3, we show that

[1] Here we ignore the problem of data representation in database systems; see Section 5.

the problem CQA($\{A(a)\}, \mathcal{T}, q$) for singleton ABoxes and *OWL 2 QL* TBoxes is NP-complete for combined complexity. As the problem QE($\{A(a)\}, q'$) is solved in linear time (LogSpace) in $|q'|$, it follows that no algorithm can construct FO rewritings q' (over $\{A(a)\}$) in polynomial time, unless P = NP. For *OWL 2 QL* without role inclusions and the logic $\mathcal{ELH}$, the problem CQA($\{A(a)\}, \mathcal{T}, q$) is polynomial for combined complexity, while for $\mathcal{ELI}$ it is ExpTime-complete. We observe that the parameterised complexity of the problem CQA($\{A(a)\}, \mathcal{T}, q$), where $|q|$ is regarded as a parameter, is fixed-parameter tractable. In Section 4, we present a polynomial FO rewriting of conjunctive queries over *OWL 2 QL* ontologies without role inclusions. This result improves on the polynomial rewriting from [14], which reduces CQA($\mathcal{A}, \mathcal{T}, q$) to QE($\mathcal{A} + Aux, q'$), where Aux is a set of fresh constants encoding the canonical model of $(\mathcal{T}, \mathcal{A})$. Note also the recent polynomial reduction [12] of CQA($\mathcal{A}, \mathcal{T}, q$) to QE($\mathcal{A} + \{0,1\}, q''$), which uses two fresh constants 0, 1 and works for the extension Datalog± of *OWL 2 QL* (see Remark 1). We discuss the implications of the obtained results for OBDA in Section 5.

2 *OWL 2 QL* and *DL-Lite*$^{\mathcal{H}}_{core}$

The description logic underlying *OWL 2 QL* was introduced under the name *DL-Lite*$_{\mathcal{R}}$ [6, 7] and called *DL-Lite*$^{\mathcal{H}}_{core}$ in the more general classification of [2] (for simplicity, we disregard some constructs such as reflexivity constraints). The language of *DL-Lite*$^{\mathcal{H}}_{core}$ contains *individual names* a_i, *concept names* A_i, and *role names* P_i, $i \geq 1$. *Roles* R and *concepts* B are defined by:

$$R \quad ::= \quad P_i \quad | \quad P_i^-, \qquad\qquad B \quad ::= \quad \bot \quad | \quad \top \quad | \quad A_i \quad | \quad \exists R.$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ *TBox*, $\mathcal{T}$, is a finite set of *concept* and *role inclusions* of the form $B_1 \sqsubseteq B_2$, $B_1 \sqcap B_2 \sqsubseteq \bot$ and $R_1 \sqsubseteq R_2$, $R_1 \sqcap R_2 \sqsubseteq \bot$, respectively. An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $B(a_i)$ and $R(a_i, a_j)$. $\mathcal{T}$ and $\mathcal{A}$ together constitute the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The semantics of *DL-Lite*$^{\mathcal{H}}_{core}$ is defined as usual in DL [4]. The presented results do not depend on the UNA. *DL-Lite*$_{core}$ is *DL-Lite*$^{\mathcal{H}}_{core}$ without role inclusions of the form $R_1 \sqsubseteq R_2$. Note also that *OWL 2 QL* contains concept inclusions of the form $B' \sqsubseteq \exists R.B$, which here will be regarded as abbreviations for *DL-Lite*$^{\mathcal{H}}_{core}$ inclusions $B' \sqsubseteq \exists R_B$, $\exists R_B^- \sqsubseteq B$ and $R_B \sqsubseteq R$, where R_B is a fresh role name.

A *conjunctive query* (CQ) $q(\boldsymbol{x})$ is a first-order formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is constructed, using only $\wedge$, from atoms of the form $A(t_1)$ and $P(t_1, t_2)$, with A being a concept name, P a role name and t_i a *term* (an individual name or variable from $\boldsymbol{x}$ or $\boldsymbol{y}$). Given an ABox $\mathcal{A}$, we use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names in $\mathcal{A}$. A tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is a *certain answer* to $q(\boldsymbol{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$; in this case we write $\mathcal{K} \models q[\boldsymbol{a}]$. To simplify notation, we will often identify q with the set of its atoms and use $P^-(t, t') \in q$ as a synonym of $P(t', t) \in q$; $\mathsf{term}(q)$ is the set of terms in q.

Query answering over *OWL 2 QL* KBs is based on the fact that, for any consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, there is an interpretation $\mathcal{U}_{\mathcal{K}}$ such that, for all CQs

$q(\boldsymbol{x})$ and $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{K} \models q[\boldsymbol{a}]$ iff $\mathcal{U}_\mathcal{K} \models q[\boldsymbol{a}]$. The interpretation $\mathcal{U}_\mathcal{K}$, called the *canonical interpretation* of $\mathcal{K}$, is constructed as follows. Let $\sqsubseteq^*_\mathcal{T}$ be the reflexive and transitive closure of the role inclusion relation given by $\mathcal{T}$, $[R] = \{S \mid R \sqsubseteq^*_\mathcal{T} S \text{ and } S \sqsubseteq^*_\mathcal{T} R\}$, and let $[R] \leq_\mathcal{T} [S]$ iff $R \sqsubseteq^*_\mathcal{T} S$. For each $[R]$, we introduce a fresh symbol $c_{[R]}$, the *witness for* $[R]$, and define a *generating relation* $\rightsquigarrow_\mathcal{K}$ on the set of these witnesses together with $\mathsf{Ind}(\mathcal{A})$ by taking:

– $a \rightsquigarrow_\mathcal{K} c_{[R]}$ if $a \in \mathsf{Ind}(\mathcal{A})$ and $[R]$ is $\leq_\mathcal{T}$-minimal such that $\mathcal{K} \models \exists R(a)$ and $\mathcal{K} \not\models R(a,b)$ for every $b \in \mathsf{Ind}(\mathcal{A})$;
– $c_{[S]} \rightsquigarrow_\mathcal{K} c_{[R]}$ if $[R]$ is $\leq_\mathcal{T}$-minimal with $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$ and $[S^-] \neq [R]$.

A *generating path* for $\mathcal{K}$ is a finite sequence $ac_{[R_1]} \cdots c_{[R_n]}$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$, $a \rightsquigarrow_\mathcal{K} c_{[R_1]}$ and $c_{[R_i]} \rightsquigarrow_\mathcal{K} c_{[R_{i+1}]}$, for $i < n$. Denote by $\mathsf{path}(\mathcal{K})$ the set of all generating paths for $\mathcal{K}$ and by $\mathsf{tail}(\sigma)$ the last element in $\sigma \in \mathsf{path}(\mathcal{K})$. Now, $\mathcal{U}_\mathcal{K}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{U}_\mathcal{K}} &= \mathsf{path}(\mathcal{K}), \quad a^{\mathcal{U}_\mathcal{K}} = a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{U}_\mathcal{K}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{\sigma \cdot c_{[R]} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\},\\
P^{\mathcal{U}_\mathcal{K}} &= \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models P(a,b)\} \cup\\
&\quad \{(\sigma, \sigma \cdot c_{[R]}) \mid \mathsf{tail}(\sigma) \rightsquigarrow_\mathcal{K} c_{[R]},\ [R] \leq_\mathcal{T} [P]\} \cup\\
&\quad \{(\sigma \cdot c_{[R]}, \sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow_\mathcal{K} c_{[R]},\ [R] \leq_\mathcal{T} [P^-]\}.
\end{aligned}$$

We shall also need a compact representation of (in general infinite) $\mathcal{U}_\mathcal{K}$ in the form of the *generating interpretation* $\mathcal{G}_\mathcal{K} = (\Delta^{\mathcal{G}_\mathcal{K}}, \cdot^{\mathcal{G}_\mathcal{K}})$ defined as follows. Its domain $\Delta^{\mathcal{G}_\mathcal{K}}$ consists of $\mathsf{Ind}(\mathcal{A})$ and all $c_{[R_n]}$ for which there are generating paths $ac_{[R_1]} \cdots c_{[R_n]} \in \mathsf{path}(\mathcal{K})$; and we set $a^{\mathcal{G}_\mathcal{K}} = a^{\mathcal{U}_\mathcal{K}}$, $A^{\mathcal{G}_\mathcal{K}} = \{\mathsf{tail}(\sigma) \mid \sigma \in A^{\mathcal{U}_\mathcal{K}}\}$ and $P^{\mathcal{G}_\mathcal{K}} = \{(\mathsf{tail}(\sigma), \mathsf{tail}(\sigma')) \mid (\sigma, \sigma') \in P^{\mathcal{U}_\mathcal{K}}\}$. It is readily seen that $\mathcal{G}_\mathcal{K}$ can be constructed in polynomial time in $\mathcal{K}$.

The problem CQA$(\mathcal{A}, \mathcal{T}, q)$, for *DL-Lite*$^\mathcal{H}_{core}$ TBoxes $\mathcal{T}$, is reducible to the database query evaluation problem QE$(\mathcal{A}, q')$, with q' being independent of $\mathcal{A}$ [7, 2]. However, in all known reductions, the size of q' is exponential in the size of q: for instance, $|q'| = \mathcal{O}((|\mathcal{T}| \cdot |q|)^{|q|})$ for both QuOnto [1] and REQUIEM [20]; Presto [23] uses sophisticated optimisation techniques and produces a non-recursive Datalog program q', which is still exponential in the worst case. The size of q' is irrelevant for data complexity, but heavily influences the performance of database systems; see Section 5 for a discussion.

In the next section, we show that no algorithm can produce FO rewritings q' of CQs q and *DL-Lite*$^\mathcal{H}_{core}$ TBoxes $\mathcal{T}$ in polynomial time (unless P = NP).

3 Intractability of Query Rewriting for *OWL 2 QL*

To see that query rewriting for *DL-Lite*$^\mathcal{H}_{core}$ is not tractable, we separate the contributions of $\mathcal{A}$ and $\mathcal{T}$ to the complexity of the problem CQA$(\mathcal{A}, \mathcal{T}, q)$. Indeed, NP-completeness of CQA$(\mathcal{A}, \mathcal{T}, q)$ for combined complexity does not give any information on the size of rewritings because the lower bound follows from NP-hardness of database query evaluation. To remove the influence of $\mathcal{A}$, one can analyse the combined complexity of CQ answering over *singleton* ABoxes of the form $\mathcal{A} = \{A(a)\}$, i.e., the problem CQA($\{A(a)\}, \mathcal{T}, q$). We call this measure the *primitive combined complexity* (PCC). The reason behind this notion is that any FO query q over such an ABox alone can be answered in linear time in $|q|$. Thus, if CQ answering is NP-hard for PCC, then no algorithm can *construct* FO rewritings QE($\mathcal{A}, q'$) of CQA($\mathcal{A}, \mathcal{T}, q$) in polynomial time, unless P = NP.

Theorem 1. *CQ answering in DL-Lite$^{\mathcal{H}}_{core}$ is* NP*-complete for PCC.*

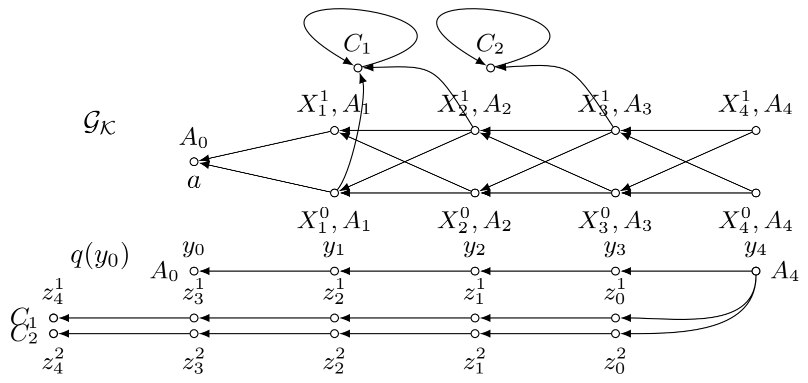
Proof. The lower bound is proved by reduction of Boolean satisfiability. Given a CNF $\varphi = \bigwedge_{j=1}^{m} D_j$ over variables $p_1, \dots, p_n$, where D_j is a clause, we consider the TBox $\mathcal{T}$ containing the following axioms, for $1 \leq i \leq n$, $1 \leq j \leq m$, $k = 0, 1$:

$$A_{i-1} \sqsubseteq \exists P^{-}.X_i^k, \qquad X_i^k \sqsubseteq A_i,$$
$$X_i^0 \sqsubseteq \exists P.C_j \quad \text{if } \neg p_i \in D_j, \qquad X_i^1 \sqsubseteq \exists P.C_j \quad \text{if } p_i \in D_j, \qquad C_j \sqsubseteq \exists P.C_j.$$

The canonical interpretation $\mathcal{U}_{\mathcal{K}}$ of $\mathcal{K} = (\mathcal{T}, \{A_0(a)\})$ is obtained by 'unravelling' the generating interpretation $\mathcal{G}_{\mathcal{K}}$ shown below. Consider the CQ $q(y_0)$:

$$q(y_0) \;=\; \exists \boldsymbol{y}\boldsymbol{z}^1 \dots \boldsymbol{z}^m \left[A_0(y_0) \wedge \bigwedge_{i=1}^{n} P(y_i, y_{i-1}) \wedge A_n(y_n) \wedge \bigwedge_{j=1}^{m} \left(P(y_n, z_0^j) \wedge \bigwedge_{i=1}^{n} P(z_{i-1}^j, z_i^j) \wedge C_j(z_n^j)\right)\right].$$

(Note n atoms P connecting y_n to y_0 and $n+1$ atoms P connecting y_n to z_n^j, which means that any match of q in $\mathcal{U}_{\mathcal{K}}$ must map z_n^j onto a point in the infinite chain containing C_j.) One can show that φ is satisfiable iff $\mathcal{K} \models q(a)$.



Theorem 2. *Unless* P = NP*, no polynomial-time algorithm can reduce the problem CQA($\mathcal{A}, \mathcal{T}, q$), for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes $\mathcal{T}$ and CQs q, to the problem QE($\mathcal{A}, q'$), where q' is a first-order query independent of $\mathcal{A}$.*

Note that it is still open whether, for any $\mathcal{A}$, $\mathcal{T}$ and q, there *exists* a polynomial FO query q' giving the same answers over $\mathcal{A}$ as q over $(\mathcal{T}, \mathcal{A})$.

Remark 1. If we extend the ABox with fresh constants 0 and 1 then $q(y_0)$ in the proof above can be rewritten as $A_0(y_0) \wedge \exists p_1 \dots \exists p_n (\bigwedge_{i=1}^{n} (p_i \neq y_0) \wedge \bigwedge_{j=1}^{m} D_j')$, where D'_j is obtained from D_j by replacing every literal p_i with $p_i = 1$ and every $\neg p_i$ with $p_i = 0$. Moreover, using $\forall p_i$, one can polynomially encode the PSPACE-complete validity problem for QBFs. A polynomial reduction of CQA$(\mathcal{A}, \mathcal{T}, q)$ to QE$(\mathcal{A} + \{0,1\}, q')$ is given in [12] for the extension Datalog± of *OWL 2 QL*, where $|\mathcal{T}| + |q|$ steps of the chase are simulated using 0 and 1.

Theorem 1 means that there are two sources of non-determinism in OBDA with *OWL 2 QL*: finding a match in the ABox and finding a match in the remaining tree part of the canonical interpretation. It turns out that, from the complexity-theoretic point of view, these two sources have different status. Recall from [19] that query evaluation QE$(\mathcal{A}, q)$ is *not* fixed-parameter tractable if $|q|$ is regarded as a *parameter*.

Our next result shows, on the contrary, that the problem CQA$(\{A(a)\}, \mathcal{T}, q)$ is *fixed-parameter tractable* for *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes $\mathcal{T}$. This means that there exist a deterministic algorithm $\boldsymbol{A}$, a computable function f and a polynomial p such that, for any TBox $\mathcal{T}$ and CQ q, $\boldsymbol{A}$ can check whether $(\mathcal{T}, \{A(a)\}) \models q$ in time bounded by $f(|q|) \cdot p(|\mathcal{T}|)$. In a nutshell, the idea of the proof is as follows. First, given a CQ q, we construct all *tree-shaped* homomorphic images of q, the number of which is bounded by a function exponential in $|q|$ and independent of $\mathcal{T}$. Then we show that $(\mathcal{T}, \{A(a)\}) \models q$ iff at least one of those tree-shaped homomorphic images can be 'embedded' in $\mathcal{U}_\mathcal{K}$, and that the existence of such an embedding can be established by a dynamic programming (elimination) algorithm in time polynomial in $|\mathcal{T}|$ and $|q|$.

Theorem 3. *The problem CQA$(\{A(a)\}, \mathcal{T}, q)$ with $|q|$ a parameter is fixed-parameter tractable for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes $\mathcal{T}$.*

Proof. A CQ q is *tree-shaped* if its primal graph $(\mathsf{term}(q), \{(t,t') \mid R(t,t') \in q\})$ is a tree. By a *tree reduct* of q we mean a pair (q', r), where q' is a set of atoms and $r \in \mathsf{term}(q')$ is such that the following conditions are satisfied (cf. [10]):

(tree) the query q' is tree-shaped and all of its predicate names occur in q;
(root) if $a \in \mathsf{term}(q')$ then $r = a$;
(hom) there exists a surjection $h\colon \mathsf{term}(q) \to \mathsf{term}(q')$ such that $h(a) = a$ for $a \in \mathsf{term}(q)$, $A(h(t)) \in q'$ for $A(t) \in q$, and $P(h(t), h(t')) \in q'$ for $P(t,t') \in q$.

By **(hom)**, for every $\mathcal{I}$ and every tree reduct (q', r) of q, if $\mathcal{I} \models q'$ then $\mathcal{I} \models q$.

Let (q', r) be a tree reduct of q and let $\mathcal{K} = (\mathcal{T}, \{A(a)\})$. An *embedding* of (q', r) in $\mathcal{U}_\mathcal{K}$ is an *injective* map $\mathfrak{a}\colon \mathsf{term}(q') \to \Delta^{\mathcal{U}_\mathcal{K}}$ such that $\mathcal{U}_\mathcal{K} \models^{\mathfrak{a}} q'$ and

(e-root) $\mathfrak{a}(t) = \mathfrak{a}(r) \cdot \sigma$, for all $t \in \mathsf{term}(q')$, i.e., $\mathfrak{a}(r)$ is located in $\mathcal{U}_\mathcal{K}$ nearer to its root than any other $\mathfrak{a}(t)$.

Let $\mathcal{U}_\mathcal{K} \models q$. Then there is a homomorphism $\mathfrak{a}$ of q in $\mathcal{U}_\mathcal{K}$. As $\mathcal{U}_\mathcal{K}$ is a tree with root $a^{\mathcal{U}_\mathcal{K}}$, we can construct a tree reduct (q', r) of q by taking q' to be the quotient of q under equivalence $\{(t,t') \mid \mathfrak{a}(t) = \mathfrak{a}(t')\}$ and r the equivalence class of t such that $\mathfrak{a}(t)$ is nearest to the root $a^{\mathcal{U}_\mathcal{K}}$. It follows that (q', r) is embeddable in $\mathcal{U}_\mathcal{K}$. Checking whether a tree reduct (q', r) of q is embeddable in $\mathcal{U}_\mathcal{K}$ can be done in time polynomial in $|\mathcal{T}|$ and $|q|$ using the interpretation $\mathcal{G}_\mathcal{K}$ (constructed in polynomial time in $|\mathcal{T}|$) and a standard dynamic programming algorithm [8].

Theorem 1 reflects the interaction between role inclusions and inverse roles. The observations below supplement this theorem by giving a somewhat broader picture (we remind the reader that *DL-Lite*$_{horn}$ extends *DL-Lite*$_{core}$ with concept inclusions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$, $\mathcal{EL}$ allows qualified existential restrictions and conjunctions in both sides of concept inclusions, $\mathcal{H}$ allows role inclusions and $\mathcal{I}$ inverse roles; for details see [3]):

Theorem 4. *With respect to primitive combined complexity, CQ answering is* (i) P*-complete for DL-Lite*$_{horn}$ *and* $\mathcal{ELH}$*, and* (ii) ExpTime*-complete for* $\mathcal{ELI}$*.*

Proof. The polynomial-time upper bound for *DL-Lite*$_{horn}$ and $\mathcal{ELH}$ can be obtained using the fact that, for each CQ q and each $r \in \mathsf{term}(q)$, one can construct a *unique* tree reduct of q with root r (by eliminating 'forks') [16] and then check whether it is embeddable in the generating interpretation as in the proof of Theorem 3 (see also Section 4). ExpTime-completeness for $\mathcal{ELI}$ follows from [3].

Although $\mathcal{ALC}$ and $\mathcal{ALCH}$ have no canonical interpretations (they are not Horn), a unique tree reduct for a CQ with a root exists [10], and CQ answering is ExpTime-complete for PCC; in $\mathcal{ALCI}$, we again have to consider multiple tree reducts, which makes CQ answering 2ExpTime-complete for PCC [16].

That CQ answering is in P for PCC does not mean yet that there is a polynomial rewriting q' for any CQ q and ontology $\mathcal{T}$. For instance, as CQ answering for $\mathcal{ELH}$ is P-complete for data complexity, we cannot have any first-order rewriting at all. The reason is that if we put an ABox element a to a concept A, then a TBox axiom of the form $\exists R.A \sqsubseteq B$ requires adding every ABox element b with $R(b, a)$ to B, and so on. In this case, a pre-processing of the ABox, constructing the generating interpretation, is required; see [18].

4 Polynomial Rewriting for *DL-Lite*$_{core}$

The *combined approach* to CQ answering [18, 14] first constructs the generating interpretation $\mathcal{G}_{\mathcal{K}}$ for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and then rewrites the given CQ q (independently of $\mathcal{A}$) to an FO query q' to be answered *over* $\mathcal{G}_{\mathcal{K}}$. An important achievement of this approach is that (*i*) $|q'| = \mathcal{O}(|q|^2 + |q| \cdot |\mathcal{T}|)$, even for *DL-Lite*$_{horn}$, and (*ii*) q' is obtained by expanding q by simple conjuncts with $=$ and without any extra variables and quantifiers. The two-step construction of $\mathcal{G}_{\mathcal{K}}$ and q' can be encoded in a polynomial non-recursive Datalog program for *DL-Lite*$_{horn}$, and a polynomial FO query for *DL-Lite*$_{core}$, which require auxiliary constants in the database domain. Here we give a polynomial FO rewriting for *DL-Lite*$_{core}$, which is based on the ideas of [14] but does not involve any constants.

Let $\mathcal{T}$ be a *DL-Lite*$_{core}$ TBox. As we do not have role inclusions, instead of $c_{[R]}$ we write c_R. Let $\mathsf{R}_{\mathcal{T}} = \{c_R \mid R \text{ a role in } \mathcal{T}\}$ and $\mathsf{R}^*_{\mathcal{T}}$ be the set of all finite words over $\mathsf{R}_{\mathcal{T}}$ (including the empty word ε). We use $\mathsf{tail}(\sigma)$ to denote the last element of $\sigma \in \mathsf{R}^*_{\mathcal{T}} \setminus \{\varepsilon\}$; by definition, $\mathsf{tail}(\varepsilon) = \varepsilon$.

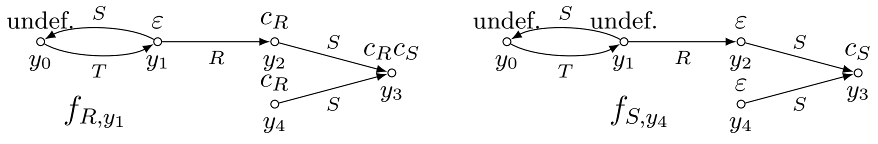
Consider a CQ $q(\boldsymbol{x})$. Without loss of generality we assume that (the primal graph of) q is connected. Let R be a role and t a term in q. A *partial* function f from $\mathsf{term}(q)$ to $(\mathsf{R}_{\mathcal{T}})^*$ is called a *tree witness for* (R, t) in q if

– the domain of f is minimal with respect to set-theoretic inclusion,
– $f(t) = \varepsilon$,
– for all atoms $S(s, s') \in q$ with $f(s)$ defined, we have

$$f(s') = \begin{cases} c_R, & \text{if } f(s) = \varepsilon \text{ and } S = R, \\ \sigma, & \text{if } f(s) = \sigma \cdot c_{S^-}, \\ f(s) \cdot c_S, & \text{if } f(s) \neq \varepsilon \text{ and } \mathsf{tail}(f(s)) \neq c_{S^-}. \end{cases}$$

By definition, if a tree witness for (R, t) exists then it is unique; in this case we denote it by $f_{R,t}$ and use $\mathsf{dom}\, f_{R,t}$ for the domain of $f_{R,t}$. Note that even if q contains no atom of the form $R(t, t')$, the tree witness for (R, t) exists and $f_{R,t}(t) = \varepsilon$. Denote by $q|_{R,t}$ the set of atoms of q whose terms are in $\mathsf{dom}\, f_{R,t}$. When we consider $q|_{R,t}$ as a query, we assume that all of its variables are *free*.

Informally, a tree witness $f_{R,t}$ has *root* t and *direction* R, and describes the situation where t is mapped to an ABox element a of some canonical interpretation without R-successors in the ABox. In this case, the only choice for mapping any t' in $R(t, t') \in q$ is $ac_R = a \cdot f_{R,t}(t')$. Further, any t'' in $S(t', t'') \in q$ has to be mapped to $ac_Rc_S = a \cdot f_{R,t}(t'')$, if $S \neq R^-$; however, if $S = R^-$ then t'' can only be mapped onto a, which reflects the fact that ac_R has a single R^--successor a in the canonical interpretation. To illustrate, consider the CQ $q = \{T(y_0, y_1),\ S(y_1, y_0),\ R(y_1, y_2),\ S(y_2, y_3),\ S(y_4, y_3)\}$. The tree witnesses for (R, y_1) and (S, y_4) in q exist and are as depicted below:



For (S, y_1), (T^-, y_1) and (R^-, y_2), tree witnesses do not exist.

Proposition 1. *Suppose a tree witness for (R, t) exists and $s \in \mathsf{dom}\, f_{R,t}$. If $f_{R,t}(s) \neq \varepsilon$ then a tree witness exists for every (S, s) with $\mathsf{tail}(f_{R,t}(s)) \neq c_{S^-}$. If $f_{R,t}(s) = \varepsilon$ then a tree witness exists for (S, s) with $S = R$. In either case, $\mathsf{dom}\, f_{S,s} \subseteq \mathsf{dom}\, f_{R,t}$ and $f_{R,t}(s') = f_{R,t}(s) \cdot f_{S,s}(s')$, for all $s' \in \mathsf{dom}\, f_{S,s}$.*

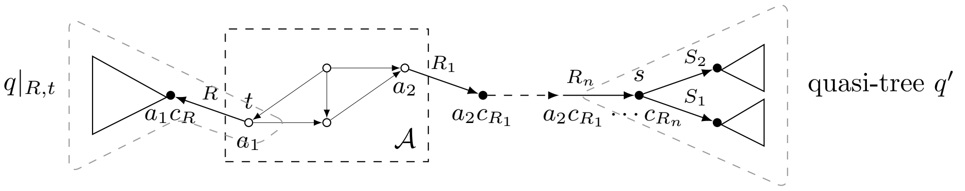
Even if a tree witness for (R, t) exists, $q|_{R,t}$ is not necessarily a tree-shaped query. Define a relation $\equiv_R$ as the set of all pairs (t, s) such that a tree witness for (R, t) exists and $f_{R,t}(s) = \varepsilon$. By Proposition 1, $\equiv_R$ is an equivalence relation (on its domain). By taking the quotient of $q|_{R,t}$ under $\equiv_R$, we obtain a tree reduct of $q|_{R,t}$ (cf. [17]). We call q a *quasi-tree with root* $t \in \mathsf{term}(q)$ if a tree witness for (R, t) exists for *all* directions R and $\bigcup_R \mathsf{dom}\, f_{R,t} = \mathsf{term}(q)$.

Proposition 2. *Suppose q is not a quasi-tree and tree witnesses exist for (R_1, t_1) and (R_2, t_2). If $f_{R_1,t_1}(t_2)$ is defined, $f_{R_1,t_1}(t_2) \neq \varepsilon$ then $\mathsf{dom}\, f_{R_2,t_2} \subsetneq \mathsf{dom}\, f_{R_1,t_1}$ and $f_{R_1,t_1}(s) = f_{R_1,t_1}(t_2) \cdot f_{R_2,t_2}(s)$, for all $s \in \mathsf{dom}\, f_{R_2,t_2}$.*

We are now in a position to introduce the ingredients of our polynomial rewriting. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $q(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$. Consider an atom $B(t)$ for a concept B. Then

$$\mathsf{ext}_B(x) \quad = \quad \bigvee_{\text{concept } B' \text{ s.t. } \mathcal{T}\models B'\sqsubseteq B} B'(x) \quad \vee \quad \bigvee_{\text{role } R \text{ s.t. } \mathcal{T}\models \exists R\sqsubseteq B} \exists w\, R(x,w)$$

gives the answers to $B(t)$ over ABox $\mathcal{A}$: for every $a \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{U}_\mathcal{K} \models B(a)$ iff $\mathcal{A} \models \mathsf{ext}_B(a)$. Note that, for all other elements σ in the domain $\Delta^{\mathcal{U}_\mathcal{K}}$ of $\mathcal{U}_\mathcal{K}$, we have $\mathcal{U}_\mathcal{K} \models B(\sigma)$ i ff$\mathcal{T} \models \exists T^- \sqsubseteq B$, where $\mathsf{tail}(\sigma) = c_T$.



Consider now an atom $R(t,t') \in q$ and the ways its terms can be mapped in $\mathcal{U}_\mathcal{K}$.
1. If both t and t' are mapped to ABox elements a, a' then $\mathcal{U}_\mathcal{K} \models R(a,a')$ i ff $R(a,a') \in \mathcal{A}$ because $\mathcal{U}_\mathcal{K}$ inherits the binary relations from $\mathcal{A}$.

2. If t is mapped to an ABox element a and t' to an 'anonymous' element in $\Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, then $R(t,t')$ can only be true if (i) $a \leadsto_\mathcal{K} c_R$, (ii) a tree witness for (R,t) exists, and (iii) $q|_{R,t}$ can be embedded into the sub-tree of $\mathcal{U}_\mathcal{K}$ beginning with the edge (a, ac_R); see the left-hand side of the picture above. Condition (i) can be defined by the formula

$$\mathsf{wt}_R(x) \quad = \quad \mathsf{ext}_{\exists R}(x) \wedge \neg\, \exists w\, R(x,w).$$

For all R and $a \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{A} \models \mathsf{wt}_R(a)$ i ff$a \leadsto_\mathcal{K} c_R$ (i.e., $ac_R \in \Delta^{\mathcal{U}_\mathcal{K}}$). For condition (iii), consider the conjunction $\mathsf{treeA}^q_{R,t}(x)$ of the formulas:

($\mathbf{t}_0$) $\mathsf{ext}_A(x)$, for all $A(s) \in q|_{R,t}$ with $f_{R,t}(s) = \varepsilon$;
($\mathbf{t}_1$) $\top$ if $\mathcal{T} \models \exists T^- \sqsubseteq A$ and $\bot$ otherwise, for all $A(s) \in q|_{R,t}$, $\mathsf{tail}(f_{R,t}(s)) = c_T$;
($\mathbf{t}_2$) $\top$ if $\mathcal{T} \models \exists T^- \sqsubseteq \exists S$ and $\bot$ otherwise, for $S(s,s') \in q|_{R,t}$, $\mathsf{tail}(f_{R,t}(s)) = c_T$.

One can show that $\mathcal{A} \models \mathsf{wt}_R(a) \wedge \mathsf{treeA}^q_{R,t}(a)$ i ff$\mathcal{U}_\mathcal{K} \models^\mathfrak{a} q|_{R,t}$ for an assignment $\mathfrak{a}$ such that $\mathfrak{a}(s) = a \cdot f_{R,t}(s)$, for all $s \in \mathsf{dom}\, f_{R,t}$.

3. If both t, t' are mapped to anonymous elements in $\Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, then two more cases need consideration.

3.1. Suppose first that there is a tree witness for some (S,s) such that s is mapped to an ABox element a with $a \leadsto_\mathcal{K} c_S$, (iv) both t and t' are in $\mathsf{dom}\, f_{S,s}$, and (v) all the terms $s' \in \mathsf{dom}\, f_{S,s}$ with $f_{S,s}(s') \neq \varepsilon$ are existentially quantified variables in q (only existential variables can be mapped to anonymous elements). In this case, as we observed above, $R(t,t')$ is true in $\mathcal{U}_\mathcal{K}$ if the formula

$$\mathsf{wt}_S(s) \wedge \mathsf{treeA}^q_{S,s}(s) \wedge \textstyle\bigwedge_{s \equiv_S s'} (s = s')$$

is true in $\mathcal{A}$ under an assignment $\mathfrak{a}$ such that $\mathfrak{a}(s') = a \cdot f_{S,s}(s')$, for $s' \in \mathsf{dom}\, f_{S,s}$. The disjunction of all such formulas for (S,s) satisfying (iv)–(v) depends only on the choice of terms t, t' and will be denoted by $\mathsf{attached\text{-}tree}_{t,t'}(\boldsymbol{x}, \boldsymbol{y})$. (This case is a generalisation of Case 2.)

3.2. Thus, it remains to consider the case (shown in the right-hand side of the picture) where the whole query is mapped to the anonymous part of $\mathcal{U}_\mathcal{K}$. Then q a quasi-tree and all terms in q are existentially quantified variables that are mapped to the sub-tree of $\mathcal{U}_\mathcal{K}$ generated by some ABox element a. More precisely, $a \in \mathsf{Ind}(\mathcal{A})$ generates a sequence of the form $a \leadsto_\mathcal{K} c_{R_1} \leadsto_\mathcal{K} \cdots \leadsto_\mathcal{K} c_{R_n}$, q has a root s (i.e., $\mathsf{term}(q) = \bigcup_S \mathsf{dom}\, f_{S,s}$), s is mapped to $\sigma = ac_{R_1} \cdots c_{R_n}$, while all other terms s' are mapped to $\sigma \cdot f_{S,s}(s')$. The latter condition can be captured by a formula similar to the one in the previous case. The difference is that now we begin with σ, $\mathsf{tail}(\sigma) = c_{R_n}$ (rather than a). To cope with this, consider the union q' of q and $\{R_n(v,s)\}$, for a fresh variable v, and let $\mathsf{treeT}^q_{c_{R_n},s}$ be $\mathsf{treeA}^{q'}_{R_n,v}$, where the tree witnesses are computed in query q'. Note that $\mathsf{treeT}^q_{c_{R_n},s}$ is a sentence because q' has no atoms for item $(\mathbf{t}_0)$. We denote by $\mathsf{detached\text{-}tree}$ the disjunction of sentences of the form

$$\exists w\, \mathsf{wt}_{R_1}(w) \land \mathsf{treeT}^q_{c_{R_n},s}$$

for all roots s of q and all pairs of roles R_1, R_n such that there are $R_2, \dots, R_{n-1}$ with $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_{i+1} \neq R_i^-$, for $1 \leq i < n$; if q is not a quasi-tree containing only existentially quantified variables, we set $\mathsf{detached\text{-}tree} = \bot$.

Denote by q^* the result of replacing each $A(t)$ and $P(t,t')$ in q with

$$A^*(t) = \mathsf{ext}_A(t) \lor \mathsf{attached\text{-}tree}_{t,t}(\boldsymbol{x},\boldsymbol{y}) \lor \mathsf{detached\text{-}tree},$$
$$P^*(t,t') = P(t,t') \lor \mathsf{attached\text{-}tree}_{t,t'}(\boldsymbol{x},\boldsymbol{y}) \lor \mathsf{detached\text{-}tree},$$

respectively. Note that these formulas depend not only on the predicate name but also on the terms in the atom. The length of q^* is $\mathcal{O}(|q|^2 \cdot |\mathcal{T}|^3)$ and can be made $\mathcal{O}(|q|^2 \cdot |\mathcal{T}|)$ if the sentence $\mathsf{detached\text{-}tree}$ is computed separately (in fact, for the majority of queries, e.g., queries with answer variables, it is simply $\bot$).

Theorem 5. *$\mathcal{U}_\mathcal{K} \models^\mathfrak{a} q(\boldsymbol{x})$ iff $\mathcal{A} \models^\mathfrak{a} q^*(\boldsymbol{x})$, whenever $\mathfrak{a}(x) \in \mathsf{Ind}(\mathcal{A})$ for all $x \in \boldsymbol{x}$.*

The rewriting above can also be adapted to *DL-Lite*$_{horn}$ and even *DL-Lite*$^\mathcal{N}_{horn}$ under the UNA. In this case, however, we need non-recursive Datalog programs to define the predicates $\mathsf{ext}_B(x)$; for details, see [14]. The non-recursive Datalog queries can be transformed to unions of CQs, but at the expense of exponential blowup. The problem whether a polynomial-size FO rewriting (without additional constants as in [12]) exists for *DL-Lite*$_{horn}$ is still open (and equivalent to the complexity problem ‘LogSpace = P?’).

5 Discussion

FO reducibility (or AC^0 data complexity) does not seem to provide enough information to judge whether a DL is suitable for OBDA. When measuring the complexity of query evaluation in database systems, it is usually assumed that queries are negligibly small compared to data. Thus, it makes sense to consider data complexity [24], which takes account of the data but ignores the query. A more subtle analysis [19] shows, however, that the obvious time $|q| \cdot |\mathcal{A}|^{|q|}$ required to check $\mathcal{A} \models q$ cannot be reduced to $f(|q|) \cdot p(|\mathcal{A}|)$, for any computable function f and polynomial p: $\mathrm{QE}(\mathcal{A}, q)$ is $W[1]$-complete for parameterised complexity, $|q|$ being a parameter. The success of database systems—despite fixed-parameter intractability—seems to imply that optimisation techniques are indispensable, that the 'real-world' queries are small and of 'special' form. In OBDA, the latter does not hold as the rewritten queries can be large and complex. However, data complexity does not differentiate among, e.g., $\mathit{DL\text{-}Lite}_{core}$, *OWL 2 QL* and the language of sticky sets of TGDs [5], all of which are in AC^0 for data complexity, while the primitive combined complexity, reflecting the size of the rewriting, ranges from P to NP and further to ExpTime. Another explanation of the database efficiency is that we only use queries with a bounded number of variables, in which case query evaluation is P-complete for combined complexity [25]. However, query rewritings may substantially increase the number of variables (for example, a CQ q is rewritten in [12] into a query with $\mathcal{O}(N \cdot \log N)$ auxiliary binary variables, where $N = |T| + |q|$).

The W3C recommendation (`www.w3.org/TR/owl2-profiles`) for OBDA is to reduce it to query evaluation in database systems. Two drawbacks of this recommendation are that it (*i*) disregards the complexity of possible reductions, and (*ii*) excludes some useful DLs from consideration. As we saw above, rewritings of CQs in *OWL 2 QL* cannot be done in polynomial time without adding extra constants, variables and quantifiers as in [12]. One might argue that, in the real-world ontologies, role inclusions do not interact with inverse roles in as sophisticated way as in Theorem 1, but then more research is needed to support this argument. A number of 'lightweight' DLs such as $\mathcal{ELH}$ or $\mathit{DL\text{-}Lite}_{horn}^{(\mathcal{HF})}$ [2] are deemed not suitable for OBDA because they are P-complete for data complexity. Recall that both of these logics are P-complete for primitive combined complexity (vs. NP in the case of *OWL 2 QL*). The combined approach to OBDA [18, 14, 15] resolves this issue by expanding the data at a pre-processing step and then rewriting and answering CQs. The expansion is linear in $|\mathcal{A}|$ and can be done by the database system itself; the size of the rewritten query for $\mathcal{EL}$ and $\mathit{DL\text{-}Lite}_{horn}^{\mathcal{F}}$ is only quadratic (for *OWL 2 QL*, it is still exponential).

In this paper, we do not touch on the problem of representing ABoxes in database systems, where usually GLAV mappings are used to connect data sources to ontologies. Such mappings introduce some problems as tuples in the same relation can come from different data sources. Also, they provide certain information on the completeness of concepts and roles, which can (and should) be exploited in order to minimise the rewritings [22]. Finally, with so many languages and rewritings for OBDA suggested, it looks like the time is ripe for comprehensive experiments that could clarify the future of OBDA with DLs.

Acknowledgments: supported by the U.K. EPSRC grant EP/H05099X/1.

References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: QUerying ONTOlogies. In: Proc. AAAI, 1670–1671 (2005)

2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artificial Intelligence Research 36, 1–69 (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) In: Proc. OWLED DC (2008)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
5. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. In: Proc. VLDB 3(1), 554–565 (2010)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. KR. pp. 260–270 (2006)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Automated Reasoning 39(3), 385–429 (2007)
8. Dasgupta, S., Papadimitriou, C., Vazirani, U.V.: Algorithms. McGraw-Hill (2008)
9. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Proc. ISWC. LNCS, vol. 5318, pp. 403–418. Springer (2008)
10. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in description logics: The knots approach. In: Proc. WoLLIC. LNCS, vol. 5514. Springer (2009)
11. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. ICDE. (2011)
12. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive Datalog programs. In: Proc. DL. (2011)
13. Heymans, S., Ma, L., *et al.*: Ontology reasoning with large data repositories. In: Ontology Management, pp. 89–128. Springer (2008)
14. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. KR (2010)
15. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Proc. IJCAI (2011)
16. Lutz, C.: Inverse roles make conjunctive queries hard. In: Proc. DL. CEUR Workshop Proceedings, vol. 250. (2007)
17. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. IJCAR. pp. 179–193. LNAI, vol. 5195. Springer (2008)
18. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. IJCAI. pp. 2070–2075 (2009)
19. Papadimitriou, C.H., Yannakakis, M.: On the complexity of database queries. J. Comput. Syst. Sci. 58(3), 407–427 (1999)
20. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for *DL-Lite*. In: Proc. DL. CEUR Workshop Proceedings, vol. 477. (2009)
21. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
22. Rodríguez-Muro M., Calvanese, D.: Dependencies to optimize ontology-based data access. In: Proc. DL. (2011)
23. Rosati, R., Almatelli, A.: Improving query answering over *DL-Lite* ontologies. In: Proc. KR (2010)
24. Vardi, M.: The complexity of relational query languages (extended abstract). In: Proc. STOC. pp. 137–146 (1982)
25. Vardi, M.: On the complexity of bounded-variable queries (extended abstract). In: Proc. PODS. pp. 266–276 (1995)

Two-Dimensional Description Logics of Context

Szymon Klarman[1] and Víctor Gutiérrez-Basulto[2]

[1] Department of Computer Science, Vrije Universiteit Amsterdam
sklarman@few.vu.nl
[2] Department of Computer Science, Universität Bremen
victor@informatik.uni-bremen.de

Abstract. We introduce an extension of Description Logics (DLs) for representing and reasoning about contextualized knowledge. Our formalism is inspired by McCarthy's theory of formalizing contexts and based on two-dimensional semantics, with one dimension representing a usual object domain and the other a domain of contexts. Additionally, it is equipped with a second DL language for describing the context domain. As a result, we obtain a family of *two-sorted*, *two-dimensional* combinations of pairs of DLs.

1 Introduction

Description Logics (DLs) provide a clear and broadly accepted paradigm for reasoning about terminological knowledge. Under the standard Kripkean semantics, a DL ontology forces a unique, global view on the represented world, in which the ontology axioms are interpreted as universally true. This philosophy is well-suited as long as everyone can share the same conceptual perspective on the domain or there is no need for considering alternative viewpoints. Alas, this is hardly ever the case since a domain can be modeled differently depending on the intended use of an ontology. Consequently, effective representation and reasoning about knowledge pertaining to such multiple, heterogenous viewpoints becomes the primary objective for many practical applications [1,2].

The challenges above resemble clearly those problems that originally inspired J. McCarthy to introduce a theory of formalizing contexts in knowledge representation systems, as a way of granting them more generality [3,4]. The gist of his proposal is to replace logical formulas φ, as the basic knowledge carriers, with assertions $ist(\boldsymbol{c}, \varphi)$ stating that φ is true in $\boldsymbol{c}$, where $\boldsymbol{c}$ denotes an abstract first-order entity called a *context*, which on its own can be described in a first-order language. For instance:

$$ist(\boldsymbol{c}, Heart(a)) \land \boldsymbol{HumanAnatomy}(\boldsymbol{c})$$

states that the object a is a heart in some context described as $\boldsymbol{HumanAnatomy}$. Based on this foundation, the theory advocates complex models of knowledge which are able to properly account for the local, context-specific scope of the represented knowledge, while at the same time provide an expressive apparatus for modeling semantic interoperability of contexts, i.e. generic rules guiding the information flow between different contexts.

The importance of contextualized knowledge in DLs has been generally acknowledged, nevertheless the framework is still not supported with a dedicated theory of handling context-dependent information. In this direction, the most commonly considered perspectives are restricted to global integration of local ontologies [5,6] or modeling levels of abstraction as subsets of models of a DL ontology [7,8]. The purpose of this paper is to introduce a novel extension of DLs for reasoning with contextualized knowledge. Our proposal is systematically derived from two formal roots. On the one hand, by resorting to McCarthy's theory we ground our approach in a longstanding tradition of formalizing contexts in AI. On the other, we build on top of two-dimensional DLs [9], which provide us with well-understood formal foundations. In particular, we extend the standard DL semantics with a second modal dimension, representing a possibly infinite domain of contexts. Additionally, our logics are equipped with a second DL language for describing the context domain. This way we obtain a family of *two-sorted*, *two-dimensional* combinations of pairs of DLs for reasoning about contextualized knowledge.

This paper is the workshop version of [10] and [11]. It extends the work presented there by discussing thoroughly the motivation underlying the formal design of the introduced DLs of contexts. We also review a number of expressive fragments of these logics and report the corresponding complexity results obtained and proven in the two papers.

2 Overview and formal motivation

Since its introduction, McCarthy's theory of formalizing contexts has inspired a significant body of work in AI studying implementations of the approach in a variety of formalisms and applications [12,13,14,4,1,15]. The great appeal of this theory stems from the simplicity of the three major postulates it is based on:

1. Contexts are formal objects. More precisely, a *context* is anything that can be denoted by a first-order term and used meaningfully in a statement of the form $ist(\boldsymbol{c}, \varphi)$, saying that formula φ *is true* (*ist*) in context $\boldsymbol{c}$ [3,4,12].

2. Contexts have properties and can be described. As first-order objects, contexts can be in a natural way described in a first-order language [14,4]. This allows for addressing them generically through quantified formulas such as $\forall x(\boldsymbol{C}(x) \to ist(x,\varphi))$, expressing that φ is true in every context of type $\boldsymbol{C}$.

3. Contexts are organized in relational structures. In the commonsense reasoning, contextual assumptions are dynamically and directionally altered [15,12], thus contexts are often accessed from other contexts. Formally, this can be captured by allowing nestings of the form $ist(\boldsymbol{c}, ist(\boldsymbol{d}, \varphi))$.

The logics proposed in this paper originate as an attempt of adopting these principles in the framework of DLs. In the following paragraphs we discuss the central design choices we made and the motivation behind them. We start from the basic semantic considerations on contexts and further trace their impact on the selection of specific logical languages.

The key to importing McCarthy's theory into a knowledge representation framework is a faithful interpretation of his three postulates on the model-theoretic grounds of the framework. By doing so within the DL paradigm, we effectively commit ourselves to a specific sort of semantic structures that must be taken into account in order to express and interpret contextualized knowledge adequately. Figure 1 illustrates one such structure — a formal model of some application domain supporting multiple contexts of representation. As an intuitive example, consider here a formal description of a society of interconnected agents, each one sustaining his own viewpoint and focus on the represented world.

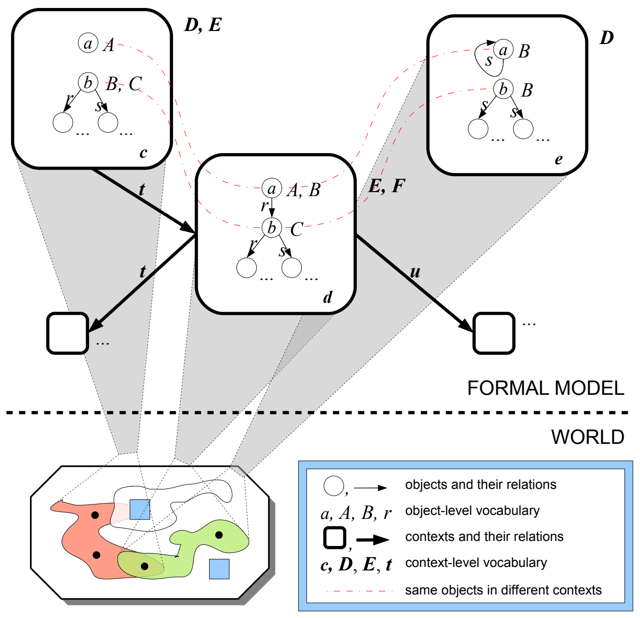


Fig. 1. A formal model of an application domain complying to McCarthy's principles.

The model has two apparent levels. The *context-level* consists of context entities (postulate 1), which are possibly interlinked with accessibility relations (postulate 3) and described in a language containing individual names, concepts and relation names (postulate 2). For instance, context $\boldsymbol{c}$ is of type $\boldsymbol{D}$ and is related to $\boldsymbol{d}$ through the relation $\boldsymbol{t}$. Intuitively, each context in the model can be seen as a box carrying a piece of the *object-level* representation. Clearly, instead of a unique global model of the object domain, we associate a single local model with every context. Naturally, these models might obviously differ from each other as each of them reflects a specific viewpoint on the object domain. Moreover, they might not necessarily cover the same fragment and aspects of the application domain and not necessarily use the same fragment of the object language for describing it. For instance, objects a and b occur at the same time in contexts $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, but in each of them they are described differently and remain in different relations to other objects.

The central insight emerging from this short analysis is that the semantic structures comprising the model theory of a reasonably expressive DL of context are inherently *two-dimensional*, with one dimension consisting of (domain) *objects* and the second — *contexts*.

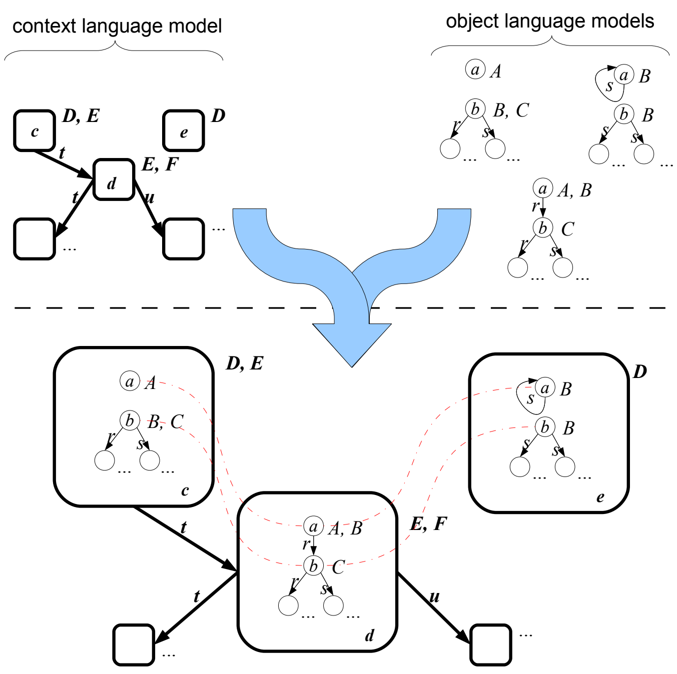


Fig. 2. Combining models of two DLs.

Once we have identified the main characteristics of the intended semantic structures, the next step is to find convenient languages for speaking about them, and constraining their possible properties. By the assumption taken in this work, DLs are suitable formalisms for representing the object-level knowledge. The key challenge is then to extend them with additional syntactic means that would facilitate accommodating the context-level information. A first crucial observation in this direction is that contexts and their relations, as pictured above, correspond to Kripke frames, with possible worlds interpreted as context entities. It is commonly known that such frames can be combined in a product-like fashion with the standard DL interpretations, giving rise to two-dimensional semantics for DLs with additional modal operators [16]. These operators are typically intended for modeling the evolution of the object knowledge across the states of the second dimension, for instance time points, as in temporal DLs [17]. Although this approach seems in general very encouraging, the caveat is that it does not offer a direct methodology for describing the elements of the second dimension. More precisely, we can easily augment a DL language with modal '*contextualization*' operators $\Diamond, \Box$ for traversing the context dimension of the models and quantifying over implicit context objects, but it is not clear how to explicitly assert properties of the accessed contexts — an essential point for obtaining a fine-grained contextualization machinery.

As a solution, we employ a second DL language for describing the context dimension. Thus, we obtain a *two-sorted* language with each sort interpreted over the respective dimension. The two languages are suitably integrated on the syntactic and semantic level, so that their models can be eventually combined as presented in Figure 2. The style of combination remains fully compatible with that underlying two-dimensional DLs described above. In fact, we are able to show that, depending on the choice of the integration mechanism, our logics are proper extensions of the well-known $(\mathbf{K}_n)_{\mathcal{L}}$ or $\mathbf{S5}_{\mathcal{L}}$ [16].

In the following sections, we first recap the basic DL nomenclature, next we formally define the syntax and semantics of the proposed DLs of context and give an overview of their expressiveness–complexity characterization. Finally, we consider intended application scenarios for the framework.

3 Description Logics of Context

A *DL language* $\mathcal{L}$ is specified by a vocabulary $\Sigma = (N_C, N_R, N_I)$, where N_C is a set of *concept names*, N_R a set of *role names* and N_I a set of *individual names*, and a number of constructors for composing complex expressions. In this paper, we focus on the well-known DLs $\mathcal{EL}, \mathcal{ALC}$ and $\mathcal{ALCO}$ [18,19] and assume the reader is familiar with those formalism and the basic notions concerning DLs.

A *Description Logic of Context* $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ consists of a DL context language $\mathcal{L}_C$, supporting context descriptions, and an object language $\mathcal{L}_O$ equipped with context operators for representing object knowledge relative to contexts. We introduce two families of such DLs, characterized by different types of context operators.

Definition 1 ($\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-context language). *The* context language *of* $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ *is a DL language* $\mathcal{L}_C$ *over the vocabulary* $\Gamma = (M_C, M_R, M_I)$*, with a designated subset* $M_I^\star \subseteq M_I$*.*

The set $M_I^\star$ contains context names. Following some common-sense intuitions, we consider contexts only as a subset of the domain of the context language. Indeed, certain elements of this domain might carry no object knowledge at all, and instead, serve only as individuals referred to in context descriptions (cf. Figure 1). This is often the case in applications concerned with provenance of knowledge [2]. For instance, a context $\boldsymbol{c}$, associated with a single knowledge source, might be there described with an axiom $\boldsymbol{hasAuthor}(\boldsymbol{c}, \boldsymbol{henry})$, where $\boldsymbol{henry}$ is an individual related to $\boldsymbol{c}$, but obviously not a context *per se.*

Definition 2 ($\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-object language). *Let $\mathcal{L}_O$ be a DL language over the vocabulary $\Sigma = (N_C, N_R, N_I)$. The* object language *of $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ is the smallest language containing $\mathcal{L}_O$ and closed under the constructors of $\mathcal{L}_O$ and one of the two types — $\mathfrak{F}_1$* resp. *$\mathfrak{F}_2$ — of concept-forming operators:*

$$\langle \boldsymbol{r}.\boldsymbol{C} \rangle\, D \ \mid\ [\boldsymbol{r}.\boldsymbol{C}]\, D \qquad (\mathfrak{F}_1)$$

$$\langle \boldsymbol{C} \rangle D \ \mid\ [\boldsymbol{C}] D \qquad (\mathfrak{F}_2)$$

where $\boldsymbol{C}$ and $\boldsymbol{r}$ are a concept and a role of the context language and D is a concept of the object language.

Intuitively, the concept $\langle \boldsymbol{r}.\boldsymbol{C} \rangle D$ denotes all objects which are D in *some* context of type $\boldsymbol{C}$ accessible from the current one through $\boldsymbol{r}$. Similarly, $[\boldsymbol{r}.\boldsymbol{C}]D$ denotes all objects which are D in *every* such context. In the case of the operators in $\mathfrak{F}_2$, the concept $\langle \boldsymbol{C} \rangle D$ denotes all objects which are D in *some* context of type $\boldsymbol{C}$, whereas $[\boldsymbol{C}]D$ — all objects which are D in *every* such context. For example, $\langle \boldsymbol{neighbor}.\boldsymbol{Country} \rangle \mathit{Citizen}$, refers to the concept *Citizen* in some context of type ***Country*** accessible through the ***neighbor*** relation from the current context. Analogically, $\langle \boldsymbol{HumanAnatomy} \rangle \mathit{Heart}$ refers to the concept *Heart* in some context of ***HumanAnatomy***.

Definition 3 ($\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-knowledge base). *A $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-*knowledge base (CKB) *is a pair $\mathcal{K} = (\mathcal{C}, \mathcal{O})$, where $\mathcal{C}$ is a set of axioms of the context language in any of the forms* (†)*, and $\mathcal{O}$ is a set of formulas of the form:*

$$\boldsymbol{c} : \varphi \mid \boldsymbol{C} : \varphi$$

where φ is an axiom of the object language (a GCI or a concept/role assertion), $\boldsymbol{c} \in M_I^\star$ and $\boldsymbol{C}$ is a concept of the context language.

A formula $\boldsymbol{c} : \varphi$ states that the axiom φ holds in the context denoted by the name $\boldsymbol{c}$. Note that this corresponds directly to McCarthy's $\mathit{ist}(\boldsymbol{c}, \varphi)$. Axioms of the form $\boldsymbol{C} : \varphi$ assert the truth of φ in all contexts of type $\boldsymbol{C}$. For example, the formula $\boldsymbol{Country} : \langle \boldsymbol{neighbor}.\boldsymbol{Country} \rangle \mathit{Citizen} \sqsubseteq \mathit{NoVisaRequirement}$ states that in every country, the citizens of its neighbor countries do not require visas.

The semantics is given through $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$*-interpretations* and $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$*-models*, which combine the interpretations of $\mathcal{L}_C$ with those of $\mathcal{L}_O$. We assume the semantics of $\mathcal{EL}, \mathcal{ALC}$ and $\mathcal{ALCO}$ to be defined in the standard way[18,19]. As explained before, the (possibly infinite) domain of contexts $\mathfrak{C}$ is subsumed by the entire interpretation domain of the context language Θ. For technical reasons, we assume a constant object domain Δ for all contexts. This assumption, though often impractical, grants greater generality to the complexity results and can be easily relaxed to the varying domain case.

Definition 4 ($\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-interpretations). *A $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-interpretation is a tuple $\mathfrak{M} = (\Theta, \mathfrak{C}, \cdot^{\mathcal{J}}, \Delta, \{\cdot^{\mathcal{I}(i)}\}_{i \in \mathfrak{C}})$, where:*

1. *$(\Theta, \cdot^{\mathcal{J}})$ is an interpretation of the context language, where Θ is a non-empty domain of individuals and $\cdot^{\mathcal{J}}$ an interpretation function, where:*

  – *$\mathfrak{C} \subseteq \Theta$ is a non-empty domain of contexts,*
  – *$\boldsymbol{c}^{\mathcal{J}} \in \mathfrak{C}$, for every $\boldsymbol{c} \in M_I^\star$,*

2. *$(\Delta, \cdot^{\mathcal{I}(i)})$, for every $i \in \mathfrak{C}$, is an interpretation of the object language, where Δ is a non-empty object domain and $\cdot^{\mathcal{I}(i)}$ an interpretation function of $\mathcal{L}_O$, such that:*
   ($\mathfrak{F}_1$) *for every $\langle \boldsymbol{r}.\boldsymbol{C} \rangle D$ and $[\boldsymbol{r}.\boldsymbol{C}]D$:*
   - $(\langle \boldsymbol{r}.\boldsymbol{C} \rangle\, D)^{\mathcal{I}(i)} = \{x \mid \exists j \in \mathfrak{C} : (i,j) \in \boldsymbol{r}^{\mathcal{J}} \wedge j \in \boldsymbol{C}^{\mathcal{J}} \wedge x \in D^{\mathcal{I}(j)}\}$,
   - $([\boldsymbol{r}.\boldsymbol{C}]\, D)^{\mathcal{I}(i)} = \{x \mid \forall j \in \mathfrak{C} : (i,j) \in \boldsymbol{r}^{\mathcal{J}} \wedge j \in \boldsymbol{C}^{\mathcal{J}} \rightarrow x \in D^{\mathcal{I}(j)}\}$.

   ($\mathfrak{F}_2$) *for every $\langle \boldsymbol{C} \rangle D$ and $[\boldsymbol{C}]D$:*
   - $(\langle \boldsymbol{C} \rangle D)^{\mathcal{I}(i)} = \{x \mid \exists j \in \mathfrak{C} : j \in \boldsymbol{C}^{\mathcal{J}} \wedge x \in D^{\mathcal{I}(j)}\}$,
   - $([\boldsymbol{C}]D)^{\mathcal{I}(i)} = \{x \mid \forall j \in \mathfrak{C} : j \in \boldsymbol{C}^{\mathcal{J}} \rightarrow x \in D^{\mathcal{I}(j)}\}$.

Clearly, the difference between the context operators of type $\mathfrak{F}_1$ and $\mathfrak{F}_2$ lies in the choice of the relational structures involved in quantifying over the context domain. $\mathfrak{F}_1$-operators bind contexts only along the roles of the context language (similarly to **K**-modalities), while $\mathfrak{F}_2$-operators ignore these relationships and rest upon the universal relation over $\mathfrak{C}$ (similarly to **S5**-modalities). This is reflected in the scope and the character of the distribution of the object knowledge over contexts in $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-models. For instance, in Figure 1, the concept $\langle \boldsymbol{t}.\boldsymbol{F} \rangle B$ is satisfied by object a only in context $\boldsymbol{c}$, while $\langle \boldsymbol{F} \rangle B$ is satisfied by a in all contexts in the model. From the perspective of McCarthy's theory, employing operators $\mathfrak{F}_2$, rather than $\mathfrak{F}_1$, is equivalent to scarifying postulate (3). This means that every two contexts in the model become in principle accessible to each other.

Definition 5 ($\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-models). *A $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-interpretation $\mathfrak{M} = (\Theta, \mathfrak{C}, \cdot^{\mathcal{J}}, \Delta, \{\cdot^{\mathcal{I}(i)}\}_{i \in \mathfrak{C}})$ is a* model *of a CKB $\mathcal{K} = (\mathcal{C}, \mathcal{O})$* iff

– *for every $\varphi \in \mathcal{C}$, $(\Theta, \cdot^{\mathcal{J}})$ satisfies φ,*
– *for every $\boldsymbol{c} : \varphi \in \mathcal{O}$, $(\Delta, \cdot^{\mathcal{I}(\boldsymbol{c}^{\mathcal{J}})})$ satisfies φ,*
– *for every $\boldsymbol{C} : \varphi \in \mathcal{O}$ and $i \in \mathfrak{C}$, if $i \in \boldsymbol{C}^{\mathcal{J}}$ then $(\Delta, \cdot^{\mathcal{I}(i)})$ satisfies φ.*

As hinted before, there is a close connection between our DLs of context and the modal DLs $(\mathbf{K}_n)_{\mathcal{L}}$ and $\mathbf{S5}_{\mathcal{L}}$. In particular, the former are proper extensions of $(\mathbf{K}_n)_{\mathcal{L}}$ *resp.* $\mathbf{S5}_{\mathcal{L}}$. This relationship is formally established in Theorem 1.

Theorem 1. *If $M_I = M_C = \emptyset$ then $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ with context operators (only) of type $\mathfrak{F}_1$* resp. *$\mathfrak{F}_2$ is a notational variant of $(\mathbf{K}_n)_{\mathcal{L}_O}$* resp. *$\mathbf{S5}_{\mathcal{L}_O}$ with global axioms.*

Proof sketch. Observe that all formulas are of the form $\top : \varphi$. First, replace every $\langle \boldsymbol{r}.\top \rangle$ with $\Diamond_{\boldsymbol{r}}$ and $[\boldsymbol{r}.\top]$ with $\Box_{\boldsymbol{r}}$, *resp.* every $\langle \top \rangle$ with $\Diamond$ and $[\top]$ with $\Box$. Next, replace every $\top : \varphi \in \mathcal{O}$ with φ. It is easy to see that the semantics of $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ coincides with that of $(\mathbf{K}_n)_{\mathcal{L}_O}$ *resp.* $\mathbf{S5}_{\mathcal{L}_O}$. Note that an axiom is global *iff* it is satisfied in all possible $\mathbf{K}_n$-worlds *resp.* **S5**-worlds. ❑

As our main technical contributions in [10] and [11], we obtained a wide panorama of complexity results for reasoning in DLs of context using particular combinations of DLs for $\mathcal{L}_C$ and $\mathcal{L}_O$, and different types of context operators. We summarize these results in Table 1, and shortly elaborate on them below.

context operators	$\mathcal{L}_O$ \ $\mathcal{L}_C$	$\mathcal{EL}$	$\mathcal{ALC}, \mathcal{ALCO}$
type $\mathfrak{F}_1$	$\mathcal{ALC}, \mathcal{ALCO}$	2ExpTime-complete	2ExpTime-complete
type $\mathfrak{F}_2$	$\mathcal{EL}$	PTime	ExpTime-hard
	$\mathcal{ALC}$	ExpTime-complete	NExpTime-complete
	$\mathcal{ALCO}$	NExpTime-complete	NExpTime-complete

Table 1. Complexity of reasoning in $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$.

The results reveal that the computational properties of the proposed logics are predominantly affected by the choice of the context operators. More precisely, reasoning in $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ with $\mathfrak{F}_1$-operators is harder than with $\mathfrak{F}_2$-operators. This behavior can be explained by the fact that such difference in the complexity is essentially present already between the underlying logics $(\mathbf{K}_n)_{\mathcal{L}}$ and $\mathbf{S5}_{\mathcal{L}}$ [10,20].

In the case of DLs of context with $\mathfrak{F}_1$-operators, we first established the 2ExpTime lower bound for the satisfiability problem for $(\mathbf{K}_n)_{\mathcal{ALC}}$ w.r.t. to global TBoxes and only local roles. The proof is a reduction of the *word problem* for an exponentially bounded, alternating Turing machine. This result turned out to be quite surprising since it could be expected that without rigid roles the satisfiability problem can be straightforwardly reduced to satisfiability in fusion models. This in turn would have to yield an ExpTime upper bound by means of the standard techniques. However, as the following example for $(\mathbf{K}_n)_{\mathcal{ALC}}$ demonstrates, this strategy fails.

$$(\dagger)\ \Diamond_i C \sqcap \exists r.\Box_i \bot \qquad (\ddagger)\ \exists succ_i.C \sqcap \exists r.\forall succ_i.\bot$$

Although (†) clearly does not have a model, its reduction (‡) to a fusion language, where modal operators are translated to restrictions on fresh $\mathcal{ALC}$ roles, is satisfiable. The reason is that while in the former case the information about the structure of the **K**-frame is global for all individuals, in the latter it becomes local. The r-successor in (‡) is simply not 'aware' that it should actually have a $succ_i$-successor. The matching 2ExpTime upper bound is proven by using the *quasistate elimination* technique, similar to the proofs for certain products of modal logics [9].

Regarding DLs of context with $\mathfrak{F}_2$-operators, for $\mathcal{L_O} \in \{\mathcal{ALC}, \mathcal{ALCO}\}$ and $\mathcal{L_C} \in \{\mathcal{ALC}, \mathcal{ALCO}\}$, we encounter a jump from ExpTime to NExpTime-completeness. The non-determinism involved can be interpreted by the need of guessing the interpretation of the context language first, before finding the model of the object component of the combination. In particular, the lower bound is obtained by an encoding of the $2^n \times 2^n$ tiling problem, known to be NExpTime-complete [9]. In the case of $\mathcal{L_O} = \mathcal{ALCO}$ and $\mathcal{L_C} = \mathcal{EL}$ this jump can be explained by the interaction of nominals and the context operators, in fact this enables to encode the $2^n \times 2^n$ tiling problem, as in the previous cases. For the upper bounds for $\mathcal{L_O} \in \{\mathcal{ALC}, \mathcal{ALCO}\}$ we devise a variant of a *type elimination algorithm*, whereas for $\mathcal{L_O} = \mathcal{EL}$ a *completion algorithm* in the style of [21].

4 Application scenarios

There are two natural application scenarios for the DLs of context. First, they can be used as native representation languages dedicated to modeling and reasoning about knowledge of inherently contextualized nature. Alternatively, the framework can be used to support an external 'integration' layer over standard DL ontologies. Observe, that a collection of DL ontologies $\mathcal{O}_1, \ldots, \mathcal{O}_n$ in some language $\mathcal{L}_\mathcal{O}$ can be seen as a set of formulas $\mathcal{O} = \{\boldsymbol{c}_i : \varphi \mid \varphi \in \mathcal{O}_i,\ i \in (1, n)\}$ in $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$, where every ontology corresponds to a unique context name. Consequently, the extra expressive power of $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ can be utilized for imposing interoperability constraints over those ontologies. Arguably, the first type of use might be of interest for knowledge-intensive/expert applications, while the second one seems appealing from the perspective of integrating information on the Semantic Web. We support the two cases with small examples, based on different types of context operators, and explain some possible inferences.

Contextualized knowledge base. Consider a simple representation of knowledge about the legal status of people, contextualized with respect to geographic locations. We define a CKB $\mathcal{K} = (\mathcal{C}, \mathcal{O})$, consisting of the context (geographic) ontology $\mathcal{C}$ and the object (people) ontology $\mathcal{O}$, as follows:

$\mathcal{C}$:	$\boldsymbol{Country}(\boldsymbol{germany})$	(1)
	$\boldsymbol{neighbor}(\boldsymbol{france}, \boldsymbol{germany})$	(2)
$\mathcal{O}$:	$\boldsymbol{germany} : \exists hasParent.Citizen(john)$	(3)
	$\boldsymbol{Country} : \exists hasParent.Citizen \sqsubseteq Citizen$	(4)
	$\boldsymbol{france} : \langle \boldsymbol{neighbor}.\boldsymbol{Country} \rangle\, Citizen \sqsubseteq NoVisaRequirement$	(5)

Visibly, ***france*** and ***germany*** play here the role of contexts, described in the context language by axioms (1) and (2). In the context of ***germany***, it is known that *john* has a parent who is a citizen (3). Since in every ***Country*** context — thus including ***germany*** — the concept $\exists hasParent.Citizen$ is subsumed by *Citizen* (4), therefore it must be true that *john* is an instance of *Citizen* in ***germany***. Finally, since ***germany*** is related to ***france*** via the role ***neighbor***, it follows that *john* (assuming rigid interpretation of this name across contexts) has to be an instance of *NoVisaRequirement* in the context of ***france*** (5). A sample $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-model of $\mathcal{K}$ is depicted in Figure 3.

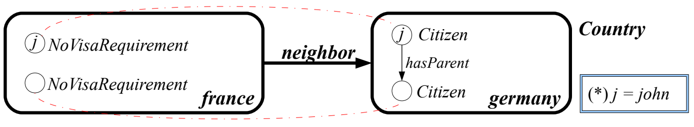


Fig. 3. A $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$-model of the CKB $\mathcal{K}$.

Interoperability constraints over DL ontologies. Consider an architecture such as the NCBO BioPortal project[3], which gathers numerous published bio-health ontologies, and categorizes them via thematic tags, e.g.: $\boldsymbol{Cell}$, $\boldsymbol{Health}$, $\boldsymbol{Anatomy}$, etc., organized in a meta-ontology. The intention of the project is to facilitate the reuse of the collected resources in new applications. Note, that the division between the context and the object language is already present in the architecture of the BioPortal, which can be immediately utilized to state, e.g.:

$$\begin{array}{lll} \mathcal{C}: & \boldsymbol{HumanAnatomy} \sqsubseteq \boldsymbol{Anatomy} & (1) \\ \mathcal{O}: & \top : \langle \boldsymbol{HumanAnatomy} \rangle Heart \sqsubseteq [\boldsymbol{Anatomy}] HumanHeart & (2) \\ & \boldsymbol{Anatomy} : Heart \sqsubseteq Organ & (3) \end{array}$$

where (2) maps the concept *Heart* from any $\boldsymbol{HumanAnatomy}$ ontology to the concept *HumanHeart* in every $\boldsymbol{Anatomy}$ ontology; (3) imposes the axiom $Heart \sqsubseteq Organ$ of an upper anatomy ontology over all $\boldsymbol{Anatomy}$ ontologies, which due to axiom (1) carries over to all $\boldsymbol{HumanAnatomy}$ ontologies.

In general, $\mathfrak{C}_{\mathcal{L}_O}^{\mathcal{L}_C}$ provides logic-based explications of some interesting notions, relevant to the problem of semantic interoperability of ontologies, such as:

concept alignment: $\top : \langle \boldsymbol{A} \rangle C \sqsubseteq [\boldsymbol{B}] D$
every instance of C in any ontology of type $\boldsymbol{A}$ is D in every ontology of type $\boldsymbol{B}$

semantic importing: $\boldsymbol{c} : \langle \boldsymbol{A} \rangle C \sqsubseteq D$
every instance of C in any ontology of type $\boldsymbol{A}$ is D in ontology $\boldsymbol{c}$

upper ontology axiom: $\boldsymbol{A} : C \sqsubseteq D$
axiom $C \sqsubseteq D$ holds in every ontology of type $\boldsymbol{A}$

5 Conclusions

The problems of 1) representing inherently contextualized knowledge within the paradigm of DLs and 2) reasoning with multiple heterogenous, but semantically interoperating DL ontologies, are both interesting and important issues, motivated by numerous practical application scenarios. It is our belief that these two challenges are in fact two sides of the same coin and, consequently, they should be approached within the same, unifying formal framework. In this paper, we have proposed two novel families of two-dimensional DLs of context. Arguably, these logics achieve the objective declared above to a great extent, by providing sufficient syntactic and semantic means to support both functionalities, seamlessly integrated within one formalism.

As our results show, such two-dimensional extension of the DL framework does not necessarily entail an increase in the computational complexity of reasoning, as for e.g. $\mathfrak{C}_{\mathcal{EL}}^{\mathcal{EL}}$ and $\mathfrak{C}_{\mathcal{ALC}}^{\mathcal{EL}}$ with $\mathfrak{F}_2$-operators, nor does it affect the generally adopted knowledge representation methodology of DLs. We therefore consider the approach a worthwhile subject to further research. In particular, it is essential to investigate how certain notions and problems central to the practical use and maintenance of multi-context knowledge systems (e.g. handling local inconsistencies) can be meaningfully restated within the presented framework.

[3] See http://bioportal.bioontology.org/.

Acknowledgements We want to thank Carsten Lutz and Stefan Schlobach for many helpful discussions and suggestions on the ideas presented in this paper. We are also grateful to the reviewers of this work for their useful feedback.

References

1. Guha, R., McCool, R., Fikes, R.: Contexts for the semantic web. In: Proc. of ISWC-04. (2004) 32–46
2. Bao, J., Tao, J., McGuinness, D.L., Smart, P.: Context representation for the semantic web. Proc. of Web Science Conference (2010)
3. McCarthy, J.: Generality in artificial intelligence. Communications of the ACM **30** (1987) 1030–1035
4. Guha, R.: Contexts: a formalization and some applications. PhD thesis, Stanford University (1991)
5. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics **1** (2003) 2003
6. Cuenca Grau, B., Kutz, O.: Modular ontology languages revisited. In: Proc. of the Workshop on Sem. Web for Collaborative Know. Acquisition. (2007)
7. Goczyla, K., Waloszek, W., Waloszek, A.: Contextualization of a DL knowledge base. In: Proc. of DL-07. (2007)
8. Grossi, D.: Desigining Invisible Handcuffs. Formal Investigations in Institutions and Org. for Multi-Agent Systems. PhD thesis, Utrecht University (2007)
9. Kurucz, A., Wolter, F., Zakharyaschev, M., Gabbay, D.M.: Many-Dimensional Modal Logics: Theory and Applications. Elsevier (2003)
10. Klarman, S., Gutiérrez-Basulto, V.: $\mathcal{ALC}_{\mathcal{ALC}}$: A context description logic. In: Proc. of JELIA-10. (2010)
11. Klarman, S., Gutiérrez-Basulto, V.: Two-dimensional description logics for context-based semantic interoperability. In: Proc. of AAAI-11. (2011)
12. Buvač, S., Mason, I.A.: Propositional logic of context. In: Proc. of AAAI-93. (1993) 412–419
13. Buvač, S., Buvac, V., Mason, I.A.: Metamathematics of contexts. Fundamenta Informaticae **23** 412–419
14. Buvač, S.: Quantificational logic of context. In: Proc. of AAAI-96. (1996) 412–419
15. Nossum, R.: A decidable multi-modal logic of context. Journal of Applied Logic **1**(1-2) (2003) 119 – 133
16. Wolter, F., Zakharyaschev, M.: Multi-dimensional description logics. In: Proc. of IJCAI-99, San Francisco, CA, USA (1999) 104–109
17. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Proc. of TIME-08. (2008)
18. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI-05. (2005)
19. Baader, F., Calvanese, D., Mcguinness, D.L., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. (2003)
20. Artale, A., Lutz, C., Toman, D.: A description logic of change. In Veloso, M., ed.: Proc. of IJCAI-07. (2007) 218–223
21. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In: Proc. of KR-10. (2010)

Query Answering over $\mathcal{SROIQ}$ Knowledge Bases with SPARQL

Ilianna Kollia[1], Birte Glimm[2], and Ian Horrocks[2]

[1] National Technical University of Athens, Greece
[2] Oxford University Computing Laboratory, UK

Abstract. W3C currently extends the SPARQL query language with so-called entailment regimes, which define how queries are evaluated using logical entailment relations. We describe a sound and complete algorithm for the OWL Direct Semantics entailment regime. Since OWL's Direct Semantics is based on Description Logics (DLs), this results in an expressive query language for DL knowledge bases. The query language differs from the commonly studied conjunctive queries in that it only has distinguished variables. Furthermore, variables can occur within complex concepts and can also bind to concept or role names. We provide a prototypical implementation and propose several novel optimization strategies. We evaluate the efficiency of the proposed optimizations and find that for ABox queries our system performs comparably to already deployed systems. For complex queries an improvement of up to three orders of magnitude can be observed.

1 Introduction

Query answering is important in the context of the Semantic Web, since it provides a mechanism via which users and applications can interact with ontologies and data. Although SPARQL [12] was standardized in 2008 by the World Wide Web Consortium for querying Semantic Web data, only the simple semantics of RDF is supported by SPARQL 1.0, which does not allow for any reasoning.

There is not yet a standardized query language for OWL knowledge bases (KBs). Several of the widely deployed systems support, however, some query language. Pellet supports SPARQL-DL [13], which is a subset of SPARQL, adapted to work with OWL's Direct Semantics. Similarly, KAON2 supports [9] SPARQL, but restricted to ABox queries. Racer Pro [3] has a proprietary query language, called nRQL [4], which allows for queries that go beyond ABox queries, e.g., one can retrieve sub- or super-concepts of a given concept. TrOWL is another system that supports ABox SPARQL queries, but the reasoning in TrOWL is approximate, i.e., an OWL DL ontology is rewritten into an ontology that uses a less expressive language before reasoning is applied [14]. Furthermore, there are systems such as QuOnto[3] or Requiem,[4] which support profiles of OWL 2, and which support conjunctive queries, e.g., written in SPARQL syntax. Of the systems that support all of OWL 2 DL, only Pellet supports non-distinguished variables as long as they are not used in cycles, which is a measure to ensure decidability.

[3] http://www.dis.uniroma1.it/~quonto/
[4] http://www.comlab.ox.ac.uk/projects/requiem/home.html

The SPARQL W3C working group is currently devising version 1.1 of SPARQL, which also includes several *entailment regimes*. These entailment regimes redefine the semantics of SPARQL queries based on standard semantic web entailment relations such as RDFS or OWL Direct Semantics entailment. This allows for using SPARQL also as a query language over OWL ontologies with query answers also including solutions that are implicit consequences of the queried ontology or knowledge base.

In this paper, we present an implementation and optimization techniques for the SPARQL OWL 2 Direct Semantics entailment regime, which we call SPARQL-OWL for brevity. SPARQL-OWL only allows for distinguished variables (for compatibility with SPARQL 1.0), but it poses significant challenges for implementations, e.g., by allowing variables that bind to concepts or roles and which can even occur within complex concepts. Our implementation supports ontologies (knowledge bases) in OWL 2 DL and is based on the HermiT reasoner.[5] Most of the devised optimization techniques are also applicable when using another OWL reasoner. In our algorithm, we extend the techniques used for conjunctive query answering to deal with arbitrary SPARQL-OWL queries and propose a range of novel optimizations in particular for SPARQL-OWL queries that go beyond SPARQL-DL.

Our prototypical system is the first to fully support SPARQL-OWL, and we have performed a preliminary evaluation in order to investigate the feasibility of our algorithm and the effectiveness of the proposed optimizations. This evaluation suggests that, in the case of standard conjunctive queries, our system performs comparably to existing ones. It also shows that a naive implementation of our algorithm behaves badly for some non-standard queries, but that the proposed optimizations can dramatically improve performance, in some cases by as much as three orders of magnitude.

An extended version of this paper is accepted at ESWC'11 [10].

2 Preliminaries

In this section we give a brief introduction to the SPARQL-OWL entailment regime and in the next section we describe an algorithm that finds answers to queries under this regime.

2.1 The Relationship between RDF, SPARQL, and OWL

SPARQL is originally an RDF query language and the WHERE clause of a SPARQL query consists of an RDF graph, where some nodes or edges are replaced by variables. There is, however, a close relationship between OWL and RDF since OWL ontologies can be represented as RDF graphs. Furthermore, OWL's RDF-Based Semantics is a direct extension of the RDF and RDFS semantics. We focus here, however, on OWL's Direct Semantics, which is based on the DL $\mathcal{SROIQ}$ [8] and which is only defined for certain well-formed RDF graphs. Well-formedness guarantees that the RDF graph can be mapped into an OWL 2 DL ontology [11], which can be seen as a $\mathcal{SROIQ}$ KB.

[5] http://www.hermit-reasoner.com

An example of a SPARQL query is

SELECT ?i FROM <ontologyIRI> WHERE { ?i rdf:type C }

where the triple in the WHERE clause is called a basic graph pattern (BGP) and is written in Turtle [1]. Since the Direct Semantics of OWL is defined in terms of OWL structural objects, such a BGP is mapped into structural objects, which can have variables in place of class, object property, data property, or individual names or literals. For example, the above BGP is mapped to ClassAssertion(C ?i) in functional-style syntax or C(?i) in DL syntax.

OWL DL is a typed language and to map RDF triples into OWL structural objects, one often has to know the type of a term. For example, in order to map the triple p rdfs:subpropertyOf p′ into an OWL structural object, we have to know whether p is an abstract or a concrete role (an object or a data property), in the former case, the mapping results in SubObjectPropertyOf(p p′), whereas in the latter case, we get SubDataPropertyOf(p p′). In DL notation, we get $\mathsf{p} \sqsubseteq \mathsf{p}'$, but p and p′ would either be abstract or concrete roles. In many cases, the typing information from the queried KB can be used to disambiguate the mapping process. For variables that map to concepts or roles, however, typing information is usually required and has to be added to the BGP. For example,

a rdf:type [rdf:type owl:Restricion ; owl:onProperty ?x ; owl:someValuesFrom ?y]

could be mapped to either (1) or (2).

$$\mathsf{ClassAsserion(ObjectSomeVauesFrom(?x\ ?y)\ a)} \tag{1}$$

$$\mathsf{ClassAsserion(DataSomeVauesFrom(?x\ ?y)\ a)} \tag{2}$$

In such a case, a triple such as ?x rdf:type owl:ObjectProperty can be added to disambiguate the mapping process. Although the SPARQL specification uses Turtle, other query syntaxes can also be defined. Pellet accepts, for example, queries where the BGP is written in Manchester Syntax [7].

For further details, we refer interested readers to the W3C specification that defines the mapping between OWL structural objects and RDF graphs [11] and to the SPARQL-OWL entailment regime[6] that defines the extension of this mapping between BGPs and OWL objects with variables.

2.2 SPARQL-OWL Queries

In the following, we directly write BGPs in DL notation extended to allow for variables in place of concept, role and individual names in axioms. For simplicity, we do not consider concrete roles (data properties) here.

Anonymous individuals in the query are treated as variables whose bindings do not appear in the query's result sequence. This is motivated by the way SPARQL handles anonymous individuals (known as blank nodes in RDF terminology). This is in contrast to conjunctive queries where they are treated as existential variables. Furthermore, anonymous individuals in the queried KB are treated as (Skolem) constants and can be returned in a query answer. For brevity, we assume here that neither the query nor the queried KB contains anonymous individuals.

[6] http://www.w3.org/TR/sparql11-entailment/

Definition 1. *Let N_C, N_R, N_I, V_C, V_R, and V_I be countable, infinite, and pairwise disjoint sets of* concept names, role names, individual names, concept variables, role variables, *and* individual variables, *respectively. We call $\mathcal{S} = (N_C, N_R, N_I, V_C, V_R, V_I)$ a* signature. *A SPARQL-OWL* query *w.r.t. $\mathcal{S}$ consists of* axiom templates, *which are $\mathcal{SROIQ}$ axioms where in place of concept names, one can use names from $N_C \cup V_C$, in place of role names, one can use names from $N_R \cup V_R$, and in place of individual names, one can use names form $N_I \cup V_I$. A $\mathcal{SROIQ}$* knowledge base *uses only terms from N_C, N_R, and N_I. The restriction of $\mathcal{S}$ to terms that occur in a knowledge base $\mathcal{K}$ (a query q) is denoted as $\mathcal{S}_\mathcal{K}$ ($\mathcal{S}_q$); we write $\mathsf{V}(\mathsf{q})$ to denote the set of all variables in q.*

Given a knowledge base $\mathcal{K}$ with $\mathcal{S}_\mathcal{K} = (N_C^\mathcal{K}, N_R^\mathcal{K}, N_I^\mathcal{K}, \emptyset, \emptyset, \emptyset)$ and a query q over $(N_C^\mathcal{K}, N_R^\mathcal{K}, N_I^\mathcal{K}, V_C, V_R, V_I)$, a solution mapping *$\mu$* for *$q$* over *$\mathcal{K}$ is a partial function $\mu\colon V_C \cup V_R \cup V_I \to N_C^\mathcal{K} \cup N_R^\mathcal{K} \cup N_I^\mathcal{K}$ such that $\mathsf{dom}(\mu) = \mathsf{V}(\mathsf{q})$, $\mu(v) \in N_C^\mathcal{K}$ for each $v \in V_C \cap \mathsf{dom}(\mu)$, $\mu(v) \in N_R^\mathcal{K}$ for each $v \in V_R \cap \mathsf{dom}(\mu)$, and $\mu(v) \in N_I^\mathcal{K}$ for each $v \in V_I \cap \mathsf{dom}(\mu)$, where $\mathsf{dom}(\mu)$ denotes the domain of μ; we write $\mu(q)$ to denote the result of replacing each variable v in q with $\mu(v)$.*

The evaluation of q over $\mathcal{K}$ yields a set of solution mappings μ with

$$\{\, \mu \mid \mathcal{K} \cup \mu(q) \text{ is a } \mathcal{SROIQ} \text{ knowledge base and } \mathcal{K} \models \mu(q)\}$$

More complex WHERE clauses, which use operators such as UNION for alternative selection criteria or OPTIONAL to query for optional bindings [12, 5], can be evaluated simply by combining solution mappings obtained by the BGP/query evaluation. Therefore, we focus here on BGP evaluation only.

In the remainder, we use $\mathcal{K}$ to denote the $\mathcal{SROIQ}$ KB obtained from a queried RDF graph, and q for the query obtained from mapping a BGP into axiom templates. We further assume that the signature of $\mathcal{K}$ is $\mathcal{S}_\mathcal{K} = (N_C^\mathcal{K}, N_R^\mathcal{K}, N_I^\mathcal{K}, \emptyset, \emptyset, \emptyset)$ and a query uses symbols from $(N_C^\mathcal{K}, N_R^\mathcal{K}, N_I^\mathcal{K}, V_C, V_R, V_I)$.

3 Evaluation of SPARQL-OWL Queries

A straightforward algorithm to realize the entailment regime simply tests, for each possible solution mapping μ, whether $\mathcal{K} \models \mu(q)$. Since only terms that are used in $\mathcal{K}$ can occur in the range of solution mappings, there are finitely many mappings to test. In the worst case, however, the number of mappings that have to be tested is still exponential in the number of variables in the query. Such an algorithm is sound and complete if the reasoner used to decide entailment is sound and complete since we check all mappings for variables that can constitute actual solution mappings.

3.1 General Query Evaluation Algorithm

Optimizations cannot easily be integrated in the above sketched algorithm since it uses the reasoner to check for the entailment of the instantiated query as a whole and, hence, does not take advantage of relations that may exist between axiom templates. For a more optimized evaluation, we evaluate the query axiom template by axiom template. Initially, our solution set contains only the identity mapping, which does not map any variable to a value. We then pick our first axiom template, extend the identity mapping to cover the variables of the chosen axiom template and use the reasoner to check which of the mappings instantiate the axiom template into an entailed axiom. We then pick the next axiom template and again extend the mappings from the previous round to cover all variables and check which of those mappings lead to an entailed axiom. Thus, axiom templates which are very selective and are only satisfied by very few solutions reduce the number of intermediate solutions. Choosing a good execution order, therefore, can significantly affect the performance.

For example, let $q = \{\mathsf{C}(?\mathsf{x}), \mathsf{r}(?\mathsf{x}\ ?\mathsf{y})\}$ with $\mathsf{r} \in N_R, \mathsf{C} \in N_C, ?\mathsf{x}, ?\mathsf{y} \in V_I$. The query belongs to the class of conjunctive queries. We assume that the queried KB contains 100 individuals, only 1 of which belongs to the concept C. This C instance has 1 r-successor, while we have overall 200 pairs of individuals related with the role r. If we first evaluate C(?x), we test 100 mappings (since ?x is an individual variable), of which only 1 mapping satisfies the axiom template. We then evaluate r(?x ?y) by extending the mapping with all 100 possible mappings for ?y. Again only 1 mapping yields a solution. For the reverse axiom template order, the first axiom template requires the test of $100 * 100$ mappings. Out of those, 200 remain to be checked for the second axiom template and we perform $10,200$ tests instead of just 200.

The importance of the execution order is well known in relational databases and cost based optimization techniques are used to find good execution orders. Ordering strategies as implemented in databases or triple stores are, however, not directly applicable in our setting. In the presence of expressive schema level axioms, we cannot rely on counting the number of occurrences of triples. We also cannot, in general, precompute all relevant inferences to base our statistics on materialized inferences. Furthermore, we should not only aim at decreasing the number of intermediate results, but also take into account the cost of checking or computing the solutions. This cost can be very significant with OWL reasoning.

For several kinds of axiom templates we can, instead of checking entailment, directly retrieve the solutions from the reasoner. For example, for C(?x), reasoners typically have a method to retrieve concept instances. Although this might internally trigger several tests, most methods of reasoners are highly optimized and avoid as many tests as possible. Furthermore, reasoners typically cache several results such as the computed concept hierarchy and retrieving sub-concepts can then be realized with a cache lookup. Thus, the actual execution cost might vary significantly. Notably, we do not have a straight correlation between the number of results for an axiom template and the actual cost of retrieving the solutions as is typically the case in triple stores or databases. This requires cost models that take into account the cost of the specific reasoning operations (depending on the state of the reasoner) as well as the number of results.

As motivated above, we distinguish between *simple* and *complex* axiom templates, where simple axiom templates are those that correspond to dedicated reasoning tasks. Complex axiom templates are, in contrast, evaluated by iterating over the compatible mappings and by checking entailment for each instantiated axiom template. An example of a complex axiom template is $(\exists \mathsf{r}.?\mathsf{x})(?\mathsf{y})$.

Algorithm 1 shows how we evaluate queries. We first explain the general outline of the algorithm and leave the details of the used submethods for the following section. We first simplify axiom templates where possible (rewrite, line 1). Next, the method

Algorithm 1 Query Evaluation Procedure

```
Input: $\mathcal{K}$: the queried knowledge base, which is a $\mathcal{SROIQ}$ knowledge base
       $q$: a $\mathcal{SROIQ}$ query
Output: a set of solutions for evaluating $q$ over $\mathcal{K}$
 1: Axt := rewrite($K_q$) {create a list Axt of simplified axiom templates from $q$}
 2: $\mathsf{Axt}^1, \ldots, \mathsf{Axt}^m$:=connectedComponents(Axt)
 3: for j=1, ..., m do
 4:     $R_j := \{\mu_0 \mid \mathsf{dom}(\mu_0) = \emptyset\}$
 5:     $\mathsf{axt}_1, \ldots, \mathsf{axt}_n := \mathsf{reorder}(\mathsf{Axt}^j)$
 6:     for $i = 1, \ldots, n$ do
 7:         $R_{new} := \emptyset$
 8:         for $\mu \in R_j$ do
 9:             if isSimple($\mathsf{axt}_i$) and $\mathsf{V}(\mathsf{axt}_i) \setminus \mathsf{dom}(\mu) \neq \emptyset$ then
10:                 $R_{new} := R_{new} \cup \{(\mu \cup \mu') \mid \mu' \in \mathsf{callReasoner}(\mu(\mathsf{axt}_i))\}$
11:             else
12:                 $B := \{\mu' \mid \mu'$ extends $\mu, \mu'$ is a solution mapping for $\mathsf{axt}_i$ and $\mathcal{K}\}$
13:                 $B := \mathsf{prune}(B, \mathsf{axt}_i, \mathcal{K})$
14:                 while $B \neq \emptyset$ do
15:                     $\mu' := \mathsf{removeNext}(B)$
16:                     if $\mathcal{K} \models \mu'(\mathsf{axt}_i)$ then $R_{new} := R_{new} \cup \{\mu'\}$
17:                     else $B := \mathsf{prune}(B, \mathsf{axt}_i, \mu')$
18:                 end while
19:             end if
20:         end for
21:         $R_j := R_{new}$
22:     end for
23: end for
24: $R := \{\mu_1 \cup \ldots \cup \mu_m \mid \mu_j \in R_j, 1 \leq j \leq m\}$
25: return $R$
```

connectedComponents (line 2) partitions the axiom templates into sets of connected components, i.e., within a component the templates share common variables, whereas between components there are no shared variables. Unconnected components unnecessarily increase the amount of intermediate results and, instead, we can simply combine the results for the components in the end (line 24). For each component, we proceed as described below: we first determine an order (method reorder in line 5). For a simple axiom template, which contains so far unbound variables, we then call a specialized reasoner method to retrieve entailed results (callReasoner in line 10). Otherwise, we check which compatible solutions yield an entailed axiom (lines 11 to 19). The method prune (lines 13 and 17) excludes mappings that cannot lead to entailed axioms.

3.2 Optimized Query Evaluation

Axiom Template Reordering We now explain how we order the axiom templates in the method reorder (line 5). Since complex axiom templates can only be evaluated with costly entailment checks, our aim is to reduce the number of bindings before we check the complex templates. The simple axiom templates are ordered by their cost,

Table 1. Axiom templates and their equivalent simpler ones, where $C_{(i)}$ are complex concepts (possibly containing variables), *a* is an individual or variable

$$C_1 \sqcap \ldots \sqcap C_n(a) \equiv \{C_i(a) \mid 1 \leq i \leq n\}$$
$$C \sqsubseteq C_1 \sqcap \ldots \sqcap C_n \equiv \{C \sqsubseteq C_i \mid 1 \leq i \leq n\}$$
$$C_1 \sqcup \ldots \sqcup C_n \sqsubseteq C \equiv \{C_i \sqsubseteq C \mid 1 \leq i \leq n\}$$

which is computed as the weighted sum of the estimated number of required consistency checks and the estimated result size. These estimates are based on statistics provided by the reasoner and this is the only part where our algorithm depends on the specific reasoner that is used. In case the reasoner cannot give estimates, one can still work with statistics computed from explicitly stated information. We do this for some simple templates, e.g., queries for domains and ranges of properties, for which the reasoner does not provide result size estimations. Since the result sizes for complex templates are difficult to estimate using either the reasoner or the explicitly stated information in $\mathcal{K}$, we order complex templates based only on the number of bindings that have to be tested. It is obvious that the reordering of axiom templates does not affect soundness and completeness of Algorithm 1.

Axiom Template Rewriting Some costly to evaluate axiom templates can be rewritten into axiom templates that can be evaluated more efficiently and yield an equivalent result. Such axiom templates are shown on the left-hand side of Table 1 and their equivalent simplified form is shown on the right-hand side. To understand the intuition behind such transformation, we consider a query with only the axiom template: $?x \sqsubseteq \exists r.?y \sqcap C$. Its evaluation requires a quadratic number of consistency checks in the number of concepts (since $?x$ and $?y$ are concept variables). The rewriting yields: $?x \sqsubseteq C$ and $?x \sqsubseteq \exists r.?y$. The first axiom template is now evaluated with a cheap cache lookup (assuming that the concept hierarchy has been precomputed). For the second one, we only have to check the usually few resulting bindings for $?x$ combined with all other concept names for $?y$. We apply the rewriting in the method rewrite in line 1 of our algorithm. Soundness and completeness is preserved since instantiated rewritten templates are semantically equivalent to the corresponding instantiated complex ones.

Concept and Role Hierarchy Exploitation The number of consistency checks required to evaluate a query can be further reduced by taking the concept and role hierarchies into account. Once the concepts and roles are classified (this can ideally be done before a system accepts queries), the hierarchies are stored in the reasoner's internal structures. We further use the hierarchies to prune the search space of solutions in the evaluation of certain axiom templates. We illustrate the intuition with an example: $\mathsf{Infection} \sqsubseteq \exists \mathsf{hasCausalLinkTo}.?x$ If C is not a solution and $B \sqsubseteq C$ holds, then B is also not a solution. Thus, when searching for solutions for $?x$, the method removeNext (line 15) chooses the next binding to test by traversing the concept hierarchy topdown. When we find a non-solution C, the subtree rooted in C of the concept hierarchy can safely be pruned, which we do in the method prune in line 17. Queries over knowledge bases with a large number of concepts and a deep concept hierarchy can, therefore, gain the maximum advantage from this optimization. We employ similar optimizations using the role hierarchies. It is obvious that we only prune mappings that cannot constitute actual solution and instance mappings, hence, soundness and completeness of Algorithm 1 is preserved.

Exploiting the Domain and Range Restrictions The implicit domains and ranges of the roles in $\mathcal{K}$ (in case the reasoner precomputes and stores them) and/or the explicit ones can be exploited to reduce the number of entailment checks that need to be performed in order to evaluate a query.

Let us assume that $\mathcal{K}$ contains $\top \sqsubseteq \forall \mathsf{takesCourse.Course}$, expressing a range restriction, and q contains $\mathsf{GraduateStudent} \sqsubseteq \exists \mathsf{takesCourse.?x}$. In case at least one solution mapping exists for ?x, the concept Course and its super-concepts can immediately be considered solution mappings for ?x. Moreover, if the reasoner precomputes the disjoint concepts, this information can be used to prune the possible concepts for ?x that are disjoint from the concept Course. This is done in the method prune (line 13), which again preserves soundness and completeness.

4 System Evaluation

Since SPARQL's entailment regimes only change the evaluation of BGPs, standard SPARQL algebra processors can be used to combine the intermediate results, e.g., in unions or joins. Furthermore, standard OWL reasoners such as HermiT, Pellet, or FaCT++ can be used to perform the required reasoning tasks.

4.1 The System Architecture

In our system, the queried KB is loaded into an OWL reasoner and the reasoner performs initial tasks such as concept classification before the system accepts queries. We use the ARQ library[7] of the Jena Semantic Web Toolkit for parsing the SPARQL queries and for the SPARQL algebra operations apart from the BGP evaluation. The BGPs are mapped to queries (as in Def. 1) and represented in a custom extension of the OWL API [6]. The query is then passed to a query optimizer, which applies the axiom template rewriting and then searches for a good query execution plan based on statistics provided by the reasoner. We use the HermiT reasoner for OWL reasoning, but only the module that generates statistics and provides cost estimations is HermiT specific.

4.2 Experimental Results

We tested our system with the Lehigh University Benchmark (LUBM) [2] and a range of custom queries that test complex axiom template evaluation over the more expressive GALEN ontology. All experiments were performed on a Windows Vista machine with a double core 2.2 GHz Intel x86 32 bit processor and Java 1.6 allowing 1GB of Java heap space. We measure the time for one-off tasks such as classification separately since such tasks are usually performed before the system accepts queries. Whether more

[7] http://jena.sourceforge.net/ARQ/

Table 2. Query answering times in milliseconds for LUBM(1,0) and in seconds for the queries of Table 3 with and without optimizations

LUBM(1, 0)	
Query	Time
1	20
2	46
3	19
4	19
5	32
6	58
7	42
8	353
9	4,475
10	23
11	19
12	28
13	16
14	45

GALEN queries from Table 3				
Query	Reordering	Hierarchy Exploitation	Rewriting	Time
1				2.1
1		x		0.1
2				780.6
2		x		4.4
3				>30 min
3		x		119.6
3			x	204.7
3		x	x	4.9
4	x		x	>30 min
4	x	x		361.9
4		x	x	>30 min
4	x	x	x	68.2
5	x			>30 min
5		x		>30 min
5	x	x		5.6

costly operations such as the realization of the ABox, which computes the types for all individuals, are done in the beginning, depends on the setting and the reasoner. Since realization is relatively quick in HermiT for LUBM (GALEN has no individuals), we also performed this task upfront. The given results are averages from executing each query three times. The ontologies and all code required to perform the experiments are available online.[8]

We first evaluate the 14 LUBM queries. These queries are simple ones and have variables only in place of individuals and literals. The LUBM ontology contains 43 concepts, 25 abstract roles, and 7 concrete roles. We tested the queries on LUBM(1,0), which contains data for one university starting from index 0, and which contains 16,283 individuals and 8,839 literals. The ontology took 3.8 s to load and 22.7 s for classification and realization. Table 2 shows the execution time for each of the queries. The reordering optimization has the biggest impact on queries 2, 7, 8, and 9. These queries require much more time or are not answered at all within the time limit of 30 min without this optimization (758.9 s, 14.7 s, >30 min, >30 min, respectively).

Conjunctive queries are supported by a range of OWL reasoners. SPARQL-OWL allows, however, the creation of very powerful queries, which are not currently supported by any other system. In the absence of suitable standard benchmarks, we created a custom set of queries as shown in Table 3. Since the complex queries are mostly based on complex schema queries, we switched from the very simple LUBM ontology to the GALEN ontology. GALEN consists of 2,748 concepts and 413 abstract roles. The ontology took 1.6 s to load and 4.8 s to classify (concepts and roles). The execution time for these queries is shown on the right-hand side of Table 2. For each query, we tested

[8] http://www.hermit-reasoner.com/2010/sparqlowl/sparqlowl.zip

Table 3. Sample complex queries for the GALEN ontology

1	$\mathsf{Infection} \sqsubseteq \exists \mathsf{hasCausalLinkTo}.?x$
2	$\mathsf{Infection} \sqsubseteq \exists ?y.?x$
3	$?x \sqsubseteq \mathsf{Infection} \sqcap \exists \mathsf{hasCausalAgent}.?y$
4	$\mathsf{NAMEDLigament} \sqsubseteq \mathsf{NAMEDInternalBodyPart} \sqcap ?x$ $?x \sqsubseteq \exists \mathsf{hasShapeAnalagousTo}?y \ \sqcap \ \exists ?z.\mathsf{linear}$
5	$?x \sqsubseteq \mathsf{NonNormalCondition}$ $?z \sqsubseteq \mathsf{ModifierAttribute}$ $\mathsf{Bacterium} \sqsubseteq \exists ?z.?w$ $?y \sqsubseteq \mathsf{StatusAttribute}$ $?w \sqsubseteq \mathsf{AbstractStatus}$ $?x \sqsubseteq \exists ?y.\mathsf{Status}$

the execution once without optimizations and once for each combination of applicable optimizations from Section 3.

As expected, an increase in the number of variables within an axiom template leads to a significant increase in the query execution time because the number of mappings that have to be checked grows exponentially in the number of variables. This can, in particular, be observed from the difference in execution time between Query 1 and 2. From Queries 1, 2, and 3 it is evident that the use of the hierarchy exploitation optimization leads to a decrease in execution time of up to two orders of magnitude and, in combination with the query rewriting optimization, we can get an improvement of up to three orders of magnitude as seen in Query 3. Query 4 can only be completed in the given time limit if at least reordering and hierarchy exploitation is enabled. Rewriting splits the first axiom template into the following two simple axiom templates, which are evaluated much more efficiently:

$$\mathsf{NAMEDLigament} \sqsubseteq \mathsf{NAMEDInternalBodyPart} \quad \text{and} \quad \mathsf{NAMEDLigament} \sqsubseteq ?x$$

After the rewriting, the reordering optimization has an even more pronounced effect since both rewritten axiom templates can be evaluated with a simple cache lookup. Without reordering, the complex axiom template could be executed before the simple ones, which leads to the inability to answer the query within the time limit of 30 min. Without a good ordering, Query 5 can also not be answered, but the additional use of the class and property hierarchy further improves the execution time by three orders of magnitude.

Although our optimizations can significantly improve the query execution time, the required time can still be quite high. In practice, it is, therefore, advisable to add as many restrictive axiom templates for query variables as possible. For example, the addition of $?y \sqsubseteq \mathsf{Shape}$ to Query 4 reduces the runtime from 68.2 s to 1.6 s.

5 Discussion

We have presented a sound and complete query answering algorithm and novel optimizations for SPARQL's OWL Direct Semantics entailment regime. Our prototypical query answering system combines existing tools such as ARQ, the OWL API, and the HermiT OWL reasoner. Apart from the query reordering optimization—which uses (reasoner dependent) statistics provided by HermiT—the system is independent of the reasoner used, and could employ any reasoner that supports the OWL API.

We evaluated the algorithm and the proposed optimizations on the LUBM benchmark and on a custom benchmark that contains queries that make use of the very expressive features of the entailment regime. We showed that the optimizations can improve query execution time by up to three orders of magnitude.

Acknowledgements This work was supported by EPSRC in the project *HermiT: Reasoning with Large Ontologies*. The work has also been supported by the EC Indicate project.

References

1. Beckett, D., Berners-Lee, T.: Turtle – Terse RDF Triple Language. W3C Team Submission (14 January 2008), available at http://www.w3.org/TeamSubmission/turtle/
2. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Semantics 3(2-3), 158–182 (2005)
3. Haarslev, V., Möller, R.: Racer system description. In: Gor, R., Leitsch, A., Nipkow, T. (eds.) Proc. 1st Int. Joint Conf. on Automated Reasoning (IJCAR'01). LNCS, vol. 2083, pp. 701–705. Springer (2001)
4. Haarslev, V., Möller, R., Wessel, M.: Querying the semantic web with Racer + nRQL. In: Proc. KI-2004 International Workshop on Applications of Description Logics (2004)
5. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
6. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Patel-Schneider, P.F., Hoekstra, R. (eds.) Proc. OWLED 2009 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org (2009)
7. Horridge, M., Patel-Schneider, P.F. (eds.): OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note (27 October 2009), available at http://www.w3.org/TR/owl2-manchester-syntax/
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
9. Hustadt, U., Motik, B., Sattler, U.: Reducing $\mathcal{SHIQ}^{-}$ description logic to disjunctive datalog programs. In: Proc. 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'04). pp. 152–162. AAAI Press (2004)
10. Kollia, I., Glimm, B., Horrocks, I.: SPARQL Query Answering over OWL Ontologies. In: Proc. 8th Extended Semantic Web Conf. (ESWC'11) (2011), to appear
11. Patel-Schneider, P.F., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-mapping-to-rdf/
12. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Recommendation (15 January 2008), available at http://www.w3.org/TR/rdf-sparql-query/
13. Sirin, E., Parsia, B.: SPARQL-DL: SPARQL query for OWL-DL. In: Golbreich, C., Kalyanpur, A., Parsia, B. (eds.) Proc. OWLED 2007 Workshop on OWL: Experiences and Directions. CEUR Workshop Proceedings, vol. 258. CEUR-WS.org (2007)
14. Thomas, E., Pan, J.Z., Ren, Y.: TrOWL: Tractable OWL 2 reasoning infrastructure. In: Proceedings of the Extended Semantic Web Conference (ESWC'10) (2010)

Module Extraction via Query Inseparability in *OWL 2 QL*

Boris Konev[1], Roman Kontchakov[2], Michel Ludwig[1], Thomas Schneider[3], Frank Wolter[1], and Michael Zakharyaschev[2]

[1] University of Liverpool, UK `{konev,michel.ludwig,wolter}@liverpool.ac.uk`
[2] Birkbeck College London, UK `{roman,michael}@dcs.bbk.ac.uk`
[3] University of Bremen, Germany `tschneider@informatik.uni-bremen.de`

Abstract. We show that deciding conjunctive query inseparability for *OWL 2 QL* ontologies is PSpace-hard and in ExpTime. We give polynomial-time (incomplete) algorithms and demonstrate by experiments that they can be used for practical module extraction.

1 Introduction

Ontology-based data access (OBDA) has recently emerged as one of the most interesting and challenging applications of description logic. The key idea is to use ontologies for enriching data with background knowledge, and thereby enable query answering over incomplete and semistructured data via a high-level conceptual interface. The W3C recognised the importance of OBDA by including in the *OWL 2* Web Ontology Language the profile *OWL 2 QL*, which was designed for OBDA with relational database systems. *OWL 2 QL* is based on a description logic that was originally introduced under the name *DL-Lite*$_\mathcal{R}$ [5, 6] and called *DL-Lite*$^\mathcal{H}_{core}$ in the more general classification [1]. It can be described as an optimal sub-language of $\mathcal{SROIQ}$, underlying *OWL 2*, which includes most of the features of conceptual models, and for which query answering can be done in AC^0 for data complexity. Thus, *DL-Lite*$^\mathcal{H}_{core}$ is becoming a major language for developing ontologies, and a target language for translation and approximation of existing ontologies formulated in more expressive DLs [11, 4]. One of the consequences of this development is that *DL-Lite*$^\mathcal{H}_{core}$ ontologies turn out to be larger and more complex than originally envisaged. As a result, reasoning support for ontology engineering tasks such as composing, re-using, comparing, and extracting ontologies—which so far has been only analysed for expressive DLs [7, 12], $\mathcal{EL}$ [10] and *DL-Lite* dialects without role inclusions [9]—is becoming increasingly important for *DL-Lite*$^\mathcal{H}_{core}$ as well.

In the context of OBDA, the basic notion underlying many ontology engineering tasks is *Σ-query inseparability*: for a signature (a set of concept and role names) Σ, two ontologies are deemed to be inseparable if they give the same answers to any conjunctive query over any data formulated in Σ. Thus, in applications using Σ-queries and data, one can safely replace any ontology by a Σ-query inseparable one. Note that the relativisation to Σ is very important here. For example, one cannot expect modules of an ontology to be query inseparable from the whole ontology for *arbitrary* queries and data sets, whereas this should be the case if we restrict the query and data language to the module's signature or a specified subset thereof. Similarly, when comparing two versions of one ontology, the subtle and potentially problematic differences are those that concern queries over their common symbols, rather than all symbols occurring in these versions. In applications where ontologies are built using imported parts, a stronger notion of inseparability is required: two ontologies are *strongly Σ-query inseparable* if they give the same answers to Σ-queries and data when imported to an arbitrary context ontology formulated in Σ.

The aim of this paper is to (*i*) investigate the computational complexity of deciding (strong) Σ-query inseparability for *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies, (*ii*) develop efficient (though incomplete) algorithms for practical inseparability checking, and (*iii*) analyse the performance of the algorithms for the challenging task of minimal module extraction.

One of our surprising discoveries is that the analysis of Σ-query inseparability for *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies requires drastically different logical tools compared with the previously considered DLs. It turns out that the new syntactic ingredient—the interaction of role inclusions and inverse roles—makes deciding (strong) query inseparability PSPACE-hard, as opposed to the known coNP and Π^p_2-completeness results for *DL-Lite* dialects without role inclusions [9]. On the other hand, the obtained EXPTIME upper bound is actually the first known decidability result for strong inseparability, which goes beyond the 'essentially' Boolean logic and might additionally indicate a way of solving the open problem of strong Σ-query inseparability for $\mathcal{EL}$ [10]. For *DL-Lite*$_{core}$ ontologies (*without* role inclusions), strong Σ-query inseparability is shown to be only NLOGSPACE-complete. We give (incomplete) polynomial-time algorithms checking (strong) Σ-inseparability and demonstrate, by a set of minimal module extraction experiments, that they are (*i*) complete for many existing *DL-Lite*$^{\mathcal{H}}_{core}$ ontologies and signatures, and (*ii*) sufficiently fast to be used in module extraction algorithms that require thousands of Σ-query inseparability checks. All omitted proofs can be found at `www.dcs.bbk.ac.uk/~roman/owl2ql-modules`.

2 Σ-Query Entailment and Inseparability

We begin by formally defining *DL-Lite*$^{\mathcal{H}}_{core}$, underlying *OWL 2 QL*, and the notions of Σ-query inseparability and entailment. The language of *DL-Lite*$^{\mathcal{H}}_{core}$ contains countably infinite sets of *individual names* a_i, *concept names* A_i, and *role names* P_i. *Roles* R and *concepts* B of this language are defined by:

$$R \quad ::= \quad P_i \quad | \quad P_i^-, \qquad\qquad B \quad ::= \quad \bot \quad | \quad \top \quad | \quad A_i \quad | \quad \exists R.$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ *TBox*, $\mathcal{T}$, is a finite set of *inclusions*

$$B_1 \sqsubseteq B_2, \qquad R_1 \sqsubseteq R_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot,$$

where B_1, B_2 are concepts and R_1, R_2 roles. An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $B(a_i)$, $R(a_i, a_j)$ and $a_i \neq a_j$, where a_i and a_j are individual names, B a concept and R a role. $\mathsf{Ind}(\mathcal{A})$ will stand for the set of individual names occurring in $\mathcal{A}$. Taken together, $\mathcal{T}$ and $\mathcal{A}$ constitute the *DL-Lite*$^{\mathcal{H}}_{core}$ *knowledge base* (KB, for short) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The sub-language of *DL-Lite*$^{\mathcal{H}}_{core}$ without role inclusions $R_1 \sqsubseteq R_2$ is denoted by *DL-Lite*$_{core}$ [6]. The semantics of *DL-Lite*$^{\mathcal{H}}_{core}$ is defined as usual in DL [2]. We only note that, in interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we do not have to comply with the UNA, that is, we can have $a_i^{\mathcal{I}} = a_j^{\mathcal{I}}$ for $i \neq j$. We write $\mathcal{I} \models \alpha$ to say that an inclusion or assertion α is true in $\mathcal{I}$. The interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. $\mathcal{K}$ is *consistent* if it has a model. A concept B is said to be $\mathcal{T}$*-consistent* if $(\mathcal{T}, \{B(a)\})$ has a model. $\mathcal{K} \models \alpha$ means that $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $\mathcal{K}$.

A *conjunctive query* (CQ) $\boldsymbol{q}(x_1, \dots, x_n)$ is a first-order formula

$$\exists y_1 \dots \exists y_m \, \varphi(x_1, \dots, x_n, y_1, \dots, y_m),$$

where φ is constructed, using only $\wedge$, from atoms of the form $B(t)$ and $R(t_1, t_2)$, with B being a concept, R a role, and t_i being an individual name or a variable from the list $x_1, \dots, x_n, y_1, \dots, y_m$. The variables in $\vec{x} = x_1, \dots, x_n$ are called *answer variables* of $\boldsymbol{q}$. We say that an n-tuple $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is an *answer* to $\boldsymbol{q}$ in an interpretation $\mathcal{I}$ if $\mathcal{I} \models \boldsymbol{q}[\vec{a}]$ (here we regard $\mathcal{I}$ to be a first-order structure); $\vec{a}$ is a *certain answer* to $\boldsymbol{q}$ over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}[\vec{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$.

To define the main notions of this paper, consider two KBs $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A})$ and $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A})$. For example, the $\mathcal{T}_i$ are different versions of some ontology, or one of them is a refinement of the other by means of new axioms. The question we are interested in is whether they give the same answers to queries formulated in a certain signature, say, in the common vocabulary of the $\mathcal{T}_i$ or in a vocabulary relevant to an application. To be precise, by a *signature*, Σ, we understand any finite set of concept and role names. A concept (inclusion, TBox, etc.) all concept and role names of which are in Σ is called a Σ*-concept* (*inclusion*, etc.). We say that $\mathcal{K}_1$ Σ*-query entails* $\mathcal{K}_2$ if, for *all* Σ-queries $\boldsymbol{q}(\vec{x})$ and all $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\mathcal{K}_2 \models \boldsymbol{q}[\vec{a}]$ implies $\mathcal{K}_1 \models \boldsymbol{q}[\vec{a}]$. In other words: any certain answer to a Σ-query given by $\mathcal{K}_2$ is also given by $\mathcal{K}_1$. As the ABox is typically not fixed or known at the ontology design stage, we may have to compare the TBoxes over *arbitrary* Σ-ABoxes rather than a fixed one, which gives our central definition:

Definition 1. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes and Σ a signature. $\mathcal{T}_1$ *Σ-query entails* $\mathcal{T}_2$ if $(\mathcal{T}_1, \mathcal{A})$ Σ-query entails $(\mathcal{T}_2, \mathcal{A})$ for any Σ-ABox $\mathcal{A}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are *Σ-query inseparable* if they Σ-query entail each other, in which case we write $\mathcal{T}_1 \equiv_\Sigma \mathcal{T}_2$.

In many applications, Σ-query inseparability is enough to ensure that $\mathcal{T}_1$ can be safely replaced by $\mathcal{T}_2$. However, if they are developed as part of a larger ontology or are meant to be imported in other ontologies, a stronger notion is required:

Definition 2. $\mathcal{T}_1$ *strongly Σ-query entails* $\mathcal{T}_2$ if $\mathcal{T}_1 \cup \mathcal{T}$ Σ-query entails $\mathcal{T}_2 \cup \mathcal{T}$, for all Σ-TBoxes $\mathcal{T}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are *strongly Σ-query inseparable* if they strongly Σ-query entail each other, in which case we write $\mathcal{T}_1 \equiv^s_\Sigma \mathcal{T}_2$.

The following example illustrates the difference between Σ-query and strong Σ-query inseparability. For further discussion and examples, consult [7, 9].

Example 3. Let $\mathcal{T}_1 = \emptyset$, $\mathcal{T}_2 = \{\top \sqsubseteq \exists R, \exists R^- \sqsubseteq B, B \sqcap A \sqsubseteq \bot\}$and $\Sigma = \{A\}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-query inseparable. However, they are not strongly Σ-query inseparable. Indeed, for the Σ-TBox $\mathcal{T} = \{\top \sqsubseteq A\}$, $\mathcal{T}_1 \cup \mathcal{T}$ is consistent, while $\mathcal{T}_2 \cup \mathcal{T}$ is inconsistent, and so $\mathcal{T}_1 \cup \mathcal{T}$ does not Σ-query entail $\mathcal{T}_2 \cup \mathcal{T}$, as witnessed by the query $\boldsymbol{q} = \bot$.

3 Σ-Query Entailment and Σ-Homomorphisms

In this section, we characterise Σ-query entailment between *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes semantically in terms of (partial) Σ-homomorphisms between certain canonical models. Then, in the next section, we use this characterisation to investigate the complexity of deciding Σ-query entailment.

The *canonical model*, $\mathcal{M}_\mathcal{K}$, of a consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ gives correct answers to all CQs. In general, $\mathcal{M}_\mathcal{K}$ is infinite; however, it can be folded up into a small *generating model* $\mathcal{G}_\mathcal{K} = (\mathcal{I}_\mathcal{K}, \leadsto_\mathcal{K})$ consisting of a finite interpretation $\mathcal{I}_\mathcal{K}$ and a *generating relation* $\leadsto_\mathcal{K}$ that defines the unfolding. Let $\sqsubseteq^*_\mathcal{T}$ be the reflexive and transitive closure of the role inclusion relation given by $\mathcal{T}$, and let $[R] = \{S \mid R \sqsubseteq^*_\mathcal{T} S \text{ and } S \sqsubseteq^*_\mathcal{T} R\}$. We write $[R] \leq_\mathcal{T} [S]$ if $R \sqsubseteq^*_\mathcal{T} S$; thus, $\leq_\mathcal{T}$ is a partial order on the set $\{[R] \mid R \text{ a role in } \mathcal{T}\}$. For each $[R]$, we introduce a *witness* $w_{[R]}$ and define a *generating relation* $\leadsto_\mathcal{K}$ on the set of these witnesses together with $\mathsf{Ind}(\mathcal{A})$ by taking:

- $a \leadsto_\mathcal{K} w_{[R]}$ if $a \in \mathsf{Ind}(\mathcal{A})$ and $[R]$ is $\leq_\mathcal{T}$-minimal such that $\mathcal{K} \models \exists R(a)$ and $\mathcal{K} \not\models R(a,b)$ for all $b \in \mathsf{Ind}(\mathcal{A})$;
- $w_{[S]} \leadsto_\mathcal{K} w_{[R]}$ if $[R]$ is $\leq_\mathcal{T}$-minimal with $\mathcal{T} \models \exists S^- \sqsubseteq \exists R$ and $[S^-] \neq [R]$.

A role R is *generating in* $\mathcal{K}$ if there are $a \in \mathsf{Ind}(\mathcal{A})$ and $R_1, \dots, R_n = R$ such that $a \leadsto_\mathcal{K} w_{[R_1]} \leadsto_\mathcal{K} \cdots \leadsto_\mathcal{K} w_{[R_n]}$. The interpretation $\mathcal{I}_\mathcal{K}$ is defined as follows:

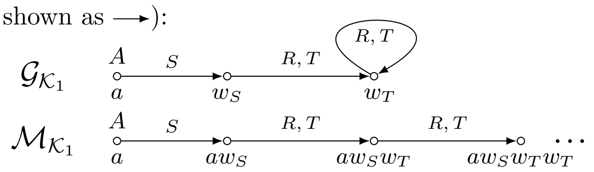
$$\begin{aligned}
\Delta^{\mathcal{I}_\mathcal{K}} &= \mathsf{Ind}(\mathcal{A}) \cup \{w_{[R]} \mid R \text{ is generating in } \mathcal{K}\},\\
a^{\mathcal{I}_\mathcal{K}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{I}_\mathcal{K}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{w_{[R]} \mid \mathcal{T} \models \exists R^- \sqsubseteq A\},\\
P^{\mathcal{I}_\mathcal{K}} &= \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \text{there is } R(a,b) \in \mathcal{A} \text{ s.t. } [R] \leq_\mathcal{T} [P]\} \cup\\
&\quad\ \{(x, w_{[R]}) \mid x \leadsto_\mathcal{K} w_{[R]} \text{ and } [R] \leq_\mathcal{T} [P]\} \cup\\
&\quad\ \{(w_{[R]}, x) \mid x \leadsto_\mathcal{K} w_{[R]} \text{ and } [R] \leq_\mathcal{T} [P^-]\}.
\end{aligned}$$

$\mathcal{G}_\mathcal{K}$ can be constructed in polynomial time in $|\mathcal{K}|$, and it is not hard to see that $\mathcal{I}_\mathcal{K} \models \mathcal{K}$. To construct the *canonical model* $\mathcal{M}_\mathcal{K}$ giving the correct answers to all CQs, we unfold the generating model $\mathcal{G}_\mathcal{K} = (\mathcal{I}_\mathcal{K}, \leadsto_\mathcal{K})$ along $\leadsto_\mathcal{K}$. A *path* in $\mathcal{G}_\mathcal{K}$ is a finite sequence $aw_{[R_1]} \cdots w_{[R_n]}$, $n \geq 0$, such that $a \in \mathsf{Ind}(\mathcal{A})$, $a \leadsto_\mathcal{K} w_{[R_1]}$ and $w_{[R_i]} \leadsto_\mathcal{K} w_{[R_{i+1}]}$, for $i < n$. Denote by $\mathsf{path}(\mathcal{G}_\mathcal{K})$ the set of all paths in $\mathcal{G}_\mathcal{K}$ and by $\mathsf{tail}(\sigma)$ the last element in $\sigma \in \mathsf{path}(\mathcal{G}_\mathcal{K})$. $\mathcal{M}_\mathcal{K}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{M}_\mathcal{K}} &= \mathsf{path}(\mathcal{G}_\mathcal{K}),\\
a^{\mathcal{M}_\mathcal{K}} &= a, \text{ for all } a \in \mathsf{Ind}(\mathcal{A}),\\
A^{\mathcal{M}_\mathcal{K}} &= \{\sigma \mid \mathsf{tail}(\sigma) \in A^{\mathcal{I}_\mathcal{K}}\},
\end{aligned}$$

$$\begin{aligned} P^{\mathcal{M}_{\mathcal{K}}} = \; & \{(a,b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid (a,b) \in P^{\mathcal{I}_{\mathcal{K}}}\} \cup \\ & \{(\sigma,\sigma \cdot w_{[R]}) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]},\ [R] \leq_{\mathcal{T}} [P]\} \cup \\ & \{(\sigma \cdot w_{[R]},\sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow_{\mathcal{K}} w_{[R]},\ [R] \leq_{\mathcal{T}} [P^-]\}. \end{aligned}$$

Example 4. For $\mathcal{T}_1 = \{A \sqsubseteq \exists S,\ \ \exists S^- \sqsubseteq \exists T,\ \ \exists T^- \sqsubseteq \exists T,\ \ T \sqsubseteq R\}$ and $\mathcal{K}_1 = (\mathcal{T}_1, \{A(a)\})$, the models $\mathcal{G}_{\mathcal{K}_1}$ and $\mathcal{M}_{\mathcal{K}_1}$ look as follows ($\rightsquigarrow_{\mathcal{K}_1}$ in $\mathcal{G}_{\mathcal{K}_1}$ is shown as $\longrightarrow$):



Theorem 5. *For all consistent DL-Lite$^{\mathcal{H}}_{core}$ KBs $\mathcal{K} = (\mathcal{T},\mathcal{A})$, CQs $\boldsymbol{q}(\vec{x})$ and $\vec{a} \subseteq \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{K} \models \boldsymbol{q}[\vec{a}]$ iff $\mathcal{M}_{\mathcal{K}} \models \boldsymbol{q}[\vec{a}]$.*

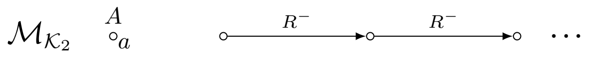
Thus, to decide Σ-query entailment between KBs $\mathcal{K}_1$ and $\mathcal{K}_2$, it suffices to check whether $\mathcal{M}_{\mathcal{K}_2} \models \boldsymbol{q}[\vec{a}]$ implies $\mathcal{M}_{\mathcal{K}_1} \models \boldsymbol{q}[\vec{a}]$ for all Σ-queries $\boldsymbol{q}(\vec{x})$ and tuples $\vec{a}$. This relationship between $\mathcal{M}_{\mathcal{K}_2}$ and $\mathcal{M}_{\mathcal{K}_1}$ can be characterised semantically in terms of finite Σ-homomorphisms. For an interpretation $\mathcal{I}$ and a signature Σ, the *Σ-types* $\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x)$ and $\boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y)$, for $x,y \in \Delta^{\mathcal{I}}$, are given by:

$$\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x) = \{\Sigma\text{-concept } B \mid x \in B^{\mathcal{I}}\}, \quad \boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y) = \{\Sigma\text{-role } R \mid (x,y) \in R^{\mathcal{I}}\}.$$

A *Σ-homomorphism* from an $\mathcal{I}$ to $\mathcal{I}'$ is a function $h\colon \Delta^{\mathcal{I}} \to \Delta^{\mathcal{I}'}$ such that $h(a^{\mathcal{I}}) = a^{\mathcal{I}'}$, for all individual names a interpreted in $\mathcal{I}$, $\boldsymbol{t}^{\mathcal{I}}_{\Sigma}(x) \subseteq \boldsymbol{t}^{\mathcal{I}'}_{\Sigma}(h(x))$ and $\boldsymbol{r}^{\mathcal{I}}_{\Sigma}(x,y) \subseteq \boldsymbol{r}^{\mathcal{I}'}_{\Sigma}(h(x),h(y))$, for all $x,y \in \Delta^{\mathcal{I}}$.

It is well-known that answers to conjunctive Σ-queries are preserved under Σ-homomorphisms. Thus, if there is a Σ-homomorphism from $\mathcal{M}_{\mathcal{K}_2}$ to $\mathcal{M}_{\mathcal{K}_1}$, then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$. However, the converse does not hold in general.

Example 6. Take $\mathcal{T}_1$ from Example 4, and let $\mathcal{T}_2$ result from replacing R in $\mathcal{T}_1$ with R^-. Let $\Sigma = \{A, R\}$ and $\mathcal{K}_i = (\mathcal{T}_i, \{A(a)\})$. Then the Σ-reduct of $\mathcal{M}_{\mathcal{K}_1}$ does not contain a Σ-homomorphic image of the Σ-*reduct* of $\mathcal{M}_{\mathcal{K}_2}$, depicted below. On the other hand, it is easily seen that $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-query inseparable.



Note that the Σ-reduct of $\mathcal{M}_{\mathcal{K}_2}$ contains points that are not reachable from the ABox by Σ-roles. In fact, using König's Lemma, one can show that if every point in $\mathcal{M}_{\mathcal{K}_2}$ is reachable from the ABox by a path of Σ-roles, then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff there exists a Σ-homomorphism from $\mathcal{M}_{\mathcal{K}_2}$ to $\mathcal{M}_{\mathcal{K}_1}$.

We say that $\mathcal{I}$ is *finitely Σ-homomorphically embeddable into $\mathcal{I}'$* if, for every finite sub-interpretation $\mathcal{I}_1$ of $\mathcal{I}$, there exists a Σ-homomorphism from $\mathcal{I}_1$ to $\mathcal{I}'$.

Theorem 7. *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be consistent DL-Lite$^{\mathcal{H}}_{core}$ KBs. Then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{M}_{\mathcal{K}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_{\mathcal{K}_1}$.*

Theorem 7 does not yet give a satisfactory semantic characterisation of Σ-query entailment between TBoxes, as one still has to consider infinitely many Σ-ABoxes. However, using the fact that inclusions in *DL-Lite*$^{\mathcal{H}}_{core}$, different from disjointness axioms, involve only *one* concept or role in the left-hand side and making sure that the TBoxes entail the same Σ-inclusions, one can show that it is enough to consider *singleton* Σ-ABoxes of the form $\{B(a)\}$. Denote the models $\mathcal{G}_{(\mathcal{T},\{B(a)\})}$ and $\mathcal{M}_{(\mathcal{T},\{B(a)\})}$ by $\mathcal{G}^B_{\mathcal{T}}$ and $\mathcal{M}^B_{\mathcal{T}}$, respectively. We thus obtain the following characterisation of Σ-entailment between *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes:

Theorem 8. *$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ iff*

(p) *$\mathcal{T}_2 \models \alpha$ implies $\mathcal{T}_1 \models \alpha$, for all Σ-inclusions α;*
(h) *$\mathcal{M}^B_{\mathcal{T}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}^B_{\mathcal{T}_1}$, for all $\mathcal{T}_1$-consistent Σ-concepts B.*

By applying condition **(p)** to $B \sqsubseteq \bot$, we obtain that every $\mathcal{T}_1$-consistent Σ-concept B is also $\mathcal{T}_2$-consistent.

4 Complexity of Σ-Query Entailment

We use Theorem 8 to show that deciding Σ-query entailment for *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes is PSpace-hard and in ExpTime. Recall that subsumption in *DL-Lite*$^{\mathcal{H}}_{core}$ is NLogSpace-complete [6, 1]; so condition **(p)** of Theorem 8 can be checked in polynomial time. And, since there are at most $2\cdot|\Sigma|$ singleton Σ-ABoxes, we can concentrate on the complexity of checking finite Σ-homomorphic embeddability of canonical models for singleton ABoxes.

We begin by considering *DL-Lite*$_{core}$, where the existence of Σ-homomorphisms between canonical models can be expressed in terms of the types of their points; cf. [9]. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be *DL-Lite*$_{core}$ TBoxes and Σ a signature.

Theorem 9. *$\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$ iff* **(p)** *holds and, for every $\mathcal{T}_1$-consistent Σ-concept B and every $x \in \Delta^{\mathcal{I}^B_{\mathcal{T}_2}}$, there is $x' \in \Delta^{\mathcal{I}^B_{\mathcal{T}_1}}$ with $\boldsymbol{t}^{\mathcal{I}^B_{\mathcal{T}_2}}_{\Sigma}(x) \subseteq \boldsymbol{t}^{\mathcal{I}^B_{\mathcal{T}_1}}_{\Sigma}(x')$.*

The criterion of Theorem 9 can be checked in polynomial time, in NLogSpace, to be more precise. Thus:

Theorem 10. *Checking Σ-query entailment for TBoxes in DL-Lite$_{core}$ is complete for* NLogSpace.

However, if role inclusions become available, the picture changes dramatically: not only do we have to compare the Σ-types of points in the canonical models, but also the Σ-*paths* to these points. To illustrate, consider the generating models $\mathcal{G}_1$, $\mathcal{G}_2$ in Fig. 1, where the arrows represent the generating relations, and the concept names A, X^0_i, X^1_i and the role names R and T_j are all symbols in Σ. The model $\mathcal{G}_2$ contains 4 R-paths from a to w, which are further extended by the infinite T_j-paths. The paths π from a to w can be homomorphically mapped to distinct R-paths $h(\pi)$ in $\mathcal{G}_1$ starting from a. But the extension of such a π

with the infinite T_j-chain can only be mapped first to a *suffix of* $h(\pi)$ (backward, along T_j^-)—because we have to map paths in the unfolding $\mathcal{M}_2$ of $\mathcal{G}_2$ to paths in $\mathcal{M}_1$—and then to a T_j-loop in $\mathcal{G}_1$. But to check whether this can be done, we may have to 'remember' the whole path π.

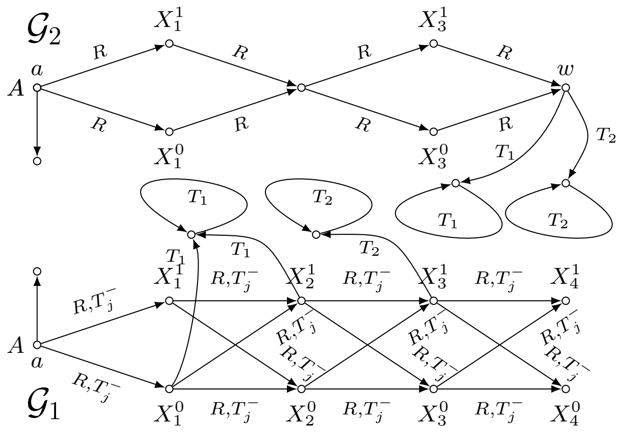


Fig. 1. Σ-reducts of generating models $\mathcal{G}_2$ and $\mathcal{G}_1$.

To see that $\mathcal{G}_1$ and $\mathcal{G}_2$ can be given by *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes, fix a QBF $\mathsf{Q}_1X_1\dots\mathsf{Q}_nX_n\bigwedge_{j=1}^m C_j$, where $\mathsf{Q}_i \in\{\forall\ ,\exists\}$ and $C_1,\dots,C_m$ are clauses over the variables $X_1,\dots,X_n$. Let $\Sigma=\{A,X_i^0,X_i^1,R,T_j \mid i\leq n,\ j\leq m\}$, $\mathcal{T}_1$ contain the inclusions

$$\begin{array}{rcc}
A \sqsubseteq \exists S_0^-, & \exists S_{i-1}^- \sqsubseteq \exists Q_i^k, & \\
\exists (Q_i^k)^- \sqsubseteq X_i^k, & Q_i^k \sqsubseteq S_i, & S_i \sqsubseteq R,\\
X_i^k \sqsubseteq \exists R_j & \multicolumn{2}{l}{\text{if } k=0,\neg X_i\in C_j \text{ or } k=1, X_i\in C_j,}\\
\exists R_j^- \sqsubseteq \exists R_j, & R_j \sqsubseteq T_j, & S_i \sqsubseteq T_j^-,
\end{array}$$

and let $\mathcal{T}_2$ contain the inclusions

$$\begin{array}{rcc}
A \sqsubseteq \exists S_0^-, & \exists S_{i-1}^- \sqsubseteq \begin{cases}\exists Q_i^k, & \text{if } \mathsf{Q}_i=\forall,\\ \exists S_i, & \text{if } \mathsf{Q}_i=\exists,\end{cases} & \\
\exists (Q_i^k)^- \sqsubseteq X_i^k, & Q_i^k \sqsubseteq S_i, & S_i \sqsubseteq R,\\
\exists S_n^- \sqsubseteq \exists P_j, & \exists P_j^- \sqsubseteq \exists P_j, & P_j \sqsubseteq T_j,
\end{array}$$

for all $i\leq n$, $j\leq m$, $k=1,2$. The generating models $\mathcal{G}^A_{\mathcal{T}_1}$ and $\mathcal{G}^A_{\mathcal{T}_2}$, restricted to Σ, look like $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 1, respectively. Moreover, one can show that $\mathcal{M}^A_{\mathcal{T}_2}$ is (finitely) Σ-homomorphically embeddable into $\mathcal{M}^A_{\mathcal{T}_1}$ iff the QBF above is satisfiable. As satisfiability of QBFs is PSPACE-complete, we obtain:

Theorem 11. *Σ-query entailment for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes is* PSPACE*-hard.*

On the other hand, the problem whether $\mathcal{M}_{\mathcal{K}_2}$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_{\mathcal{K}_1}$ can be reduced to the emptiness problem for alternating two-way automata, which belongs to EXPTIME [13]. In a way similar to [13, 8], where these automata were employed to prove EXPTIME-decidability of the modal μ-calculus with converse and the guarded fixed point logic of finite width, one can use their ability to 'remember' paths (in the sense illustrated in the example above) to obtain the EXPTIME upper bound:

Theorem 12. *Σ-query entailment for DL-Lite$^{\mathcal{H}}_{core}$ TBoxes is in* EXPTIME.

The precise complexity of Σ-query entailment for *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes is still unknown. Recall that deciding Σ-query entailment for *DL-Lite*$^{\mathcal{N}}_{horn}$ is CONP-complete [9]. Compared to *DL-Lite*$^{\mathcal{H}}_{core}$, *DL-Lite*$^{\mathcal{N}}_{horn}$ allows (unqualified) number restrictions and conjunctions in the left-hand side of concept inclusions, but does not have role inclusions: *DL-Lite*$^{\mathcal{N}}_{horn} \cap$ *DL-Lite*$^{\mathcal{H}}_{core}$ = *DL-Lite*$_{core}$. CQ answering is in AC^0 for data complexity in all three languages under the UNA. However, the computational properties of these logics become different as far as Σ-query entailment is concerned: NLOGSPACE-complete for *DL-Lite*$_{core}$, CONP-complete for *DL-Lite*$^{\mathcal{N}}_{horn}$, and between PSPACE and EXPTIME for *DL-Lite*$^{\mathcal{H}}_{core}$. It may be of interest to note that Σ-query entailment for *DL-Lite*$^{\mathcal{N}}_{bool}$, allowing full Booleans as concept constructs, is Π^p_2-complete.

Let us consider strong Σ-query entailment. It is easy to construct an exponential-time algorithm checking strong Σ-query entailment between *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$: enumerate all Σ-TBoxes $\mathcal{T}$ and check whether $\mathcal{T}_1 \cup \mathcal{T}$ Σ-query entails $\mathcal{T}_2 \cup \mathcal{T}$. As there are quadratically many Σ-inclusions, this algorithm calls the Σ-query entailment checker $\leq 2^{|\Sigma|^2}$ times. We now show that one can do much better than that. First, it turns out that instead of expensive Σ-query entailment checks for the TBoxes $\mathcal{T}_i \cup \mathcal{T}$, it is enough to check consistency (in polynomial time). More precisely, suppose $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. One can show then that $\mathcal{T}_1$ does not strongly Σ-query entail $\mathcal{T}_2$ iff there exist a Σ-TBox $\mathcal{T}$ and a Σ-concept B such that $(\mathcal{T}_1 \cup \mathcal{T}, \{B(a)\})$ is consistent but $(\mathcal{T}_2 \cup \mathcal{T}, \{B(a)\})$ is not (cf. Example 3). Moreover, checking consistency for all Σ-TBoxes $\mathcal{T}$ can further be reduced—using the primitive form of *DL-Lite*$^{\mathcal{H}}_{core}$ axioms—to checking consistency for all *singleton* Σ-TBoxes $\mathcal{T}$. Thus, we obtain the following:

Theorem 13. *Suppose that $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. Then $\mathcal{T}_1$ does not strongly Σ-query entail $\mathcal{T}_2$ iff there is a Σ-concept B and a Σ-TBox $\mathcal{T}$ with a single inclusion of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$ such that $(\mathcal{T}_1 \cup \mathcal{T}, \{B(a)\})$ is consistent but $(\mathcal{T}_2 \cup \mathcal{T}, \{B(a)\})$ is inconsistent.*

So, if we already know that $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$, then checking whether this entailment is actually *strong* can be done in polynomial time (and NLOGSPACE).

5 Incomplete Algorithm for Σ-Query Entailment

The interplay between role inclusions and inverse roles, required in the proof of PSPACE-hardness, appears to be too artificial compared to how roles are used in 'real-world' ontologies. Thus, in conceptual modelling, the number of roles is comparable with the number of concepts, but the number of role inclusions is much smaller. For this reason, instead of a complete (exponential) Σ-query entailment checker, we have implemented a polynomial-time correct but incomplete algorithm, which is based on testing simulations between transition systems.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes, Σ a signature, B a Σ-concept. Denote $\mathcal{K}_i = (\mathcal{T}_i, \{B(a)\})$ and $\mathcal{I}_i = \mathcal{I}_{\mathcal{K}_i}$, $i = 1, 2$. A relation $\rho \subseteq \Delta^{\mathcal{I}_2} \times \Delta^{\mathcal{I}_1}$ is called a *Σ-simulation of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$* if the following conditions hold:

(s1) the domain of ρ is $\Delta^{\mathcal{I}_2}$ and $(a^{\mathcal{I}_2}, a^{\mathcal{I}_1}) \in \rho$;
(s2) $\boldsymbol{t}^{\mathcal{I}_2}_{\Sigma}(x) \subseteq \boldsymbol{t}^{\mathcal{I}_1}_{\Sigma}(x')$, for all $(x, x') \in \rho$;
(s3) if $x \rightsquigarrow_{\mathcal{K}_2} w_{[R]}$ and $(x, x') \in \rho$, then there is $y' \in \Delta^{\mathcal{I}_1}$ such that $(w_{[R]}, y') \in \rho$ and $S \in \boldsymbol{r}^{\mathcal{I}_1}_{\Sigma}(x', y')$ for every Σ-role S with $[R] \leq_{\mathcal{T}_2} [S]$.

We call ρ a *forward Σ-simulation* if it satisfies **(s1)**, **(s2)** and the condition **(s3′)**, which strengthens **(s3)** with the extra requirement: $y' = w_{[T]}$, for some role T, with $x' \rightsquigarrow_{\mathcal{K}_1} w_{[T]}$ and $[T] \leq_{\mathcal{T}_1} [S]$ for every Σ-role S with $[R] \leq_{\mathcal{T}_2} [S]$.

Example 14. In Example 6, there is a Σ-simulation of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$, but no forward Σ-simulation. The same applies to $\mathcal{G}_2$ and $\mathcal{G}_1$ in the proof of the PSPACE bound.

In contrast to finite Σ-homomorphic embeddability of $\mathcal{M}_{\mathcal{K}_2}$ in $\mathcal{M}_{\mathcal{K}_1}$, the problem of checking the existence of (forward) Σ-simulations of $\mathcal{G}_{\mathcal{K}_2}$ in $\mathcal{G}_{\mathcal{K}_1}$ is tractable and well understood from the literature on program verification [3]. Consider now the following conditions, which can be checked in polynomial time:

(y) condition **(p)** holds and there is a *forward* Σ-simulation of $\mathcal{G}^{B}_{\mathcal{T}_2}$ in $\mathcal{G}^{B}_{\mathcal{T}_1}$, for every $\mathcal{T}_1$-consistent Σ-concept B;
(n) condition **(p)** does not hold or there is no Σ-simulation of $\mathcal{G}^{B}_{\mathcal{T}_2}$ in $\mathcal{G}^{B}_{\mathcal{T}_1}$, for any $\mathcal{T}_1$-consistent Σ-concept B.

Theorem 15. *Let $\mathcal{T}_1, \mathcal{T}_2$ be DL-Lite$^{\mathcal{H}}_{core}$ TBoxes and Σ a signature. If* **(y)** *holds, then $\mathcal{T}_1$ Σ-query entails $\mathcal{T}_2$. If* **(n)** *holds, then $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$.*

Thus, an algorithm checking conditions (**y**) and (**n**) can be used as a correct but incomplete Σ-query entailment checker. It cannot be complete since neither (**y**) nor (**n**) holds in Example 14. On the other hand, condition **(n)** proves to be a criterion of Σ-query entailment in two important cases:

Theorem 16. *Let* (*a*) *$\mathcal{T}_1$, $\mathcal{T}_2$ be DL-Lite$_{core}$ TBoxes, or* (*b*) *$\mathcal{T}_1 = \emptyset$ and $\mathcal{T}_2$ a DL-Lite$^{\mathcal{H}}_{core}$ TBox. Then condition* **(n)** *holds iff $\mathcal{T}_1$ does not Σ-query entail $\mathcal{T}_2$.*

6 Experiments

Checking (strong) Σ-query entailment has multiple applications in ontology versioning, re-use, and extraction. We have used the algorithms, suggested by Theorems 15 and 13, for *minimal module extraction* to see how efficient they are in practice and whether the incompleteness of the **(y)**–**(n)** conditions is problematic. Extracting minimal modules from medium-sized real-world ontologies requires thousands of calls of the (strong) Σ-query entailment checker, and thus provides a tough test for our approach.

For a TBox $\mathcal{T}$ and a signature Σ, a subset $\mathcal{M} \subseteq \mathcal{T}$ is

– a *Σ-query module* of $\mathcal{T}$ if $\mathcal{M} \equiv_\Sigma \mathcal{T}$;
– a *strong Σ-query module* of $\mathcal{T}$ if $\mathcal{M} \equiv^s_\Sigma \mathcal{T}$;
– a *depleting Σ-query module* of $\mathcal{T}$ if $\emptyset \equiv^s_{\Sigma \cup \mathsf{sig}(\mathcal{M})} \mathcal{T} \setminus \mathcal{M}$, where $\mathsf{sig}(\mathcal{M})$ is the signature of $\mathcal{M}$.

We are concerned with computing a *minimal* (w.r.t. $\subseteq$) Σ-query (MQM), a *minimal* strong Σ-query (MSQM), and the (uniquely determined) *minimal* depleting Σ-query (MDQM) module of $\mathcal{T}$. The general extraction algorithms, which call Σ-query entailment checkers, are taken from [9]. For MQMs and MSQMs, the number of calls to the checker coincides with the number of inclusions in $\mathcal{T}$. For MDQMs (where one of the TBoxes given to the checker is empty, and so the checker is complete, by Theorem 16), the number of checker calls is quadratic in the number of inclusions in $\mathcal{T}$.

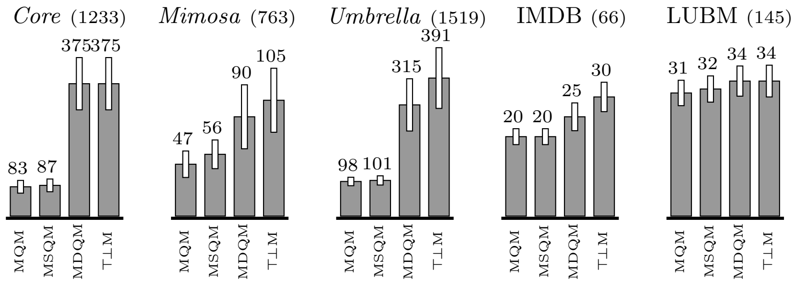
We extracted modules from *OWL 2 QL* approximations of 3 commercial software applications called *Core*, *Umbrella* and *Mimosa* (the original ontologies use a few axioms that are not expressible *OWL 2 QL*). *Mimosa* is a specialisation of the MIMOSA OSA-EAI specification[4] for container shipping. *Core* is based on a supply-chain management system used by the bookstore chain Ottakar's (now merged with Waterstone's), and *Umbrella* on a research data validation and processing system used by the Intensive Care National Audit and Research Centre.[5] The original *Core* and *Umbrella* were used for the experiments in [9].

ontology	*Mimosa*	*Core*	*Umbrella*	IMDB	LUBM
concept inclusions	710	1214	1506	45	136
role inclusions	53	19	13	21	9
concept names	106	82	79	14	43
role names	145	76	64	30	31

For comparison, we extracted modules from *OWL 2 QL* approximations of the well-known IMDB and LUBM ontologies. For each of these ontologies, we randomly generated 20 signatures Σ of 5 concept and 5 roles names. We extracted Σ-MQMs, MSQMs, MDQMs as well as the $\top\bot$-module [7] from the whole *Mimosa*, IMBD and LUBM ontologies. For the larger *Umbrella* and *Core* ontologies, we first computed the $\top\bot$-modules, and then employed them to further extract MQMs, MSQMs, MDQMs, which are all contained in the $\top\bot$-modules. The average size of the resulting modules and its standard deviation is shown below. Details of the experiments and ontologies are available at `www.dcs.bbk.ac.uk/~roman/owl2ql-modules`. Here we briefly comment on efficiency and incompleteness. Checking Σ-query inseparability turned out to be very fast: a single call of the checker never took more than 1s for our ontologies. For strong Σ-query inseparability, the maximal time was less than 1 min. For

[4] htpp://www.mimosa.org/?q=resources/specs/osa-eai-v321
[5] http://www.icnarc.org

comparisons with the empty TBox, the maximal time for strong Σ-query inseparability tests was less than 10s. In the hardest case, *Mimosa*, the average total extraction times were 2.5 mins for MQMs, 140 mins for MSQMs, and 317 mins for MDQMs. Finally, only in 9 out of about 75,000 calls, the Σ-query entailment checker was not able to give a certain answer due to incompleteness of the **(y)**–**(n)** condition, in which case the inclusions in question were added to the module.



References

[1] Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. Journal of Artificial Intelligence Research 36, 1–69 (2009)

[2] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)

[3] Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press (2007)

[4] Botoeva, E., Calvanese, D., Rodriguez-Muro, M.: Expressive approximations in *DL-Lite* ontologies. In: Proc. of AIMSA. pp. 21–31. Springer (2010).

[5] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR. pp. 260–270 (2006)

[6] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)

[7] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research 31, 273–318 (2008)

[8] Grädel, E., Walukiewicz, I.: Guarded fixed point logic. In: Proc. of LICS. pp. 45–54 (1999)

[9] Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to *DL-Lite*. Artificial Intelligence 174, 1093–1141 (2010)

[10] Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. Journal of Symbolic Computation 45(2):194–228 (2010)

[11] Pan, J.Z., Thomas, E.: Approximating OWL-DL Ontologies. In: Proc. of AAAI. pp. 1434–1439 (2007)

[12] Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies, LNCS, vol. 5445. Springer (2009)

[13] Vardi, M.Y.: Reasoning about the past with two-way automata. In: Proc. of ICALP. LNCS, vol. 1443, pp. 628–641. Springer (1998)

Nominal Schemas for Integrating Rules and Description Logics

Markus Krötzsch[1], Frederick Maier[2], Adila A. Krisnadhi[2], and Pascal Hitzler[2]

[1] University of Oxford
`markus.kroetzsch@comlab.ox.ac.uk`
[2] Kno.e.sis Center, Wright State University
`{fred,adila,pascal}@knoesis.org`

Abstract. We propose an extension of $\mathcal{SROIQ}$ with *nominal schemas* which can be used like "variable nominal concepts" within axioms. This feature allows us to express arbitrary *DL-safe rules* in description logic syntax. We show that adding nominal schemas to $\mathcal{SROIQ}$ does not increase its worst-case reasoning complexity, and we identify a family of tractable DLs $\mathcal{SROELV}_n$ that allow for restricted use of nominal schemas.

1 Introduction

A significant body of work has developed investigating the integration of description logics (DLs) and rule languages (typically Datalog). Conceptually, one can distinguish two approaches. On the one hand, description logics have been extended with additional "description-logic-style" expressive features which make it possible to express certain types of rules. For instance, $\mathcal{SROIQ}$ role inclusion axioms (RIAs) can be viewed as a type of rule. By combining RIAs with local reflexivity (Self) and the universal role U, many rules with a *tree-shaped body* can be expressed indirectly [10]. The restriction to tree-shaped rules ensures decidability, but it also excludes many rules. An example is the following rule that defines a concept C of children whose parents are married:

$$\text{hasParent}(x,y) \land \text{hasParent}(x,z) \land \text{married}(y,z) \to C(x). \qquad (1)$$

On the other hand, there are approaches of a *hybrid* nature, in the sense that both DL axioms and rules are syntactically allowed, and a combined formal semantics defines how the hybrid language is to be understood. Unfortunately, such a combination often leads to undecidability. This is the case for the *Semantic Web Rule Language* SWRL [5,6], which is the most straightforward rule extension of OWL, and for the combination of OWL DL ontologies and the *Rule Interchange Format* RIF (even when restricted to RIF Core) [1,2]. A prominently discussed idea for retaining decidability is to restrict the applicability of rules to *named individuals*, i.e., to logical constants that are explicitly mentioned in the ontology. Rules that are understood in this sense are called *DL-safe*, and the combination of OWL DL and DL-safe rules is indeed decidable [5,14].

A generalization of DL-safe rules, based on *DL-safe variables*, has been introduced [11] as part of the definition of the tractable rule language ELP. Rather than restricting *all* variables in a (DL-safe) rule to binding only to known individuals, DL-safe variables allow the ontology engineer to explicitly specify the variables to be treated this way. This approach was subsequently generalized to obtain *DL+safe Rules* as a class of expressive rule languages for which reasoning is still decidable [9].

In this paper, we expand on the above idea and improve it in several ways. The key technical innovation is the introduction of *nominal schemas* as new elements of DL syntax. While the semantic intuition behind nominal schemas is the same as that behind DL-safe variables, the difference lies in the fact that DL-safe variables are tied to rule languages, while nominal schemas integrate seamlessly with DL syntax. As a consequence, the language which we propose encompasses DL-safe variable SWRL *while staying within the DL/OWL language paradigm.* It thus achieves within the DL framework what has hitherto only been achieved by hybrid approaches.

To give an initial example, consider again the rule (1) extended by the axioms

$$\text{hasParent}(\text{mary}, \text{john}) \tag{2}$$

$$(\exists\text{hasParent}.\exists\text{married}.\{\text{john}\})(\text{mary}) \tag{3}$$

Axiom (2) asserts that John is a parent of Mary, while axiom (3) states that Mary belongs to the class of individuals with *some* (unnamed) parent who is married to John. Using a first-order logic semantics as in SWRL, rule (1) would thus entail that Mary belongs to the class C. Interpreting rule (1) as DL-safe, however, does not allow this conclusion, since John's spouse is not named by any constant in the ontology. To retain the conclusion, one can weaken this restriction to require only z to be DL-safe, while x and y can still take arbitrary values. This is possible in the rule-based approach of DL+safe Rules, but cannot be captured in an axiom of existing description logics.

In contrast, using nominal schemas, rule (1) can be expressed as

$$\exists\text{hasParent}.\{z\} \sqcap \exists\text{hasParent}.\exists\text{married}.\{z\} \sqsubseteq C. \tag{4}$$

The desired conclusion again follows. The expression $\{z\}$ is a nominal schema, which is to be read as a *variable nominal* that can only represent nominals (i.e., z binds to known individuals), where the binding is the same for all occurrences of the nominal schema in an axiom.

The main contributions of this paper are as follows:

1. We introduce nominal schemas as a new general constructor for description logics, denoted by the letter $\mathcal{V}$ in the DL nomenclature, and define the expressive DL $\mathcal{SROIQV}$ as an extension of $\mathcal{SROIQ}$.
2. We establish the worst-case complexity of reasoning in $\mathcal{SROIQV}$ to be N2ExpTime-complete, and thus not harder than for $\mathcal{SROIQ}$.

3. We define $\mathcal{SROELV}_n$ ($n \geq 0$) as a new family of DLs with nominal schemas for which reasoning is possible in polynomial time.

The expressivity of nominal schemas is also witnessed by the fact that it allows DLs to incorporate arbitrary DL-safe rules, given that concept intersections, existential role restrictions, and the universal (top) role are available. Since such rules preclude polytime reasoning, our tractable DLs $\mathcal{SROELV}_n$ employ restrictions on the number of certain occurrences of nominal schemas in each axiom.

The close relationship to nominals suggests simple ways of introducing nominal schemas into concrete syntactic forms of OWL 2, e.g. by using the existing syntax for nominal classes with special individual names that represent variables (using some suitable naming convention). This opens a path for introducing this feature into practical applications. While the above worst-case complexity result for $\mathcal{SROIQV}$ may seem encouraging, we believe that the tractable ontology languages $\mathcal{SROELV}_n$ are the most promising candidates for implementations.

This paper is a condensed presentation of the main results of [13] where we develop all results for the slightly more general case of DLs with Boolean role constructors and concept products [18]. Moreover, [13] also explains how DL-safe rules (and hence DLs with nominal schemas) can be used to express OWL RL ontologies, and provides an extended discussion of related approaches which include description graphs, existential rules and tuple-generating dependencies (TGDs) in Datalog, and DL Rules.

In this paper, we introduce the syntax and semantics of nominal schemas for $\mathcal{SROIQV}$, and establish the worst-case complexity of reasoning in Section 2. The DLs $\mathcal{SROELV}_n$ are introduced in Section 3, and their tractability is established in Section 4. In Section 5 we show how DL-safe rules can be expressed with nominal schemas, and Section 6 concludes.

2 Nominal Schemas in $\mathcal{SROIQ}$

We start by introducing *nominal schemas* as an extension of existing description logics. We first generally introduce the feature for $\mathcal{SROIQ}$ to obtain the very expressive DL $\mathcal{SROIQV}$.

A signature of $\mathcal{SROIQV}$ is a tuple $\Sigma = \langle N_I, N_C, N_R, N_V \rangle$ of mutually disjoint sets of *individual names*, *concept names*, *role names*, and *variables*. Variables can be used like individuals in nominal expressions, and concept expressions of $\mathcal{SROIQV}$ are thus defined as follows:

$$\mathbf{C} ::= \top \mid \bot \mid N_C \mid \{N_I\} \mid \{N_V\} \mid \neg\mathbf{C} \mid \mathbf{C} \sqcap \mathbf{C} \mid \mathbf{C} \sqcup \mathbf{C} \mid \\ \exists\mathbf{R}.\mathbf{C} \mid \forall\mathbf{R}.\mathbf{C} \mid \exists\mathbf{S}.\mathsf{Self} \mid \leqslant k\,\mathbf{S}.\mathbf{C} \mid \geqslant k\,\mathbf{S}.\mathbf{C}$$

where k is a natural number, and $\mathbf{R}$ ($\mathbf{S}$) is a (simple) $\mathcal{SROIQ}$ role as usual. We use U to denote the universal role. The common axiom types are defined as usual, but with the extended set of concept expressions. Herein, we restrict our attention to $\mathcal{SROIQV}$ knowledge bases with regular RBoxes, which are defined as in $\mathcal{SROIQ}$.

Axiom (4) above is an example of a $\mathcal{SROIQV}$ TBox axiom, where $\{z\}$ is a nominal schema. Intuitively, each nominal schema appearing in an axiom is universally quantified, but ranges only over elements that are referred to by an individual name. We note that it would also be straightforward to introduce nominal schemas into the normative RDF syntax for OWL 2 [16]. One way to do this would be to provide URIs for variables in the OWL namespace, used instead of individuals in `owl:oneOf` statements (which are used for the RDF syntax for nominals in OWL 2). Attaching the semantics of nominal schemas to "reserved" variable URIs would allow the reuse of existing tools for representation, manipulation, parsing, and serialization.

The semantics of $\mathcal{SROIQV}$ is defined by interpreting variables as placeholders for *named* individuals, i.e. elements of the interpretation domain that are represented by individual names in N_I. This can be accomplished by using a suitably restricted form of variable assignment. Equivalently, one can eliminate nominal schemas by replacing them with the (finitely many) nominals that they can represent, and apply the standard $\mathcal{SROIQ}$ semantics to the result [13].

Definition 1. *The* grounding $\mathsf{ground}(\alpha)$ *of a $\mathcal{SROIQV}$ axiom α is the set of all axioms that can be obtained by uniformly replacing nominal schemas in α with nominals of the given signature. Given a $\mathcal{SROIQV}$ knowledge base KB, we define* $\mathsf{ground}(KB) := \bigcup_{\alpha \in KB} \mathsf{ground}(\alpha)$.

A DL interpretation $\mathcal{I}$ is a model *of a $\mathcal{SROIQV}$ axiom α, written $\mathcal{I} \models \alpha$, if and only if $\mathcal{I}$ is a model of the knowledge base* $\mathsf{ground}(\alpha)$. *Satisfaction and entailment of $\mathcal{SROIQV}$ axioms and knowledge bases is defined as usual.*

Note that grounding does not affect the structural restrictions of simplicity and regularity. Definition 1 provides a direct approach for reasoning with $\mathcal{SROIQV}$, though not necessarily a very practical one given that each $\mathcal{SROIQV}$ axiom represents an exponential number of $\mathcal{SROIQ}$ axioms obtained by grounding. However, this observation already yields an upper bound for the complexity of reasoning with $\mathcal{SROIQV}$ that is exponentially larger than that of $\mathcal{SROIQ}$, i.e. N3ExpTime. In the remainder of this section, we prove that this result can be refined to obtain an N2ExpTime upper complexity bound, showing that this reasoning problem must be N2ExpTime-complete. To accomplish this, we extend the original proof for the worst-case complexity of $\mathcal{SROIQ}$ [8].

We first recall the complexity proof of [8] which is based on an exponential reduction of DL knowledge bases to theories of $\mathcal{C}^2$, the two-variable fragment of first-order logic with counting quantifiers, for which satisfiability can be checked in NExpTime [17]. The reduction proceeds in three steps: (1) axioms are transformed into a simplified normal form, (2) complex RIAs are eliminated, and (3) the resulting axioms are expressed as formulae of $\mathcal{C}^2$.

Step (1) yields an equisatisfiable knowledge base that contains only axioms of the following forms:

$$\begin{array}{ccc}
A \sqsubseteq \forall R.B & \bigsqcap A_i \sqsubseteq \bigsqcup B_j & S_1 \sqsubseteq S_2 \\
A \sqsubseteq \geqslant n\, S.B & A \equiv \{a\} & R_1 \sqsubseteq R^- \\
A \sqsubseteq \leqslant n\, S.B & A \equiv \exists S.\mathsf{Self} & R_1 \circ \cdots \circ R_n \sqsubseteq R
\end{array}$$

where $R_{(i)}, S_1, S_2 \in N_R$ with S_1, S_2 simple, and $C \equiv D$ is short for $\{C \sqsubseteq D, D \sqsubseteq C\}$. This normalization can be done in linear time; see [8] for details. The only axioms that are not readily expressed in $\mathcal{C}^2$ are complex RIAs. They are eliminated next, with exponential effort.

Step (2) applies a technique from [3] using nondeterministic finite automata (NFA) to represent RIAs that entail non-simple roles. Suitable NFA for $\mathcal{SROIQ}$ were defined in [4,7]. We do not repeat the details of this construction here, and merely quote the essential results. Proofs for the following facts can be found in [4] and the accompanying technical report.

Fact 1 *Consider a $\mathcal{SROIQ}$ knowledge base KB. For each (possibly inverse) non-simple role $R \in \mathbf{R}$, there is an NFA $\mathcal{A}_R$ over the alphabet N_R such that for every model $\mathcal{I}$ of KB, and for every word $S_1 \ldots S_n$ accepted by $\mathcal{A}_R$:*

$$\textit{If } \langle \delta_i, \delta_{i+1} \rangle \in S_i^{\mathcal{I}} \textit{ for all } i = 1, \ldots, n, \textit{ then } \langle \delta_1, \delta_{n+1} \rangle \in R^{\mathcal{I}}.$$

Moreover, let $\prec$ denote a strict linear order that witnesses regularity of KB as required in [4]. For each non-simple $R \in N_R$, the number of states of $\mathcal{A}_R$ is bounded exponentially in the depth *of KB that is defined as:*

$$\max\{n \mid \textit{there are } S_1 \prec \ldots \prec S_n \textit{ such that} \\ T_{i1} \circ \ldots \circ S_i \circ \ldots \circ T_{im_i} \sqsubseteq S_{i+1} \in \textit{KB}\}$$

It suffices to construct the respective NFA for non-simple roles. Now step (2) proceeds by replacing every axiom of the form $A \sqsubseteq \forall R.B$ by the following set of axioms, where $\mathcal{A}_R$ is the NFA as introduced above, and X_q are fresh concept names for each state q of $\mathcal{A}_R$:

$$\begin{array}{rl}
A \sqsubseteq X_q & q \text{ is the initial state of } \mathcal{A}_R \\
X_q \sqsubseteq \forall S.X_{q'} & \mathcal{A}_R \text{ has a transition } q \xrightarrow{S} q' \\
X_q \sqsubseteq B & q \text{ is a final state of } \mathcal{A}_R
\end{array}$$

Moreover, all complex RIAs of the form $R_1 \circ \ldots \circ R_n \sqsubseteq R$ with $n \geq 2$ are deleted. The number of new axioms (and fresh concept names) that are introduced for each axiom of the form $A \sqsubseteq \forall R.B$ is bounded by the sum of the number of states and transitions in $\mathcal{A}_R$, and the number of transitions in turn is linear in the number of role names and states. According to Fact 1, the number of axioms introduced for each axiom $A \sqsubseteq \forall R.B$ is exponentially bounded in the depth of the knowledge base. The overall size of the knowledge base after step (2) therefore is bounded by a function that is linear in the size of the knowledge base and exponential in the depth of the knowledge base.

Step (3), finally, is a simple rewriting to $\mathcal{C}^2$ that does not increase the size of the knowledge base. To obtain the main result of this section, it suffices to observe that grounding does not increase the depth of the knowledge base:

Theorem 1. *The problem of deciding the satisfiability of $\mathcal{SROIQV}$ knowledge bases is* N2ExpTime*-complete.*

Proof. The depth of KB is only affected by RBox axioms. RBox axioms are not affected by grounding, hence the depth of $\mathsf{ground}(KB)$ is equal to the depth of KB.

Since $\mathsf{ground}(KB)$ is in $\mathcal{SROIQ}$, one can apply the transformation steps (1)–(3). This yields a $\mathcal{C}^2$ theory T that is equisatisfiable to $\mathsf{ground}(KB)$ [8] and thus to KB. The size of T is linear in the size of $\mathsf{ground}(KB)$ and exponential in the depth of KB. Both measures are exponential in the size of KB, and so is T. Deciding satisfiability of T can be done in NExpTime [17], thus deciding satisfiability of KB in N2ExpTime.

Hardness follows since $\mathcal{SROIQV}$ includes $\mathcal{SROIQ}$, for which deciding satisfiability is N2ExpTime-hard [8]. □

3 A Tractable Fragment

The result that reasoning in $\mathcal{SROIQV}$ has the same worst-case complexity as $\mathcal{SROIQ}$ is encouraging, yet we are far from a practical reasoning procedure for this DL. In particular, Theorem 1 is based on a procedure that still takes exponentially longer than the original approach for $\mathcal{SROIQ}$, without this affecting the worst-case complexity. In this section, we therefore focus on identifying cases where inferencing is possible in polynomial time. This still leads to a rather expressive tractable DL. In [13], we also further discuss the relationship to the tractable profiles of OWL 2.

Concretely, we define DLs $\mathcal{SROELV}_n$ for each integer $n \geq 0$, n restricting the number of "problematic" occurrences of nominal schemas detailed below. The DLs are based on the tractable DL $\mathcal{SROEL}(\times)$, introduced as an extension of OWL EL [12]. In essence, $\mathcal{SROEL}(\times)$ is an extension of $\mathcal{EL}$ with $\top$, $\bot$, nominals, complex role inclusions, Self, and concept products [18]. Here, we only need the special concept product $\top \times \top$, denoted as the universal role U. In particular, we also omit range restrictions $R \sqsubseteq \top \times C$ since they do not contribute to our treatment.

To preserve tractability when adding nominal schemas, we must avoid the increase in the number of axioms during grounding, which is exponential in the number of nominal schemas per axiom. Unfortunately, one cannot reduce the number of nominal schemas by normal form transformations in general, since they represent complex dependencies that cannot be simplified. But there are special cases where nominal schemas on the left-hand side of TBox axioms can be eliminated, or separated using independent axioms. One such case was identified in [11] for the rule language ELP: if the dependencies expressed in a rule body are *tree-shaped* then the rule can always be reduced to a small set of normalized rules with a limited number of variables in each. For example, a rule body that consists of a conjunction $A(x) \wedge R(x, z) \wedge S(x, y) \wedge B(y) \wedge T(y, z)$ is not tree-shaped since there are parallel paths $x \overset{R}{\to} z$ and $x \overset{S}{\to} y \overset{T}{\to} z$ in the corresponding dependency structure. In our case, binary predicates are role names, unary predicates are concept names, and constant symbols correspond to nominals. Variables can either be "hidden" in the structure of the DL concept expression, or occur explicitly as nominal schemas (the latter are called *DL-safe variables* in ELP). For example, the above rule body can be expressed as a concept $A \sqcap \exists R.\{z\} \sqcap \exists S.(B \sqcap \exists T.\{z\})$.

Here, we do not introduce tree-shaped dependency structures as a general mechanism for ensuring that normal form transformations are possible, and merely identify sufficient conditions for which this is the case. An obvious condition that implies tree-shaped dependencies is that a nominal schema occurs only once, and only on the left-hand side of a TBox axiom. As in [11], the tree-shape only refers to variables (DL-safe or not), not to constants, in rule bodies. This means that nominals (our syntax for constants) disconnect a concept's dependency structure. For instance, if B in the above rule body is replaced by a nominal $\{a\}$, then the concept would be tree-shaped. In such a case, we say that the nominal $\{z\}$ occurs in a *safe environment*, as defined next.

Definition 2. *An occurrence of a nominal schema $\{x\}$ in a concept C is* safe *if C has a sub-concept of the form $\{a\} \sqcap \exists R.D$ for some $a \in N_I$, such that D contains the occurrence of $\{x\}$ but no other occurrence of any nominal schema. In this case, $\{a\} \sqcap \exists R.D$ is a* safe environment *for this occurrence of $\{x\}$. $S(a, x)$ will sometimes be used to denote an expression of the form $\{a\} \sqcap \exists R.D$ within which $\{x\}$ occurs safely.*

A nominal schema $\{x\}$ is safe *for a $\mathcal{SROIQV}$ TBox axiom $C \sqsubseteq D$ if $\{x\}$ does not occur in D, and at most one occurrence of $\{x\}$ in C is not safe.*

Definition 3. *Let $n \geq 0$. A $\mathcal{SROELV}_n$* concept *is a $\mathcal{SROIQV}$ concept that may contain $\top$, $\bot$, $\sqcap$, $\exists$,* Self*, the universal role, nominals and nominal schemas, but which does not contain $\sqcup$, $\neg$, $\forall$, $\leqslant k$, $\geqslant k$, or inverse roles.*

A $\mathcal{SROELV}_n$ TBox axiom *is a $\mathcal{SROIQV}$ TBox axiom α that uses $\mathcal{SROELV}_n$ concepts only, and where at most n nominal schemas are not safe for α. An* RBox axiom *of $\mathcal{SROELV}_n$ is an RBox axiom of $\mathcal{SROIQV}$ that does not contain inverse roles. A $\mathcal{SROELV}_n$* knowledge base *is a $\mathcal{SROIQV}$ knowledge base that contains only $\mathcal{SROELV}_n$ axioms.*

Restricting to at most n non-safe nominal schemas per axiom ensures that at most $|N_I|^n$ axioms are introduced during grounding. We will fix n at a constant small value, so this increase is polynomial. When viewing nominal schemas as a way of augmenting DL expressivity in existing applications, it seems plausible that this number remains small. Axiom (4) is an example of a $\mathcal{SROELV}_1$ axiom.

4 Reasoning with $\mathcal{SROELV}_n$

If n is constant, the problem of checking satisfiability in $\mathcal{SROELV}_n$ is possible in polynomial time w.r.t. the size of the knowledge base. To show this, we provide a polynomial transformation to the DL $\mathcal{SROEL}(\times)$ [12].

Let KB be a $\mathcal{SROELV}_n$ knowledge base. We define a $\mathcal{SROEL}(\times)$ knowledge base $\mathsf{ground}^+(KB)$ as follows. The RBox and ABox of $\mathsf{ground}^+(KB)$ are the same as the RBox and ABox of KB. For each TBox axiom $\alpha = C \sqsubseteq D \in KB$, the following axioms are added to $\mathsf{ground}^+(KB)$:

1. For each nominal schema $\{x\}$ safe for α, with safe occurrences in environments $S_i(a_i, x)$ for $i = 1, \ldots, l$, introduce a fresh concept name $O_{x,\alpha}$. For every individual $b \in N_I$ in *KB*, $\mathsf{ground}^+(KB)$ contains an axiom
$$\prod_{i=1}^{l} \exists U.S_i(a_i, b) \sqsubseteq \exists U.(\{b\} \sqcap O_{x,\alpha}),$$
where $S_i(a_i, b)$ denotes $S_i(a_i, x)$ with $\{x\}$ replaced by $\{b\}$, and the empty conjunction ($l = 0$) denotes $\top$.
2. A concept C' is obtained from C as follows. Initialize $C' \coloneqq C$. For each nominal schema $\{x\}$ that is safe for α: (a) replace all safe occurrences $S(a, x)$ in C' by $\{a\}$; (b) replace the non-safe occurrence (if any) of $\{x\}$ in C' by $O_{x,\alpha}$; (c) set $C' \coloneqq C' \sqcap \exists U.O_{x,\alpha}$. After these steps, C' contains only nominal schemas that are not safe for α, and neither for $C' \sqsubseteq D$.
Now add axioms $\mathsf{ground}(C' \sqsubseteq D)$ to $\mathsf{ground}^+(KB)$.

Theorem 2. *Given a $\mathcal{SROELV}_n$ knowledge base KB, the size of* $\mathsf{ground}^+(KB)$ *is exponential in n and polynomial in the size of KB.*

Proof. The size of the RBox and ABox of $\mathsf{ground}^+(KB)$ is linear in the size of *KB* and does not depend on n. If m is the number of individual names in *KB*, then step 1 above introduces at most mk axioms for each axiom α with k nominal schemas. This is polynomial in the size of *KB*. The second step introduces $|\mathsf{ground}(C' \sqsubseteq D)|$ many axioms, and hence at most m^n axioms for each α. □

As shown in [13], a $\mathcal{SROELV}_n$ knowledge base *KB* is satisfiable if and only if $\mathsf{ground}^+(KB)$ is satisfiable. A knowledge base is unsatisfiable if and only if it entails $\{a\} \sqsubseteq \bot$ for arbitrary $a \in N_I$. This reduces satisfiability testing to instance retrieval (checking if a is an instance of $\bot$). Using the polynomial time instance retrieval method for $\mathcal{SROEL}(\times)$ from [12], we thus obtain the following result. Hardness for P follows from the hardness of $\mathcal{SROEL}(\times)$.

Theorem 3. *If KB is a $\mathcal{SROELV}_n$ knowledge base of size s, satisfiability of KB can be reduced to instance retrieval w.r.t. a set of Datalog rules of size proportional to s^n and at most 4 variables per rule. If n is constant, the problem is* P*-complete.*

5 DL-Safe Rules

An interesting feature of nominal schemas is that they can be used to express arbitrary DL-safe rules [14]. These are Datalog rules with unary and binary predicates that are restricted – just like nominal schemas – to apply to domain elements that are represented by individual names. Identifying unary predicates with concept names, binary predicates with role names, constants with individual names, and (DL-safe) variables with the variables in nominal schemas, the syntax of DL-safe rules can be based on a DL signature. As before, we assume the signature $\Sigma = \langle N_I, N_C, N_R, N_V \rangle$ to be fixed and omit explicit references to it. The set of *terms* $\mathbf{T}$ of Σ is $N_I \cup N_V$.

Definition 4. *A* concept atom *is an expression of the form $A(t)$ with $t \in \mathbf{T}$ and $A \in N_C \cup \{\top, \bot\}$. A* role atom *is an expression of the form $R(s,t)$ with $s,t \in \mathbf{T}$ and $R \in N_R$. An* atom *is a concept or role atom.*

If B is a finite and non-empty conjunction of atoms and H is an atom, then $B \to H$ is a DL-safe rule*. B is called the* body*, and H is called the* head*. A DL-safe rule that contains at most n distinct variables is called an n-*variable rule*. A 0-variable rule is a* ground rule*. The* grounding $\mathsf{ground}(B \to H)$ *of a DL-safe rule $B \to H$ is the set of all rules that can be obtained by uniformly replacing variables in $B \to H$ with individual names of the signature.*

An interpretation $\mathcal{I}$ satisfies *a ground DL-safe rule $B \to H$, written $\mathcal{I} \models B \to H$, if either $\mathcal{I} \models H$ or $\mathcal{I} \not\models B$. An interpretation $\mathcal{I}$* satisfies *a DL-safe rule $B \to H$ if it satisfies all rules in $\mathsf{ground}(B \to H)$. A set of rules is satisfied if all of its elements are. Models, satisfiability, and entailment are defined as usual.*

Since DL-safe rules use the same models as $\mathcal{SROIQV}$, it is easy to combine DL-safe rules and DL knowledge bases. The entailment relation is immediate: a DL-safe rule or DL axiom φ is entailed by a DL knowledge base KB extended with a set of rules RB if φ is satisfied by all interpretations that satisfy both KB and RB.

Definition 5. *A syntactic transformation* dl *from atoms and DL-safe rules to $\mathcal{SROIQV}$ concepts and TBox axioms is defined as follows. For a unary atom $A(t)$ and binary atom $R(s,t)$, we set*

$$\mathsf{dl}(A(t)) \coloneqq \exists U.(\{t\} \sqcap A) \qquad \textit{and} \qquad \mathsf{dl}(R(s,t)) \coloneqq \exists U.(\{s\} \sqcap \exists R.\{t\}).$$

Given a DL-safe rule $B \to H$, we set $\mathsf{dl}(B \to H) \coloneqq \bigsqcap_{F \in B} \mathsf{dl}(F) \sqsubseteq \mathsf{dl}(H)$, and for a set of DL-safe rules RB we define $\mathsf{dl}(RB) \coloneqq \bigcup_{B \to H \in RB} \mathsf{dl}(B \to H)$.

The function dl transforms rules into $\mathcal{SROELV}_n$ TBox axioms, where n is the number of variables in the rule. This ensures that none of the restrictions on simple and non-simple roles or regularity are violated. In consequence, $\mathsf{dl}(RB)$ is a $\mathcal{SROELV}_n$ knowledge base if RB is a set of n-variable rules. The following result of [13] is not hard to show:

Theorem 4. *The models of a set RB of DL-safe rules are the same as the models of $\mathsf{dl}(RB)$, i.e. RB and $\mathsf{dl}(RB)$ are semantically equivalent.*

Importantly, this result confirms that nominal schemas are powerful enough to express arbitrary DL-safe rules. The use of nominal schemas, however, in $\mathcal{SROIQV}$ is more general than the extension of $\mathcal{SROIQ}$ with DL-safe rules, since the latter correspond to a special form of $\mathcal{SROIQV}$ axioms only. Combining Theorem 3 with the observation that $\mathsf{dl}(RB)$ is linear in the size of RB, we can state the following:

Theorem 5. *The problem of deciding whether a knowledge base $RB \cup KB$ is satisfiable, where RB is a set of n-variable rules with n constant, and KB is a $\mathcal{SROELV}_n$ knowledge base, is* P*-complete.*

6 Conclusions and Future Work

We have introduced nominal schemas as an extension to description logics, giving DLs sufficient expressivity to incorporate rule-based modeling. In particular, the use of nominal schemas supports the integration of DL-safe rules into DL knowledge bases. An important next step is to realize these ideas for the concrete serialization formats of these languages, and to make the corresponding modeling features available in practice.

The latter task especially includes the implementation of inference algorithms to handle nominal schemas more efficiently. We have shown that our extension does not increase the worst-case complexity of reasoning in $\mathcal{SROIQ}$, and that versatile tractable sublanguages exist. Whether and how these theoretical results can be put into efficient reasoning algorithms is an open research question. Two different approaches to addressing this problem appear viable. On the one hand, nominal schemas could be implemented by modifying/extending existing $\mathcal{SROIQ}$ implementations that have good support for nominals, such as the OWL 2 reasoner *HermiT* [15]. This can be accomplished by treating nominal schemas like nominals in the deduction procedure, instantiating them with concrete individuals only when this enables relevant deduction steps. This can be viewed as a method of *deferred grounding*.

On the other hand, our light-weight description logics could be implemented using rule-based procedures as proposed for $\mathcal{SROEL}$ [12]. In this setting, nominal schemas can be treated like DL-safe variables. Thus, the rule-based deduction remains similar with the only modification that some variables can only be instantiated with certain constants. Specifically, the approach in [12] introduces new constant symbols for eliminating existentials, and DL-safe variables must not be allowed to represent these auxiliary symbols.

In conclusion, the close relationship to nominals is not merely of syntactic convenience but prepares a path for the further practical adoption of this feature. Instead of a paradigm shift from ontologies to rules, existing applications could be augmented with bits of rule-based modeling to overcome restrictions of classical DLs. Nominal schemas thus may provide an opportunity for enhancing the expressive power of ontologies without giving up on established tools, formats, or methodologies.

Acknowledgements This work was partially supported by the National Science Foundation under award 1017225 "III: Small: TROn—Tractable Reasoning with Ontologies" and by EPSRC in project "HermiT: Reasoning with Large Ontologies" (EP/F065841/1). The third author acknowledges support by a Fulbright Indonesia Presidential Scholarship PhD Grant 2010.

References

1. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-core/

2. de Bruijn, J.: RIF RDF and OWL Compatibility. W3C Recommendation (22 June 2010), available at http://www.w3.org/TR/rif-rdf-owl/
3. Demri, S., Nivelle, H.: Deciding regular grammar logics with converse through first-order logic. J. of Logic, Language and Information 14(3), 289–329 (2005)
4. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
5. Horrocks, I., Patel-Schneider, P., Bechhofer, S., Tsarkov, D.: OWL Rules: A proposal and prototype implementation. J. of Web Semantics 3(1), 23–40 (2005)
6. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004), see http://www.w3.org/Submission/SWRL/
7. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. Artificial Intelligence 160(1), 79–104 (2004)
8. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08). pp. 274–284. AAAI Press (2008)
9. Krötzsch, M.: Description Logic Rules, Studies on the Semantic Web, vol. 008. IOS Press/AKA (2010)
10. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Ghallab, M., et al. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008. pp. 80–84. IOS Press (2008)
11. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In: Sheth, A., et al. (eds.) Proceedings of the 7th International Semantic Web Conference (ISWC-08). Lecture Notes in Computer Science, vol. 5318, pp. 649–664. Springer (2008)
12. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: Walsh, T. (ed.) Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'11). IJCAI (2011), to appear
13. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Proc. 20th Int. Conf. on World Wide Web (WWW'11). pp. 645–654. ACM (2011)
14. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. J. of Web Semantics 3(1), 41–60 (2005)
15. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36(1), 165–228 (2009)
16. Patel-Schneider, P., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-mapping-to-rdf/
17. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. J. of Logic, Language and Information 14, 369–395 (2005)
18. Rudolph, S., Krötzsch, M., Hitzler, P.: Cheap Boolean role constructors for description logics. In: Hölldobler, S., et al. (eds.) Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08). LNAI, vol. 5293, pp. 362–374. Springer (2008)

On P/NP Dichotomies for $\mathcal{EL}$ Subsumption under Relational Constraints

A. Kurucz[1], F. Wolter[2], and M. Zakharyaschev[3]

[1] Department of Informatics
King's College London, U.K.
`agi.kurucz@kcl.ac.uk`
[2] Department of Computer Science
University of Liverpool, U.K.
`frank@csc.liv.ac.uk`
[3] Department of Computer Science and Information Systems,
Birkbeck College London, U.K.
`michael@dcs.bbk.ac.uk`

Abstract. We consider the problem of characterising relational constraints under which TBox reasoning in $\mathcal{EL}$ is tractable. We obtain P vs. coNP-hardness dichotomies for tabular constraints and constraints imposed on a single reflexive role.

1 Introduction

In recent years, the problem of describing role boxes (aka relational constraints) under which reasoning is within a given complexity class has become an important research topic in description logic (DL). For example, the development of $\mathcal{SROIQ}$ from $\mathcal{SHIQ}$ has mainly been driven by the desire to allow for more expressive relational constraints for which reasoning is still decidable and tableau decision procedures can be developed. As a result, in $\mathcal{SROIQ}$ one can express, among others, role inclusions of the form $r \circ s \sqsubseteq r$ and $s \circ r \sqsubseteq r$, reflexivity, transitivity and symmetry of roles [6, 7].

For $\mathcal{EL}$, underlying the *OWL 2 EL* profile of the *OWL 2* Web Ontology Language, the complexity of reasoning under relational constraints was investigated in [1, 2, 9]. For example, the subsumption problem for general TBoxes in $\mathcal{EL}$ is tractable for any finite set of constraints of the form

$$r_1(x_1, x_2) \land \cdots \land r_n(x_n, x_{n+1}) \to r_{n+1}(x_1, x_{n+1}) \qquad (1)$$

(the order of the variables is essential). On the other hand, subsumption becomes ExpTime-complete in the presence of symmetry or functionality constraints [2].

The aim of this paper is to take a fresh look at how relational constraints influence the complexity of DL reasoning: rather than putting forward a new class of role boxes for which reasoning is decidable or within a certain complexity class, we attempt to *classify* relational constraints according to whether they lead to decidable or undecidable reasoning problems, or to reasoning within a given complexity bound. The ultimate aim of this approach is to obtain a complete map of how relational constraints determine the complexity of reasoning for most important DLs. Apart from its theoretical interest, such a map can also be used for the selection of role boxes with acceptable computational properties in future standardisation efforts.

In this paper, which extends [8], we take first steps in this program by starting to map out the border between tractability and intractability of TBox reasoning in $\mathcal{EL}$ under arbitrary relational constraints. One of the fundamental questions (left unanswered in this paper) is the following

Dichotomy Question: Is it the case that for any relational constraint, TBox reasoning in $\mathcal{EL}$ is either in P or coNP-hard?

By Ladner's Theorem, unless P = coNP, there exist problems that are coNP-intermediate (neither in P nor coNP-hard). The existence of relational constraints for which TBox reasoning in $\mathcal{EL}$ is coNP-intermediate would indicate that a general and complete map of the boundary between tractable and intractable is extremely hard to obtain. In contrast, a positive answer would probably come with an informative description of the tractable constraints.

Our initial findings indicate that informative dichotomy results on P versus coNP-hardness can indeed be obtained. For example, we show that

(d1) there are only *four universal* constraints on a single reflexive role r under which $\mathcal{EL}$ TBox reasoning is in P: (1) r is arbitrary, (2) the domain of r is a singleton, (3) r is transitive, (4) r is an equivalence relation. All other universal constraints are either invisible to $\mathcal{EL}$ TBox reasoning or lead to coNP-hard $\mathcal{EL}$ subsumption.

Here, by 'invisibility' we understand the following. It is well known that many relational constraints do not influence—or are invisible to—TBox reasoning: for example, for $\mathcal{EL}$ (and even $\mathcal{ALC}$), TBox reasoning over irreflexive relations coincides with TBox reasoning over arbitrary relations, and similarly for the class of finite and tree-like relational structures. In fact, one can use dichotomy **(d1)** to show that there are uncountably many 'visible' universal relational constraints on a single reflexive role for which $\mathcal{EL}$ subsumption is coNP-hard, but only four 'visible' universal constraints for which $\mathcal{EL}$ subsumption is in P.

Another dichotomy we prove in this paper is as follows:

(d2) Consider an arbitrary relational constraint (over a finite number of roles) such that the size of the domain of all interpretations satisfying this constraint is bounded by some natural number $n > 0$. Then $\mathcal{EL}$ subsumption over the interpretations satisfying the constraint is in P if all roles in those interpretations are functional. Otherwise $\mathcal{EL}$ subsumption is coNP-complete.

Currently, not much is known about dichotomies for more expressive languages. We note, however, recent work on an NP vs. PSpace dichotomy for satisfiability of classical modal formulas over frame classes definable by Horn sentences [5].

The paper is structured as follows. In Section 2, we define the extension $\mathcal{EL}_\bot$ of $\mathcal{EL}$ with the concept $\bot$ and all the model-theoretic notions we need. We prove our results for $\mathcal{EL}_\bot$ rather than $\mathcal{EL}$ and show, by a straightforward reduction in Section 6, that they hold for $\mathcal{EL}$ as well. In Section 3, we consider the relation between tractability and convexity (the disjunction property) and prove two general sufficient conditions for non-tractability. Then, in Sections 4 and 5, we prove the dichotomies **(d1)** and **(d2)** mentioned above.

2 Preliminaries

Fix two disjoint countably infinite sets NC of *concept names* and NR of *role names*. We use arbitrary concept names in NC for constructing concepts, but may restrict the set of available role names to some $\mathsf{R} \subseteq \mathsf{NR}$. Throughout this paper, we work with $\mathcal{EL}$ extended with the concept $\bot$, denoting the empty set. Thus, for $\mathsf{R} \subseteq \mathsf{NR}$, the $\mathcal{EL}_\bot$-*concepts* C *over* R are defined inductively as follows:

$$C \quad ::= \quad \top \quad | \quad \bot \quad | \quad A \quad | \quad C_1 \sqcap C_2 \quad | \quad \exists r.C,$$

where $A \in \mathsf{NC}$, $r \in \mathsf{R}$ and C, C_1, C_2 range over $\mathcal{EL}_\bot$-concepts over R. An R-*TBox* is a finite set of *concept inclusions* (CIs) $C \sqsubseteq D$, where C and D are $\mathcal{EL}_\bot$-concepts over R. An R-*interpretation* is of the form $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I} \neq \emptyset$ and $\cdot^\mathcal{I}$ is an *interpretation function* for concept names and role names in R. Complex concepts over R are interpreted in $\mathcal{I}$ as usual. If $C^\mathcal{I} \subseteq D^\mathcal{I}$, we say that $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ and write $\mathcal{I} \models C \sqsubseteq D$. $\mathcal{I}$ is a *model* of an R-TBox $\mathcal{T}$, $\mathcal{I} \models \mathcal{T}$ in symbols, if it satisfies all the CIs in $\mathcal{T}$.

We now define what we understand by *relational constraints* on interpretations. An R-*frame* is a structure $\mathfrak{F} = (\Delta^\mathfrak{F}, \cdot^\mathfrak{F})$ where $\Delta^\mathfrak{F} \neq \emptyset$ and $\cdot^\mathfrak{F}$ is a map associating with each $r \in \mathsf{R}$ a relation $r^\mathfrak{F} \subseteq \Delta^\mathfrak{F} \times \Delta^\mathfrak{F}$. We say that an R-interpretation $\mathcal{I}$ is *based on* an R-frame $\mathfrak{F}$ if $\Delta^\mathcal{I} = \Delta^\mathfrak{F}$ and $r^\mathcal{I} = r^\mathfrak{F}$ for all $r \in \mathsf{R}$. An R-*constraint* is any class $\mathcal{K}$ of R-frames closed under isomorphic copies. For example, a constraint for $\mathsf{R} = \{r_1, r_2, r_3\}$ can consist of all R-frames $\mathfrak{F} = (\Delta^\mathfrak{F}, \cdot^\mathfrak{F})$ with arbitrary $r_1^\mathfrak{F}$, transitive $r_2^\mathfrak{F}$ and functional $r_3^\mathfrak{F}$. An interpretation $\mathcal{I}$ *satisfies* an R-constraint $\mathcal{K}$ if $\mathcal{I}$ is based on some $\mathfrak{F} \in \mathcal{K}$.

The *subsumption problem for an* R-*constraint* $\mathcal{K}$ is to decide, given an R-TBox $\mathcal{T}$ and two concepts C, D over R, whether $\mathcal{I} \models C \sqsubseteq D$ for every model $\mathcal{I}$ of $\mathcal{T}$ based on an R-frame in $\mathcal{K}$, in which case we write $\mathcal{T} \models_\mathcal{K} C \sqsubseteq D$. For singleton $\mathcal{K} = \{\mathfrak{F}\}$, we sometimes write $\mathcal{T} \models_\mathfrak{F} C \sqsubseteq D$.

Example 1. In the extension $\mathcal{EL}_\bot^+$ of $\mathcal{EL}_\bot$ [1], along with a TBox one can define an *RBox* containing inclusions of the form $r_1 \circ \cdots \circ r_n \sqsubseteq r_{n+1}$, where $r_1, \dots, r_{n+1}$ are role names. Reasoning with RBoxes $\mathcal{R}$ is clearly captured by the frame condition $\mathcal{K}_\mathcal{R}$ containing all NR-frames $\mathfrak{F}$ such that

$$\mathfrak{F} \models \forall x_1 \dots \forall x_{n+1} \ \big(r_1(x_1, x_2) \land \cdots \land r_n(x_n, x_{n+1}) \to r_{n+1}(x_1, x_{n+1})\big)$$

for all $r_1 \circ \cdots \circ r_n \sqsubseteq r_{n+1}$ in $\mathcal{R}$. According to [1, 9], the subsumption problem for any such $\mathcal{K}_\mathcal{R}$ is decidable in P. On the other hand, the subsumption problem for the class of symmetric frames is ExpTime-complete [2].

We say that R-constraints $\mathcal{K}_1$ and $\mathcal{K}_2$ are *TBox-equivalent* (in $\mathcal{EL}_\bot$) if we have $\mathcal{T} \models_{\mathcal{K}_1} C \sqsubseteq D$ iff $\mathcal{T} \models_{\mathcal{K}_2} C \sqsubseteq D$, for all R-TBoxes $\mathcal{T}$ and $\mathcal{EL}_\bot$-concepts C, D over R. For example, as is well-known, the class of all frames is TBox equivalent to the class of all irreflexive frames, and to the class of all finite frames. For an R-constraint $\mathcal{K}$, we denote by $\mathrm{FrTh}\mathcal{K}$ the union of all those R-constraints that are TBox equivalent in $\mathcal{EL}_\bot$ to $\mathcal{K}$. $\mathrm{FrTh}\mathcal{K}$ and $\mathcal{K}$ are TBox equivalent in $\mathcal{EL}_\bot$, and $\mathrm{FrTh}\mathcal{K}$ is the largest class that is TBox equivalent in $\mathcal{EL}_\bot$ to $\mathcal{K}$.

An R-constraint $\mathcal{K}$ is *TBox-definable* (in $\mathcal{EL}_\bot$) if there exists a set Γ of pairs $(\mathcal{T}, C \sqsubseteq D)$, where $\mathcal{T}$ is an R-TBox and C, D are $\mathcal{EL}_\bot$-concepts over R, such that $\mathcal{K} = \{\mathfrak{F} \mid \mathcal{T} \models_{\mathfrak{F}} C \sqsubseteq D, \text{ for all } (\mathcal{T}, C \sqsubseteq D) \in \Gamma\}$. Thus, $\mathcal{K}$ is TBox-definable iff $\mathcal{K} = \mathrm{FrTh}\mathcal{K}$, and any class of TBox-equivalent constraints contains exactly one TBox-definable class. In a similar way we can define TBox-definable classes of R-constraints for $\mathcal{EL}$ and more expressive DLs, say $\mathcal{ALC}$.

A *universal* R-constraint is a class of R-frames definable by universal first-order sentences in the signature R. Equivalently, by [10], a universal constraint is a first-order definable class of frames closed under taking subframes. The vast majority of frame constraints considered in modal and description logics are universal: transitivity, reflexivity, symmetry, weak linearity, just to mention a few. Typical examples of non-universal (first-order) constraints are the Church-Rosser property and density. As far as universal R-constraints are concerned, $\mathcal{EL}_\bot$ defines the same R-constraints as $\mathcal{ALC}$ (the proof is given in [8]):

Theorem 1. *Let $\mathcal{K}$ be a universal class of R-frames, for some $\mathsf{R} \subseteq \mathsf{NR}$. Then $\mathcal{K}$ is TBox-definable in $\mathcal{EL}_\bot$ iff it is TBox-definable in $\mathcal{ALC}$.*

We conjecture that Theorem 1 can be generalised to arbitrary (not necessarily first-order definable) classes of R-frames closed under subframes. Note that, without the subframe condition, there are classes of frames that are TBox-definable in $\mathcal{ALC}$ but not in $\mathcal{EL}_\bot$. One example is the *Church-Rosser property*

$$\forall x, y_1, y_2 \big(r(x, y_1) \land r(x, y_2) \to \exists z (r(y_1, z) \land r(y_2, z))\big).$$

3 Tractability and Convexity

In this section, we investigate the relationship between convexity (sometimes also called the disjunction property) and tractability. To this end, we need (formally not allowed in $\mathcal{EL}_\bot$) concepts of the form $C \sqcup D$, where C and D are $\mathcal{EL}_\bot$-concepts, which are interpreted in the obvious way by the union of the extensions of the disjuncts C and D. An R-constraint $\mathcal{K}$ is said to be *convex* if, for any R-TBox $\mathcal{T}$ and $\mathcal{EL}_\bot$-concepts F, C, D over R,

(conv) if $\mathcal{T} \models_{\mathcal{K}} F \sqsubseteq C \sqcup D$ then $\mathcal{T} \models_{\mathcal{K}} F \sqsubseteq C$ or $\mathcal{T} \models_{\mathcal{K}} F \sqsubseteq D$.

Although convexity is closely related to tractability, they do not imply each other. It is readily checked that every relational constraint $\mathcal{K}$ defined by Horn sentences is convex. Thus, symmetry and functionality are examples of relational constraints that are convex but non-tractable [2]. The following example shows that tractability of $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ does not imply that $\mathcal{K}$ is convex:

Example 2. Consider the smallest class $\mathcal{K}$ of R-frames, for $\mathsf{R} = \{s, r, r'\}$, which is closed under subframes and contains all two-element irreflexive s-chains such that if $s(x, y)$ then either $r(x, y)$ or $r'(x, y)$. Thus, $\mathcal{K}$ is a universal constraint and $\emptyset \models_{\mathcal{K}} \exists s.\top \sqsubseteq \exists r.\top \sqcup \exists r'.\top$. As, $\emptyset \not\models_{\mathcal{K}} \exists s.\top \sqsubseteq \exists r.\top$ and $\emptyset \not\models_{\mathcal{K}} \exists s.\top \sqsubseteq \exists r'.\top$, $\mathcal{K}$ is not convex. On the other hand, as will be shown in the next section (see Theorem 4), $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is in P.

We now prove two general conditions, based on non-convexity, that imply non-tractability. The proofs of coNP-hardness are by reduction of the following *set splitting problem*, which is known to be NP-complete [4]:

– given a family I of subsets of a finite set S, decide whether there exists a *splitting* of (S, I), i.e., a partition S_1, S_2 of S such that each set $G \in I$ is split by S_1 and S_2 in the sense that it is not the case that $G \subseteq S_i$ for $i \in \{1, 2\}$.

We say that a class $\mathcal{K}$ of R-frames is *concept non-convex* if, for some R-TBox $\mathcal{T}$ and concepts F, C, D over R, we have $\mathcal{T} \models_{\mathcal{K}} F \sqsubseteq C \sqcup D$, and there exist an R-frame $\mathfrak{F} \in \mathrm{FrTh}\mathcal{K}$, a point $x \in \Delta^{\mathfrak{F}}$ and two models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $\mathcal{T}$ based on $\mathfrak{F}$ such that $x \in F^{\mathcal{I}_1} \setminus D^{\mathcal{I}_1}$ and $x \in F^{\mathcal{I}_2} \setminus C^{\mathcal{I}_2}$. Our main tool for proving non-tractability results is the following:

Theorem 2. *If a class $\mathcal{K}$ of R-frames is concept non-convex, then $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is coNP-hard.*

Proof. Consider $\mathcal{T}$, F, C and D over R for which $\mathcal{T} \models_{\mathcal{K}} F \sqsubseteq C \sqcup D$, and there exist an R-frame $\mathfrak{F} \in \mathcal{K}$ with $x \in \Delta^{\mathfrak{F}}$ and two models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $\mathcal{T}$ based on $\mathfrak{F}$ such that $x \in F^{\mathcal{I}_1} \setminus D^{\mathcal{I}_1}$ and $x \in F^{\mathcal{I}_2} \setminus C^{\mathcal{I}_2}$. Suppose (S, I) is an instance of the set splitting problem. Denote by $\mathcal{T}_i$, F_i, C_i and D_i, for $i \in S$, the copies of $\mathcal{T}$, F, C and D obtained by replacing every concept name A in them with A_i. Let

$$\mathcal{T}_{S,I} = \bigcup_{i \in S} \mathcal{T}_i \cup \{\bigsqcap_{i \in G} (B \sqcap C_i) \sqsubseteq \bot \mid G \in I\} \cup \{\bigsqcap_{i \in G} (B \sqcap D_i) \sqsubseteq \bot \mid G \in I\},$$

where B is a fresh concept name. We show now that there exists a splitting of (S, I) i ff$\mathcal{T}_{S,I} \not\models_{\mathcal{K}} \bigsqcap_{i \in S}(B \sqcap F_i) \sqsubseteq \bot$. ($\Rightarrow$) Let S_1, S_2 be a splitting of (S, I). Define an interpretation $\mathcal{I}$ on $\mathfrak{F}$ by taking $A_i^{\mathcal{I}} = A^{\mathcal{I}_1}$ if $i \in S_1$, $A_i^{\mathcal{I}} = A^{\mathcal{I}_2}$ if $i \in S_2$, for all concept names A different from B, and $B^{\mathcal{I}} = \{x\}$. One can readily check that $\mathcal{I} \models \mathcal{T}_{S,I}$ and $\mathcal{I} \not\models \bigsqcap_{i \in S}(B \sqcap F_i) \sqsubseteq \bot$. ($\Leftarrow$) Suppose that $\mathcal{I} \models \mathcal{T}_{S,I}$ and there is $y \in \bigcap_{i \in S}(B^{\mathcal{I}} \cap F_i^{\mathcal{I}})$. We then set $S_1 = \{i \in S \mid y \in C_i^{\mathcal{I}}\}$ and $S_2 = S \setminus S_1$. It is readily checked that S_1, S_2 is a splitting of (S, I).

An R-constraint $\mathcal{K}$ is *closed under disjoint unions* if, for any $\mathfrak{F}_1, \mathfrak{F}_2 \in \mathcal{K}$ with $\Delta^{\mathfrak{F}_1} \cap \Delta^{\mathfrak{F}_2} = \emptyset$, we have $\mathfrak{F}_1 \cup \mathfrak{F}_2 \in \mathrm{FrTh}\mathcal{K}$, where $\Delta^{\mathfrak{F}_1 \cup \mathfrak{F}_2} = \Delta^{\mathfrak{F}_1} \cup \Delta^{\mathfrak{F}_2}$ and $r^{\mathfrak{F}_1 \cup \mathfrak{F}_2} = r^{\mathfrak{F}_1} \cup r^{\mathfrak{F}_2}$. We also say that $\mathcal{K}$ has a *free role* r if, for any $\mathfrak{F} \in \mathcal{K}$ and any $x, y \in \Delta^{\mathfrak{F}}$, the frame obtained by extending $r^{\mathfrak{F}}$ in $\mathfrak{F}$ with the pair (x, y) belongs to $\mathrm{FrTh}\mathcal{K}$. Note that all RBoxes, currently used in DL, correspond to constraints that are closed under disjoint unions and have infinitely many free roles (since typically DLs admit infinitely many role names and have finite RBoxes). The following condition is proved similarly to Theorem 2:

Theorem 3. *Suppose that an* R*-constraint* $\mathcal{K}$ *is closed under disjoint unions and has infinitely many free roles. If* $\mathcal{K}$ *is not convex then* $\mathcal{EL}_\bot$ *subsumption over* $\mathcal{K}$ *is* coNP*-hard.*

4 P/coNP Dichotomy for Tabular Constraints

A class $\mathcal{K}$ of R-frames is called *tabular* if there is $n > 0$ such that $|\Delta^{\mathfrak{F}}| \leq n$ for all $\mathfrak{F} \in \mathcal{K}$. The aim of this section is to characterise the tabular constraints $\mathcal{K}$ over which $\mathcal{EL}_\bot$ subsumption is tractable, that is, there is an algorithm which, given a TBox $\mathcal{T}$ and concepts C, D over R, can decide, in polynomial time, whether $\mathcal{T} \models_{\mathcal{K}} C \sqsubseteq D$. Clearly, $\mathcal{EL}_\bot$ subsumption over any tabular $\mathcal{K}$ belongs to coNP.

The characterisation of tabular constraints we are about to prove dichotomises them into functional and non-functional. A class $\mathcal{K}$ of R-frames is R*-functional* if, for any $\mathfrak{F} \in \mathcal{K}$, $r \in \mathsf{R}$ and $w \in \Delta^{\mathfrak{F}}$, we have $|\{v \in \Delta^{\mathfrak{F}} \mid (w,v) \in r^{\mathfrak{F}}\}| \leq 1$. For R-interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ based on a functional frame $\mathfrak{F}$, we write $\mathcal{I}_1 \leq \mathcal{I}_2$ if $A^{\mathcal{I}_1} \subseteq A^{\mathcal{I}_2}$ for all $A \in \mathsf{NC}$. Clearly, $\leq$ is a partial order.

Lemma 1. *Suppose that* $\mathcal{I}$ *is an interpretation based on a finite* R*-functional frame* $\mathfrak{F}$ *and* $w \in \Delta^{\mathcal{I}}$*. Given any* R*-concept* C*, one can decide in polynomial time in* $|C|$ *whether there exists an* R*-interpretation* $\mathcal{J}$ *such that* $\mathcal{I} \leq \mathcal{J}$ *and* $w \in C^{\mathcal{J}}$*. If such an interpretation exists, then there is a unique minimal (with respect to* $\leq$*)* R*-interpretation* $\mathcal{I}(w,C) \geq \mathcal{I}$ *with* $w \in C^{\mathcal{I}(w,C)}$*; moreover, this minimal interpretation can be constructed in polynomial time in* $|C|$*.*

We are now in a position to prove the main result of this section.

Theorem 4. *Let* $\mathcal{K}$ *be a tabular class of* R*-frames for a finite* $\mathsf{R} \subseteq \mathsf{NR}$*. If* $\mathcal{K}$ *is functional then* $\mathcal{EL}_\bot$ *subsumption over* $\mathcal{K}$ *is in* P*. Otherwise,* $\mathcal{EL}_\bot$ *subsumption over* $\mathcal{K}$ *is* coNP*-complete.*

Proof. Assume first that $\mathcal{K}$ is functional and we are given a TBox $\mathcal{T}$ and a CI $C' \sqsubseteq D'$ over R. Our polynomial time algorithm checking whether $\mathcal{T} \models_{\mathcal{K}} C' \sqsubseteq D'$ runs as follows. Let $\mathfrak{F}_1, \dots, \mathfrak{F}_n$ be a list of all frames in $\mathcal{K}$ (up to isomorphism). For each $\mathfrak{F}_i$ and each $w \in \mathfrak{F}_i$, we do the following:

1. Let $\mathcal{I}$ be the R-interpretation based on $\mathfrak{F}_i$ with $A^{\mathcal{I}} = \emptyset$ for all $A \in \mathsf{NC}$.
2. Compute $\mathcal{I} := \mathcal{I}(w, C')$ if it exists (cf. Lemma 1). If it does not exist, return 'yes' and stop.
3. Apply the following rule exhaustively: for $C \sqsubseteq D \in \mathcal{T}$ and $v \in \Delta^{\mathcal{I}}$, if $v \in C^{\mathcal{I}}$ and $\mathcal{I}(v, D)$ does not exist, return 'yes' and stop; otherwise, if $\mathcal{I}(v,D) \neq \mathcal{I}$, set $\mathcal{I} = \mathcal{I}(v, D)$.
4. If $w \in (D')^{\mathcal{I}}$, return 'yes.' Otherwise, return 'no.'

It is easy to see that $\mathcal{T} \models_{\mathcal{K}} C' \sqsubseteq D'$ iff the output is 'yes' for all $\mathfrak{F}_i$ and $w \in \Delta^{\mathfrak{F}_i}$.

Suppose $\mathcal{K}$ is not R-functional. Then there exists $\mathfrak{F} \in \mathcal{K}$ with $w \in \Delta^{\mathfrak{F}}$ such that $|\{v \mid (w,v) \in r^{\mathfrak{F}}\}| \geq 2$. Let m be the maximal number for which there exist $r \in \mathsf{R}$, $\mathfrak{F} \in \mathcal{K}$ and $w \in \Delta^{\mathfrak{F}}$ with $|\{v \mid (w,v) \in r^{\mathfrak{F}}\}| = m$. Fix such r, $\mathfrak{F}$ and w. We prove coNP-hardness of $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ using Theorem 2. To show that $\mathcal{K}$ is concept non-convex, consider the $\{r\}$-TBox $\mathcal{T}$ with the following CIs:

– $A \sqsubseteq \exists r.B_i$, for $1 \leq i \leq m$;
– $B_i \sqcap B_j \sqsubseteq \bot$, for $1 \leq i < j \leq m$;
– $A \sqsubseteq \exists r.B$
– $B_i \sqsubseteq E$, for $2 \leq i \leq m$.

Clearly, $\mathcal{T} \models_{\mathcal{K}} A \sqsubseteq \exists r.(B \sqcap B_1) \sqcup \exists r.(B \sqcap E)$. Consider next the interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ over $\mathfrak{F}$ where $w_1, \dots, w_m$ are the $r^{\mathfrak{F}}$-successors of w in $\mathfrak{F}$ and

– $A^{\mathcal{I}_i} = \{w\}$ and $B^{\mathcal{I}_i} = \{w_i\}$, for $i = 1, 2$;
– $B_j^{\mathcal{I}_i} = \{w_j\}$, for $i = 1, 2$ and $1 \leq j \leq m$;
– $E^{\mathcal{I}_i} = \{w_2, \dots, w_m\}$, for $i = 1, 2$.

Then we have $\mathcal{I}_i \models \mathcal{T}$, $w \in A^{\mathcal{I}_1} \setminus (\exists r.(B \sqcap E))^{\mathcal{I}_1}$ and $w \in A^{\mathcal{I}_2} \setminus (\exists r.(B \sqcap B_1))^{\mathcal{I}_2}$. By Theorem 2, $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is coNP-hard. And as we have mentioned above, $\mathcal{EL}_\bot$ subsumption for tabular constraints is in coNP.

The above proof of coNP-hardness goes through for many other constraints:

Theorem 5. *Let $\mathcal{K}$ be a class of* R*-frames such that there are $r \in$ R and $n \geq 2$ for which (i) no point in frames from $\mathcal{K}$ has $> n$ r-successors, and (ii) at least one point in a frame from $\mathcal{K}$ has ≥ 2 r-successors. Then $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is* coNP*-hard.*

5 P/coNP-hardness Dichotomy for Universal Reflexive Constraints

In this section, we assume that R $= \{r\}$ and consider *universal* classes of R-frames $\mathfrak{F}$ with *reflexive* $r^{\mathfrak{F}}$.

Theorem 6. *Let $\mathcal{K}$ be a universal constraint for a single reflexive relation. If $\mathcal{K}$ is not TBox equivalent to any of the following classes:*

(sin) *the class of all singleton frames,*
(tra) *the class of all transitive frames,*
(equ) *the class of all equivalence relations,*
(all) *the class of all frames,*
(sym) *the class of all symmetric frames,*

then $\mathcal{K}$ is concept non-convex, and so $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is coNP*-hard. $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is also* coNP*-hard if $\mathcal{K}$ is TBox equivalent to* **(sym)**. *However, if $\mathcal{K}$ is TBox equivalent to one of* **(sin)**, **(tra)**, **(equ)** *or* **(all)**, *then $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is in* P.

Note that there are uncountably many distinct universal TBox definable classes of frames with a single reflexive relation (see [8], where this is proved for quasi-orders). Thus, only four out of uncountably many possible constraints lead to tractable TBox reasoning; for all the rest, $\mathcal{EL}_\bot$ subsumption is coNP-hard.

Here we only give a brief sketch of the proof of Theorem 6. Note first that the polynomial upper bound follows from [1, 8]; non-tractability for **(sym)** is shown similarly to Theorem 7 below. To prove the remaining claim, we require

Lemma 2. *Let $\mathcal{K}$ be a universal class of reflexive frames.*

- *If $\mathcal{K}$ is not TBox equivalent to* **(all)***, then there exists a finite reflexive tree $\mathfrak{F}$ such that $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$.*
- *If $\mathcal{K}$ consists of symmetric frames and is not TBox equivalent to* **(sym)***, then there exists a finite reflexive and symmetric tree $\mathfrak{F}$ such that $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$.*
- *If $\mathcal{K}$ consists of transitive frames and is not TBox equivalent to* **(tra)***, then there exists a finite reflexive and transitive tree $\mathfrak{F}$ such that $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$.*

Proof sketch. We prove the first claim; the remaining ones are treated similarly. As $\mathcal{K}$ is not TBox equivalent to **(all)**, there are $\mathcal{T}$, C, D such that $\mathcal{T} \models_{\mathcal{K}} C \sqsubseteq D$ and $\mathcal{T} \not\models_{\mathcal{K}'} C \sqsubseteq D$, where $\mathcal{K}'$ is the class of all frames. By applying standard unravelling to a witness interpretation for $\mathcal{T} \not\models_{\mathcal{K}'} C \sqsubseteq D$, we obtain a (possibly infinite) reflexive tree $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$. If $\mathfrak{F}$ is finite, we are done. Otherwise, using the fact that $\mathcal{K}$ is universal and employing Tarski's finite embedding property [10], we can show that there is a finite subtree $\mathfrak{F}'$ of $\mathfrak{F}$ such that $\mathfrak{F}' \notin \mathrm{FrTh}_T\mathcal{K}$.

Having Lemma 2 at hand, we can now proceed with a case distinction. Suppose $\mathcal{K}$ is a non-empty universal class of reflexive frames that is not TBox equivalent to any of the classes mentioned in Theorem 6. Then, by Lemma 2, there exists a reflexive tree $\mathfrak{F}$ such that $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$, $\mathfrak{F}' \in \mathrm{FrTh}\mathcal{K}$, for any *proper* subframe $\mathfrak{F}'$ of $\mathfrak{F}$, and one of the following conditions holds:

1. $\mathfrak{F}$ is the singleton frame;
2. $\mathfrak{F}$ is the two-element r-chain;
3. $\mathfrak{F}$ contains a point w with least two r-successors, and all r-successors of w are leaves in $\mathfrak{F}$;
4. $\mathfrak{F}$ contains distinct points w, w_1, w_2 such that $(w, w_1) \in r^{\mathfrak{F}}$, $(w_1, w_2) \in r^{\mathfrak{F}}$ and w_2 is a leaf, which is the only r-successor of w_1.

Case 1. This case is actually impossible because it implies that $\mathcal{K}$ is empty (remember that $\mathcal{K}$ is universal, and so closed under subframes).

Case 2. In this case, $\mathcal{K}$ is a class of symmetric frames. Since we assume that $\mathcal{K}$ is not TBox equivalent to **(sym)**, one can apply the second claim of Lemma 2 to obtain a finite reflexive and symmetric tree $\mathfrak{F}$ such that $\mathfrak{F} \notin \mathrm{FrTh}\mathcal{K}$. A case distinction (similar to the one we are currently doing) shows that, since $\mathcal{K}$ is not TBox equivalent to **(sin)**, $\mathcal{K}$ is concept non-convex.

Case 3. Let us remove a proper r-successor of w from $\mathfrak{F}$ and denote by $\mathfrak{H}$ the resulting frame, which belongs to $\mathrm{FrTh}\mathcal{K}$. Let w_1 be one of the remaining successors of w in $\mathfrak{H}$. Denote by $\mathfrak{H}'$ the frame obtained from $\mathfrak{H}$ by adding a fresh r-successor w_2 to w_1, and by w_0 the root of $\mathfrak{H}'$. Two cases are possible now.

Case 3.1: either $\mathfrak{H}' \in \mathrm{FrTh}\mathcal{K}$ or the expansion of $\mathfrak{H}'$ by adding (w, w_2) to $r^{\mathfrak{H}'}$ is in $\mathrm{FrTh}\mathcal{K}$. Take additional concept names A and $\bar{A}$. To show that $\mathcal{K}$ is concept non-convex, we will use $C_1 = \exists r^2.(A' \sqcap \exists r^2.\bar{A}')$ and $C_2 = \exists r^2.(\bar{A}' \sqcap \exists r^2.A')$, where $A' = A_{w_1} \sqcap A$, $\bar{A}' = A_{w_1} \sqcap \bar{A}$ and $\exists r^m.C$ is an abbreviation defined inductively by taking $\exists r^0.C = C$ and $\exists r^{m+1}.C = \exists r.\exists r^m.C$.

In addition, we require a generic way of describing frames using TBoxes. Given an R-frame $\mathfrak{R}$, let A_u be a fresh concept name for every $u \in \Delta^{\mathfrak{R}}$. Let $\mathcal{T}_S(\mathfrak{R})$ be the (possibly infinite) TBox with the following CIs:

– $A_u \sqsubseteq \exists r.A_v$, for $(u,v) \in r^{\mathfrak{R}}$;
– $A_u \sqcap A_v \sqsubseteq \bot$, for $u \neq v$;
– $A_u \sqcap \exists r.A_v \sqsubseteq \bot$, for $(u,v) \notin r^{\mathfrak{R}}$.

One can show that, for any R-frame $\mathfrak{R}$ with root w (from which all other points are reachable via roles) and any R-frame $\mathfrak{F}$, we have $\mathcal{T}_S(\mathfrak{R}) \not\models_{\mathfrak{F}} A_w \sqsubseteq \bot$ iff $\mathfrak{R}$ is a p-morphic image of a subframe of $\mathfrak{F}$.

Returning to Case 3.1., define $\mathcal{T}$ to be the TBox with the following CIs:

$$\mathcal{T}_S(\mathfrak{H}), \quad A_w \sqsubseteq \exists r^2.A', \quad A_w \sqsubseteq \exists r^2.\bar{A}'.$$

Then $\mathcal{T} \models_{\mathcal{K}} A_{w_0} \sqsubseteq \exists r^m.(A_w \sqcap C_1) \sqcup \exists r^m.(A_w \sqcap C_2)$, where m is the distance between w_0, w, but $\mathcal{T} \not\models_{\mathcal{K}} A_{w_0} \sqsubseteq \exists r^m.(A_w \sqcap C_1)$, $\mathcal{T} \not\models_{\mathcal{K}} A_{w_0} \sqsubseteq \exists r^m.(A_w \sqcap C_2)$.
Case 3.2: suppose that Case 3.1 does not hold. Denote by w_0 the root of $\mathfrak{H}$. Take a fresh concept name A and consider the TBox $\mathcal{T}$ with the following CIs:

– $\mathcal{T}_S(\mathfrak{H})$,
– $A \sqcap \exists r.A_v \sqsubseteq \bot$, for all v with $(w,v) \notin r^{\mathfrak{H}}$,
– $A_v \sqcap \exists r.A \sqsubseteq \bot$, for all v with both $(v,w) \notin r^{\mathfrak{H}}$ and $(v,w_1) \notin r^{\mathfrak{H}}$,
– $A \sqcap \exists r.A_{w'} \sqsubseteq \exists r.A_w$, for $(w,w') \in r^{\mathfrak{H}}$, $w' \neq w_1$,
– $A_w \sqsubseteq \exists r.(A \sqcap \exists r.A_{w_1})$,
– if w has an r-predecessor w_p, then $A_{w_p} \sqcap \exists r.A \sqsubseteq \exists r.(A_w \sqcap \exists r.(A \sqcap \exists r.A_w))$.

Then $\mathcal{T} \models_{\mathcal{K}} A_{w_0} \sqsubseteq \exists r^m.(A_w \sqcap \exists r.(A \sqcap \exists r.A_w)) \sqcup \exists r^m.(A_w \sqcap \exists r.(A_{w_1} \sqcap \exists r.A))$, but $\mathcal{T} \not\models_{\mathcal{K}} A_{w_0} \sqsubseteq B$ for either of the disjuncts B in the right-hand side.
Case 4. A case distinction similar to, but much more tedious than the previous ones shows that $\mathcal{K}$ is concept non-convex if the constraint $\mathcal{K}$ is not transitive. The case where $\mathcal{K}$ is a class of transitive frames has been considered in [8], and one can easily modify the proofs given there to show that all universal classes of transitive and reflexive frames, which are not TBox equivalent to **(sin)**, **(equ)** or the class of all transitive and reflexive frames, are concept non-convex.

Typically, in DL applications one role is not enough. Therefore, the question is whether the four universal constraints guaranteeing tractability for a single reflexive relation still ensure tractability if more than one role is considered. This is well known to be the case for transitivity and reflexivity, and this is trivially the case for the singleton frame. Equivalence relations behave not so well:

Theorem 7. *If $\mathcal{K}$ is a constraint consisting of two (or more) equivalence relations, then $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is* NP*-hard. In particular, tractability of $\mathcal{EL}_\bot$ subsumption is not preserved under fusions in the sense of [3].*

Proof sketch.[4] The proof is by reduction of SAT. Let φ be a formula in NNF with the variables $p_1, \dots, p_{2n}$, and let r_1, r_2 be equivalence relations. We use T_k, F_k for the truth-values of the variable p_k, and L_j as a marker for the level j in a 'tree.' We generate a full binary tree of depth $2n+1$, using the CIs

$$L_{2i} \sqsubseteq \exists r_1.(T_{2i+1} \sqcap L_{2i+1}) \sqcap \exists r_1.(F_{2i+1} \sqcap L_{2i+1}), \tag{2}$$
$$L_{2i+1} \sqsubseteq \exists r_2.(T_{2i+2} \sqcap L_{2i+2}) \sqcap \exists r_2.(F_{2i+2} \sqcap L_{2i+2}), \tag{3}$$

[4] Based on an idea suggested by Carsten Lutz.

for $i < n$. Then we propagate the truth-values T_k and F_k to the leaves using

$$L_{2j} \sqcap \exists r_2.(L_{2j-1} \sqcap Q_k) \sqsubseteq Q_k, \quad \text{for } 1 \leq j \leq n,\ 1 \leq k \leq 2j-1, \tag{4}$$

$$L_{2j+1} \sqcap \exists r_1.(L_{2j} \sqcap Q_k) \sqsubseteq Q_k, \quad \text{for } 1 \leq j < n,\ 1 \leq k \leq 2j, \tag{5}$$

for $Q_k = T_k, F_k$. Take a fresh X_ψ, for every subformula ψ of φ, and the CIs

$$X_{p_k} \equiv T_k, \quad X_{\neg p_k} \equiv F_k, \quad X_{\psi_1 \wedge \psi_2} \equiv X_{\psi_1} \sqcap X_{\psi_2}, \tag{6}$$

$$X_{\psi_1} \sqsubseteq X_{\psi_1 \vee \psi_2}, \quad X_{\psi_2} \sqsubseteq X_{\psi_1 \vee \psi_2}. \tag{7}$$

Let $\mathcal{T}$ be the TBox containing all the CIs (2)–(7), and $L_{2n} \sqcap X_\varphi \sqsubseteq \bot$. One can show that $\mathcal{T} \models_{\mathcal{K}} L_0 \sqsubseteq \bot$ iff φ is satisfiable.

6 $\mathcal{EL}$ and $\mathcal{EL}_\bot$

So far, we have considered $\mathcal{EL}_\bot$ rather than $\mathcal{EL}$. The main reason is that $\bot$ makes proofs more transparent. We now show that Theorems 4–7 above hold for $\mathcal{EL}$.

An R-frame $\mathfrak{F}'$ is called a *generated subframe* of an R-frame $\mathfrak{F}$ if it is a subframe of $\mathfrak{F}$ and, for all $u, v \in \Delta^{\mathfrak{F}}$ and $r \in \mathsf{R}$, if $(u, v) \in r^{\mathfrak{F}}$ and $u \in \Delta^{\mathfrak{F}'}$ then $v \in \Delta^{\mathfrak{F}'}$. Given $v \in \Delta^{\mathfrak{F}}$, the *subframe of* $\mathfrak{F}$ *generated by* v is the smallest generated subframe of $\mathfrak{F}$ containing v.

Theorem 8. *Let $\mathcal{K}$ be an R-constraint closed under generated subframes, for a finite R. Then $\mathcal{EL}_\bot$ subsumption over $\mathcal{K}$ is polynomially reducible to $\mathcal{EL}$ subsumption over $\mathcal{K}$, and, for any R-constraint $\mathcal{K}'$ closed under generated subframes, $\mathcal{K}'$ is TBox-equivalent to $\mathcal{K}$ in $\mathcal{EL}_\bot$ iff $\mathcal{K}'$ is TBox-equivalent to $\mathcal{K}$ in $\mathcal{EL}$.*

Proof. Let $\mathcal{T}$ and $C \sqsubseteq D$ in $\mathcal{EL}_\bot$ be given. We may assume that $\bot$ occurs in them only in the form $E \sqsubseteq \bot$, with E being an $\mathcal{EL}$-concept. Let B be a fresh concept name, and let $\mathcal{T}'$ and D' result from $\mathcal{T}$ and D, respectively, by replacing all $\bot$ with B. Set $\mathcal{T}'' = \mathcal{T}' \cup \{\exists\, r.B \sqsubseteq B \mid r \in \mathsf{R}\} \cup \{B \sqsubseteq D'\}$. We claim that $\mathcal{T} \models_{\mathcal{K}} C \sqsubseteq D$ iff $\mathcal{T}'' \models_{\mathcal{K}} C \sqsubseteq D'$. Clearly, if $\mathcal{T} \not\models_{\mathcal{K}} C \sqsubseteq D$, then $\mathcal{T}'' \not\models_{\mathcal{K}} C \sqsubseteq D'$: for if we have a witness model for $\mathcal{T} \not\models_{\mathcal{K}} C \sqsubseteq D$, then we can interpret B by the empty set to obtain a model of $\mathcal{T}''$ refuting $C \sqsubseteq D'$. Conversely, if $\mathcal{T}'' \not\models_{\mathcal{K}} C \sqsubseteq D'$, take an interpretation $\mathcal{I}$ based on a frame in $\mathcal{K}$ and $v \in \Delta^{\mathcal{I}}$ such that $\mathcal{I} \models \mathcal{T}''$ but $v \in C^{\mathcal{I}} \setminus (D')^{\mathcal{I}}$. Let $\mathfrak{F}$ be the subframe generated by v in the underlying frame of $\mathcal{I}$. Then $\mathfrak{F} \in \mathcal{K}$ and $B^{\mathcal{I}} \cap \Delta^{\mathfrak{F}} = \emptyset$. Hence $\mathcal{T} \not\models_{\mathfrak{F}} C \sqsubseteq D$, as required.

It follows from Theorem 8 that Theorems 6 and 7 hold for $\mathcal{EL}$ in place of $\mathcal{EL}_\bot$. Theorem 4 can be proved for $\mathcal{EL}$ as follows. Let $\mathcal{K}$ be a non-functional tabular constraint. Then the class $\mathcal{K}'$ of subframes of frames from $\mathcal{K}$ is still a non-functional tabular constraint and $\models_{\mathcal{K}'}$ is polynomially reducible to $\models_{\mathcal{K}}$, both for $\mathcal{EL}$ and $\mathcal{EL}_\bot$ (using relativisation). Thus, by Theorem 4 for $\mathcal{EL}_\bot$ and Theorem 8, the $\mathcal{EL}$ subsumption problem for $\mathcal{K}'$ is coNP-hard. Hence it is coNP-hard for $\mathcal{K}$. Theorem 5 can be proved similarly.

7 Open Problems and Conjectures

The main open problem in the area is the dichotomy question formulated in the introduction. If the answer to this question is positive, then the proof will probably require some new techniques and a great number of case distinctions.

We conjecture that a transparent dichotomy, possibly more involved than Theorem 6, can be obtained for arbitrary relational constraints on a single reflexive relation. Of course, an additional problem in this case is how to deal with non first-order constraints. A possible approach can be illustrated by the following result from [8]. Call a constraint *subframe* if it is closed under the formation of subframes. A *Noetherian* partial order is a reflexive and transitive relation without infinite ascending chains. Let $\mathcal{N}$ be the (non-elementary) class of all Noetherian partial orders. It is proved in [8] that $\mathcal{EL}_\perp$ subsumption over a subframe constraint $\mathcal{K} \subseteq \mathcal{N}$ is tractable iff $\mathcal{K}$ is TBox equivalent either to the single element frame or to $\mathcal{N}$.

When moving beyond the 'bounded' constraints of Theorems 4 and 5, it seems to be much harder to obtain general results for relations that can be non-reflexive than for the reflexive ones. For example, in contrast to the reflexive case, $\mathcal{EL}_\perp$ subsumption is now also in P for the constraints $\mathcal{K}_n$ consisting of (irreflexive) trees of depth $\leq n$. Thus, there are infinitely many transitive classes with a single relation for which $\mathcal{EL}_\perp$ subsumption is tractable.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI. pp. 364–369 (2005)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Proc. OWLED (2008)
3. Baader, F., Lutz, C., Sturm, H., Wolter, F.: Fusions of description logics and abstract description systems. J. Artif. Intell. Res. (JAIR) 16, 1–58 (2002)
4. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co. (1979)
5. Hemaspaandra, E., Schnoor, H.: On the complexity of elementary modal logics. In: Proc. of STACS, pp. 349–360 (2008)
6. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. KR, pp. 57–67 (2006)
7. Kazakov, Y.: An extension of complex role inclusion axioms in the description logic $\mathcal{SROIQ}$. In: Proc. of IJCAR. pp. 472–486 (2010)
8. Kurucz, A., Wolter, F., Zakharyaschev, M.: Islands of tractability for relational constraints: towards dichotomy results for the description logic $\mathcal{EL}$. In: Advances in Modal Logic, Vol.8. pp. 271–291 (2010)
9. Sofronie-Stokkermans, V.: Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In: Advances in Modal Logic, Vol.7. pp. 315–339 (2008)
10. Tarski, A.: Contributions to the theory of models I, II. Indag. Mathematicae 16, 572–588 (1954)

On the evolution of the instance level of *DL-Lite* knowledge bases

Maurizio Lenzerini, Domenico Fabio Savo

Dipartimento di Informatica e Sistemistica Antonio Ruberti
SAPIENZA Università di Roma
lastname@dis.uniroma1.it

Abstract. Recent papers address the issue of updating the instance level of knowledge bases expressed in Description Logic following a model-based approach. One of the outcomes of these papers is that the result of updating a knowledge base $\mathcal{K}$ is generally not expressible in the Description Logic used to express $\mathcal{K}$. In this paper we introduce a formula-based approach to this problem, by revisiting some research work on formula-based updates developed in the '80s, in particular the WIDTIO (When In Doubt, Throw It Out) approach. We show that our operator enjoys desirable properties, including that both insertions and deletions according to such operator can be expressed in the DL used for the original KB. Also, we present polynomial time algorithms for the evolution of the instance level knowledge bases expressed in *DL-Lite*$_{A,id}$, which the most expressive Description Logics of the *DL-Lite* family.

1 Introduction

Description Logics (DLs) are logics for expressing knowledge bases (KBs) constituted by two components, namely, the TBox, asserting general properties of concepts and roles (binary relations), and the ABox, which is a set of assertions about individuals that are instances of concepts and roles. It is widely accepted that such logics are well-suited for expressing ontologies, with the TBox capturing the intensional knowledge about the domain of interest, and the ABox expressing the knowledge about the instance level of the predicates defined in the TBox. Following this idea, several Knowledge Representation Systems, called DL systems, have been recently built, providing methods and tools for managing ontologies expressed in DLs [1]. Notice that numerous DLs have been studied in the last decades, with the goal of analyzing the impact of the expressive power of the DL language to the complexity of reasoning. Consequently, each DL system is tailored towards managing KB expressed in a specific DL.

By referring to the so-called *functional view of knowledge representation* [11], DL systems should be able to perform two kinds of operations, called ASK and TELL. ASK operations, such as subsumption checking, or query answering, are

[1] http://www.cs.man.ac.uk/ sattler/reasoners.html

used to extract information from the KB, whereas TELL operations aim at changing the KB according to new knowledge acquired over the domain. In other words, TELL operations should be able to cope with the *evolution of the KB*.

There are two types of evolution operators, corresponding to inserting, and deleting chunks of knowledge, respectively. In the case of insertion, the aim is to incorporate new knowledge into the KB, and the corresponding operator should be defined in such a way to compute a consistent KB that supports the new knowledge. In the case of deletion, the aim is to come up with a consistent KB where the retracted knowledge is not valid. In both cases, the crucial aspect to take into account is that evolving a consistent knowledge base should not introduce inconsistencies.

While ASK operations have been investigated in detail by the DL community, existing DL reasoners do not provide explicit services for KB evolution. Nevertheless, many recent papers demonstrate that the interest towards a well-defined approach to KB evolution is growing significantly [9, 12, 7, 13, 6]. Following the tradition of the work on knowledge revision and update [10], all the above papers advocate some minimality criterion in the changes of the KB that must be undertaken to realize the evolution operations. In other words, the need is commonly perceived of keeping the distance between the original KB and the KB resulting from the application of an evolution operator minimal. There are two main approaches to define such a distance, called *model-based* and *formula-based*, respectively. In the model-based approaches, the result of an evolution operation applied to the KB $\mathcal{K}$ is defined in terms of a set of models, with the idea that such a set should be as close as possible to the models of $\mathcal{K}$. One basic problem with this approach is to characterize the language needed to express the KB that exactly captures the resulting set of models. Conversely, in the formula-based approaches, the result is explicitly defined in terms of a formula, by resorting to some minimality criterion with respect to the formula expressing $\mathcal{K}$. Here, the basic problem is that the formula constituting the result of an evolution operation is not unique in general.

In this paper, we study the problem of DL KB evolution, by focusing our attention to scenarios characterized by the following elements:

(1) We consider the case where the evolution affects only the instance level of the KB, i.e., the ABox. In other words, we enforce the condition that the KB resulting from the application of the evolution operators has the same TBox as the original KB (similarly to [12, 7]).

(2) We aim at a situation where the KB resulting from the evolution can be expressed in the same DL as the original KB. This is coherent with our goal of providing the foundations for equipping DL systems with evolution operators: indeed, if a DL system S is able to manage KBs expressed in a DL $\mathcal{L}$, the result of evolving such KBs should be expressible in $\mathcal{L}$.

(3) The KBs resulting from the application of an evolution operator on two logically equivalent KBs should be mutually equivalent. In other words, we want the result to be independent of the syntactic form of the original KB.

Assumption (1), although limiting the generality of our approach, captures several interesting scenarios, including *ontology-based data management*, where the DL KB is used as a logic-based interface to existing data sources.

As for item (2), we note that virtually all model-based approaches suffer from the expressibility problem. This has been reported in many recent papers, including [12, 7, 6], for various DLs. For this reason, we adopt a formula-based approach, inspired in particular by the work developed in [8] for updating logical theories. As in [8], we consider both insertions and deletions. However, we differ from [8] for an important aspect. We already noted that the formula constituting the result of an evolution operation is not unique in general. While [8] essentially proposes to keep the whole set of such formulas, we take a radical approach, and consider their intersection as the result of the evolution. In other words, we follow the *When In Doubt Throw It Out* (WIDTIO) [14] principle.

Finally, to deal with item (3), we sanction that the notion of distance between KBs refers to the closure of the ABox of a KB, rather than to the ABox itself. The closure of an ABox $\mathcal{A}$ with respect to an TBox $\mathcal{T}$ is defined as the set of all ABox assertions that logically follows from $\mathcal{T}$ and $\mathcal{A}$. By basing the definition of distance on the closure of ABoxes, we achieve the goal of making the result of our operators independent of the form of the original KB.

After a brief introduction to DLs (Section 2), we provide the definition of our evolution operators in Section 3. The remaining sections are devoted to illustrating algorithms for deletion (Section 4), and insertion (Section 5) for KBs expressed in the DL $DL\text{-}Lite_{A,id}$, which is the most expressive logic in the *DL-Lite* family [4]. The *DL-Lite* family[2] has been specifically designed to keep all reasoning tasks polynomially tractable, and we show that this property still holds for the evolution operators proposed in this paper.

2 Preliminaries

Let $\mathcal{S}$ be a signature of symbols for individual (object and value) constants, and atomic elements, i.e., concepts, value-domains, attributes, and roles. If $\mathcal{L}$ is a DL, then an $\mathcal{L}$-KB $\mathcal{K}$ over $\mathcal{S}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$, called *TBox*, is a finite set of intensional assertions over $\mathcal{S}$ expressed in $\mathcal{L}$, and $\mathcal{A}$, called *ABox*, is a finite set of instance assertions, i.e, assertions on individuals, over $\mathcal{S}$ expressed in $\mathcal{L}$. Different DLs allow for different kinds of concept, attribute, and role expressions, and different kinds of TBox and ABox assertions over such expressions. In this paper we assume that ABox assertions are always *atomic*, i.e., they correspond to ground atoms, and therefore we omit to refer to $\mathcal{L}$ when we talk about ABox assertions.

The semantics of a DL KB is given in terms of interpretations. An interpretation is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies all assertions in $\mathcal{T} \cup \mathcal{A}$, where the notion of satisfaction depends on the constructs allowed by the specific DL in which $\mathcal{K}$ is expressed. We denote the set of models of $\mathcal{K}$ with $Mod(\mathcal{K})$.

[2] Not to be confused with the set of DLs studied in [2], which form the $DL\text{-}Lite_{bool}$ family.

Let $\mathcal{T}$ be a TBox in $\mathcal{L}$, and let $\mathcal{A}$ be an ABox. We say that $\mathcal{A}$ is *$\mathcal{T}$-consistent* if $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, i.e. if $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) \neq \emptyset$, $\mathcal{T}$-inconsistent otherwise. The *$\mathcal{T}$-closure* of $\mathcal{A}$ with respect to $\mathcal{T}$, denoted $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$, is the set of all atomic ABox assertion that are formed with individuals in $\mathcal{A}$, and are logically implied by $\langle \mathcal{T}, \mathcal{A} \rangle$. Note that if $\langle \mathcal{T}, \mathcal{A} \rangle$ is an $\mathcal{L}$-KB, then $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$ is an $\mathcal{L}$-KB as well, and is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle$, i.e., $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod(\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle)$. $\mathcal{A}$ is said to be *$\mathcal{T}$-closed* if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) = \mathcal{A}$. Finally, for an ABox assertion γ_1, we denote by $\mathsf{Subsumee}_{\langle \mathcal{T}, \mathcal{A} \rangle}(\gamma_1)$ the set of atoms $\gamma_2 \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\langle \mathcal{T}, \mathcal{A} \rangle | = \gamma_2 \supset \gamma_1$.

The *DL-Lite* family [4] is a family of low complexity DLs particularly suited for dealing with KBs with very large ABoxes, and forms the basis of OWL 2 QL, one of the profile of OWL 2, the official ontology specification language of the World-Wide-Web Consortium (W3C)[3].

We now present the DL *DL-Lite*$_{A,id}$, which is the most expressive logic in the family. Expressions in *DL-Lite*$_{A,id}$ are formed according to the following syntax:

$$\begin{array}{lll} B \longrightarrow A \mid \exists Q \mid \delta(U) & E \longrightarrow \rho(U) & C \longrightarrow B \mid \neg B \\ Q \longrightarrow P \mid P^- & V \longrightarrow U \mid \neg U & R \longrightarrow Q \mid \neg Q \\ T \longrightarrow \top_D \mid T_1 \mid \cdots \mid T_n & & \end{array}$$

where A, P, and U are symbols in $\mathcal{S}$ denoting respectively an atomic concept name, an atomic role name and an attribute name, $T_1, \ldots, T_n$ are all the value-domains allowed in the logic (those corresponding to the data types adopted by Resource Description Framework (RDF)[4]), $\top_D$ denotes the union of all domain values, P^- denotes the inverse of P, $\exists Q$ denotes the objects related to by the role Q, $\neg$ denotes negation, $\delta(U)$ denotes the *domain* of U, i.e., the set of objects that U relates to values, and $\rho(U)$ denotes the *range* of U, i.e., the set of values related to objects by U.

A *DL-Lite*$_{A,id}$ TBox $\mathcal{T}$ contains intensional assertions of three types, namely inclusion assertions, functionality assertions, and identification assertions [5] (IDs). More precisely, *DL-Lite*$_{A,id}$ assertions are of the form:

$$\begin{array}{llll} B \sqsubseteq C & (\textit{concept inclusion}) & E \sqsubseteq T & (\textit{value-domain inclusion}) \\ Q \sqsubseteq R & (\textit{role inclusion}) & (\mathsf{funct}\ U) & (\textit{attribute functionality}) \\ (id\ B \pi\ _1, ..., \pi_n) & (\textit{identification}) & & \end{array}$$

In the identification assertions, π denotes a *path*, which is an expression built according to the following syntax rule:

$$\pi \longrightarrow S \mid B? \mid \pi_1 \circ \pi_2$$

where S denotes an atomic role, the inverse of an atomic role, or an atomic attribute, $\pi_1 \circ \pi_2$ denotes the composition of the paths π_1 and π_2, and $B?$, called *test relation*, represents the identity relation on instances of the concept B. In our logic, identification assertions are *local*, i.e., at least one $\pi_i \in \{\pi_1, ..., \pi_n\}$ has length 1, i.e., it is an atomic role, the inverse of an atomic role, or an atomic attribute. In what follows, we only refer to IDs which are local.

[3] `http://www.w3.org/TR/2008/WD-owl2-profiles-20081008/`

[4] `http://www.w3.org/RDF/`

The set of positive (resp., negative) inclusions in $\mathcal{T}$ will be denoted by $\mathcal{T}^+$ (resp., $\mathcal{T}^-$), and the set of identification assertions in $\mathcal{T}$ will be denoted by $\mathcal{T}_{id}$.

A concept inclusion assertion expresses that a (basic) concept B is subsumed by a (general) concept C. Analogously for the other types of inclusion assertions. Inclusion assertions that do not contain (resp. contain) the symbols '$\neg$' in the right-hand side are called *positive inclusions* (resp. *negative inclusions*). Attribute functionality assertions are used to impose that attributes are actually functions from objects to domain values. An ID $(id\ B\pi\ _1, ..., \pi_n)$ asserts that for any two different instances a,b of B, there is at least one π_i such that a and b differ in the set of their π_i-fillers. Note that IDs can be used to assert functionality of roles. Specifically, the assertion $(id\ \exists Q^-\ Q^-)$ imposes that Q is functional.

Finally, a TBox *DL-Lite*$_{A,id}$ $\mathcal{T}$ satisfies the following condition: every role or attribute that occurs (in either direct or inverse direction) in a path of an ID $\alpha \in \mathcal{T}_{id}$ or in a functional assertion, is not specialized in $\mathcal{T}'$, i.e., it does not appear in the right-hand side of assertions of the form $Q \sqsubseteq Q'$ or $U \sqsubseteq U'$.

A *DL-Lite*$_{A,id}$ ABox $\mathcal{A}$ is a finite set of assertions of the form $A(a)$, $P(a,b)$, and $U(a,v)$, where A, P, and U are as above, a and b are object constants in $\mathcal{S}$, and v is a value constant in $\mathcal{S}$.

Example 1. We consider a portion of the Formula One domain. We know that official drivers (OD) and test drivers (TD) are both team members (TM), and official drivers are not test drivers. Every team member is a member of (mf) a exactly one team (FT), and every team has at most one official driver. Finally, no race director (RD) is a member of a team. We also know that s is the official driver of team t_1, that b is a test driver, and that p is a team member. The corresponding *DL-Lite*$_{A,id}$-KB $\mathcal{K}$ is:

$\mathcal{T}$: $OD \sqsubseteq TM\ \ TD \sqsubseteq TM\ \ OD \sqsubseteq \neg TD\ \ RD \sqsubseteq \neg TM\ \ TM \sqsubseteq \exists mf$
$TM \sqsubseteq \neg FT\ \ \exists mf \sqsubseteq TM\ \ \exists mf^- \sqsubseteq FT\ \ (id\ OD\ mf)\ \ (id\ FT\ mf^-)$
$\mathcal{A}$: $OD(s)\quad mf(s,t_1)\quad TD(b)\quad TM(p)$ $\square$

We conclude this section with a brief discussione on the complexity of reasoning about a *DL-Lite*$_{A,id}$-KB $\langle \mathcal{T}, \mathcal{A} \rangle$. Satisfiability can be checked in polynomial time with respect to $|\mathcal{T} \setminus \mathcal{T}_{id}|$ and $|\mathcal{A}|$, and in NP with respect to $|\mathcal{T}_{id}|$. Moreover, if $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, then answering a query q posed to $\langle \mathcal{T}, \mathcal{A} \rangle$ can be done in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$, and in NP with respect to $|q|$. Finally, $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ can be computed in quadratic time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$.

3 WIDTIO approach to KB evolution in DLs

In this section we first present our semantics for the evolution of DL knowledge bases at the instance level, and then we provide a comparison between our operator and other work in the literature.

In the rest of this section, $\mathcal{L}$ is a DL, and $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a satisfiable $\mathcal{L}$-KB. In other words, we do not consider the evolution of unsatisfiable KBs. In addition, F is a finite set of atomic ABox assertions in $\mathcal{L}$.

The following definition specifies when a set of ABox assertions "realizes" the insertion or deletion of a set of ABox assertions with respect to $\mathcal{K}$.

Definition 1. *Let $\mathcal{A}'$ be an ABox. Then, $\mathcal{A}'$ accomplishes the insertion of F into $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{T}$-consistent, and $\langle \mathcal{T}, \mathcal{A}' \rangle \models F$ (i.e., $F \subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$). Similarly, $\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{A}'$ is $\mathcal{T}$-consistent, and $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models F$ (i.e., $F \not\subseteq \mathsf{cl}_{\mathcal{T}}(\mathcal{A}')$).*

Obviously, we are interested in KBs which accomplish the evolution of a KB with a *minimal change*. In order to formalize the notion of *minimal change*, we first need to provide some definitions.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two ABoxes. Then, we say that $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \subset \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Similarly, we say that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) = \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2)$. Finally, we say that $\mathcal{A}_1$ has fewer insertions than $\mathcal{A}_2$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}_1) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \subset \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_2) \setminus \mathsf{cl}_{\mathcal{T}}(\mathcal{A})$.

Definition 2. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two ABoxes. Then, $\mathcal{A}_1$ has* fewer changes *than $\mathcal{A}_2$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{A}_1$ has fewer deletions than $\mathcal{A}_2$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$, or $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same deletions with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$, and $\mathcal{A}_!$ has fewer insertions than $\mathcal{A}_2$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$.*

Now that we have defined the relation of *fewer changes* between two KBs w.r.t. another one, we can define the notion of a KB which accomplishes the insertion (resp. deletion) of a set of facts into (resp. from) another KB minimally.

Definition 3. *Let $\mathcal{A}'$ be an ABox. Then $\mathcal{A}'$ accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally if $\mathcal{A}'$ accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{T}, \mathcal{A} \rangle$, and there is no $\mathcal{A}''$ that accomplishes the insertion (deletion) of F into (from) $\langle \mathcal{T}, \mathcal{A} \rangle$, and has fewer changes than $\mathcal{A}'$ with respect to $\langle \mathcal{T}, \mathcal{A} \rangle$.*

With these notions in place, we can now define our evolution operator.

Definition 4. *Let $\mathcal{U} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be the set of all ABoxes accomplishing the insertion (deletion) of F into (from) $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and let $\mathcal{A}'$ be an ABox. Then, $\langle \mathcal{T}, \mathcal{A}' \rangle$ is the result of changing $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion (deletion) of F if* (1) *$\mathcal{U}$ is empty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \rangle$, or* (2) *$\mathcal{U}$ is nonempty, and $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}') \rangle = \langle \mathcal{T}, \bigcap_{1 \leq i \leq n} \mathsf{cl}_{\mathcal{T}}(\mathcal{A}_i) \rangle$.*

It is immediate to verify that, up to logical equivalence, the result of changing $\langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion or the deletion of F is unique. In the rest of this paper, the result of changing $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the insertion (resp. deletion) of F according to our semantics will be denoted by $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$ (resp. $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$). Notice that, by definition of our operator, in the case where F is $\mathcal{T}$-inconsistent, the result of changing $\langle \mathcal{T}, \mathcal{A} \rangle$ with both the insertion and the deletion of F is logically equivalent to $\langle \mathcal{T}, \mathcal{A} \rangle$ itself.

Example 2. Consider the *DL-Lite*$_{A,id}$ KB $\mathcal{K}$ of the Example 1, and suppose that p becomes now a race director, and b becomes the new official driver of the team t_1. To reflect this new information, we change $\mathcal{K}$ with the insertion of $F_1 = \{RD(p), OD(b), mf(b, t_1)\}$. Since the TBox implies that a race director cannot be a team member, $RD(p)$ contradicts $TM(p)$. Also, since every team has at most one official driver, $OD(b)$ and $mf(b, t_1)$ contradict $mf(s, t)$. According to Definition 3, the KBs accomplishing the insertion of F_1 into $\mathcal{K}$ minimally are:

$\mathcal{K}_1 = \langle \mathcal{T}, \{RD(p), OD(b), mf(b, t_1), TM(s), mf(s, t_1)\}\rangle$
$\mathcal{K}_2 = \langle \mathcal{T}, \{RD(p), OD(b), mf(b, t_1), TM(s), OD(s)\}\rangle$

Thus, $\mathcal{K} \oplus_{\cap}^{\mathcal{T}} F_1$ is:

$\mathcal{K}_3 = \langle \mathcal{T}, \{RD(p), OD(b), mf(b, t_1), TM(s)\}\rangle$.

Now, suppose that we do not know anymore whether b is a member of t_1, and, even more, whether b is a team member at all. Then, we change $\mathcal{K}_3$ with the deletion of $F_2 = \{TM(b), mf(b, t_1)\}$, thus obtaining

$\mathcal{K}_3 \ominus_{\cap}^{\mathcal{T}} F_2 = \langle \mathcal{T}, \{RD(p), TM(s), OD(b)\}\rangle$. $\square$

The following theorem is an adaptation to our setting of two results reported in [8], and will be used in the next two sections.

Theorem 1. *Let $\mathcal{A}'$ be an ABox. Then*

1. *$\mathcal{A}'$ accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally if and only if $cl_{\mathcal{T}}(\mathcal{A}')$ is a maximal $\mathcal{T}$-closed subset of $cl_{\mathcal{T}}(\mathcal{A})$ such that $F \not\subseteq cl_{\mathcal{T}}(\mathcal{A}')$.*
2. *$\mathcal{A}'$ accomplishes the insertion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally if and only if $cl_{\mathcal{T}}(\mathcal{A}') = \mathcal{A}'' \cup cl_{\mathcal{T}}(F)$, where $\mathcal{A}''$ is a maximal $\mathcal{T}$-closed subset of $cl_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent.*

We end this section with a brief discussion on related work. We mentioned in the introduction several model-based approaches to DL KB evolution, and noticed that they all suffer from the expressibility problem. This problem is also shared by [13], that uses *features* instead of models, and proposes the notion of approximation to cope with the expressibility problem, similarly to [7].

Related to our proposal are several formula-based approaches presented in the literature. Perhaps, the closest approach to the one proposed in this paper is that reported in [6], where formula-based evolution (actually, insertion) of *DL-Lite* KBs is studied. The main difference with our work is that we base our semantics on the WIDTIO principles, and therefore we compute the intersection of all KBs accomplishing the change minimally. Conversely, in the *bold* semantics discussed in [6], the result of the change is chosen non-deterministically among the KBs accomplishing the change minimally. Another difference is that while [6] addresses the issue of evolution of both the TBox and the ABox, we only deal with the case of fixed TBox (in the terminology of [6], this corresponds to keep the TBox *protected*). It is interesting to observe that the specific DL considered in [6] is *DL-Lite*$_{\mathcal{FR}}$, and for this logic, exactly one KB accomplishes the insertion of a set of ABox assertions minimally. It follows that for instance-level insertion, their bold semantics coincides with ours. On the other hand, the presence of identification assertions in *DL-Lite*$_{A,id}$ changes the picture considerably, since with such assertions in the TBox, many KBs may exist accomplishing the insertion minimally. In this case, the two approaches are indeed different. Finally, [6] proposes a variant of the bold semantics, called *careful semantics*, for instance-level insertion in $\mathit{DL\text{-}Lite}_{\mathcal{FR}}$. Intuitively, such a semantics aims at disregarding knowledge that is entailed neither by the original KB, nor by the set of newly asserted facts. Although such principle is interesting, we believe that the careful semantics is too drastic, as it tends to eliminate too much information from the original KB.

Finally, we point out that, to our knowledge, the evolution operator presented in this work is the first tractable evolution operator based on the WIDTIO principle.

4 Deletion in $\boldsymbol{\mathit{DL\text{-}Lite}_{A,id}}$

We study deletion under the assumption that the DL language $\mathcal{L}$ is $\mathit{DL\text{-}Lite}_{A,id}$. Thus, in this section, we implicitly refer to a $\mathit{DL\text{-}Lite}_{A,id}$-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and we address the problem of changing $\mathcal{K}$ with the deletion of a finite set F of ABox assertions. We assume that both $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, F \rangle$ are satisfiable.

We first consider the case where the set F is constituted by just one assertion f. By exploiting Theorem 1, it is easy to conclude that there is exactly one KB accomplishing the deletion of $\{f\}$ from a given KB.

Theorem 2. *Let f be an ABox assertion. Up to logical equivalence, there is exactly one KB of the form $\langle \mathcal{T}, \mathcal{A}' \rangle$ that accomplishes the deletion of $\{f\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, and such KB can be computed in polynomial time with respect to $|\mathcal{T}|$ and $|\mathcal{A}|$.*

Let us now consider the case of arbitrary F, i.e., the case where $F = \{f_1, \ldots, f_m\}$, for $m \geq 0$. Suppose that, for every $1 \leq i \leq m$, $\mathcal{A}_i$ accomplishes the deletion of $\{f_i\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally. One might wonder whether the set $\Gamma_1 = \{\langle \mathcal{T}, \mathcal{A}_j \rangle \mid A_j$ accomplishes the deletion of F minimally from $\langle \mathcal{T}, \mathcal{A} \rangle\}$ coincides (modulo logical equivalence) with $\Gamma_2 = \{\langle \mathcal{T}, \mathcal{A}_1 \rangle, \ldots \langle \mathcal{T}, \mathcal{A}_m \rangle\}$. The next theorem tells us that one direction is indeed valid: for each KB $\mathcal{K}_1 \in \Gamma_1$ there exists a KB $\mathcal{K}_2 \in \Gamma_2$ such that $\mathit{Mod}(\mathcal{K}_1) = \mathit{Mod}(\mathcal{K}_2)$.

Theorem 3. *If $\langle \mathcal{T}, \mathcal{A}' \rangle$ accomplishes the deletion of $\{f_1, \ldots, f_m\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally, then there exists $i \in \{1..m\}$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ accomplishes the deletion of $\{f_i\}$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ minimally.*

However, the following example shows that the other direction does not hold: there may exist a $\mathcal{K}_2 \in \Gamma_2$ that is not logically equivalent to any $\mathcal{K}_1 \in \Gamma_1$.

Example 3. Let $\mathcal{T} = \{B \sqsubseteq C, C \sqsubseteq D, E \sqsubseteq D\}$, $\mathcal{A} = \{B(a), E(a)\}$, and $F = \{C(a), D(a)\}$. It is easy to see that the deletion of $D(a)$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ is accomplished minimally by $\langle \mathcal{T}, \emptyset \rangle$, while the deletion of $C(a)$ from $\langle \mathcal{T}, \mathcal{A} \rangle$ is accomplished minimally by $\langle \mathcal{T}, \{E(a)\} \rangle$. Therefore, in this case, we have $\Gamma_2 = \{\langle \mathcal{T}, \emptyset \rangle, \langle \mathcal{T}, \{E(a)\} \rangle\}$. Also, one can verify that $\langle \mathcal{T}, \{E(a)\} \rangle$ is the only (up to logical equivalence) KB accomplishing the deletion of F minimally, i.e., $\Gamma_1 = \{\langle \mathcal{T}, \{E(a)\}\rangle\}$. Thus, there is a KB in Γ_2, namely $\langle \mathcal{T}, \emptyset\rangle$, that is not logically equivalent to any KB in Γ_1. $\square$

The next theorem characterizes when a given $\langle \mathcal{T}, \mathcal{A}_i\rangle \in \Gamma_2$ accomplishes the deletion of F minimally.

Theorem 4. *Let $F = \{f_1, \ldots, f_m\}$, and, for every $1 \leq i \leq m$, let $\langle \mathcal{T}, \mathcal{A}_i\rangle$ accomplish the deletion of $\{f_i\}$ from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally. Then, $\langle \mathcal{T}, \mathcal{A}_j\rangle$, where $j \in \{1..m\}$, accomplishes the deletion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally if and only if there is no $h \in \{1..m\}$ such that $h \neq j$, and $\langle \mathcal{T}, \{f_h\}\rangle \models f_j$.*

By exploiting Theorems 2, 3, and 4, we can directly prove that $\mathcal{K} \ominus^{\mathcal{T}}_{\cap} F$ can be computed by the algorithm ComputeDeletion below. It is easy to see that the time complexity of the algorithm is $O(|\mathcal{T}|^2 \times |F|^2 + |\mathcal{A}|^2)$.

```
Input: a satisfiable $DL\text{-}Lite_{A,id}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, a finite set of ABox assertions
        $F$ such that $\langle \mathcal{T}, F\rangle$ is satisfiable
Output: a $DL\text{-}Lite_{A,id}$ KB
begin
    $F' \leftarrow F$;
    foreach $f_i \in F'$ and $f_j \in F$ such that $i \neq j$ do
        if $\langle \mathcal{T}, \{f_j\}\rangle \models f_i$ then $F' \leftarrow F' \setminus \{f_i\}$;
    return $\langle \mathcal{T}, \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \{\alpha \in \mathsf{Subsumee}_{\mathcal{K}}(f) \mid f \in F'\}\rangle$;
end
```

Algorithm 1: *ComputeDeletion*$(\langle \mathcal{T}, \mathcal{A}\rangle, F)$

Theorem 5. *ComputeDeletion$(\langle \mathcal{T}, \mathcal{A}\rangle, F)$ terminates, and computes $\langle \mathcal{T}, \mathcal{A}\rangle \ominus^{\mathcal{T}}_{\cap} F$ in polynomial time with respect to $|\mathcal{T}|$, $|\mathcal{A}|$ and $|F|$.*

5 Insertion in *DL-Lite$_{A,id}$*

In this section, we refer to a *DL-Lite$_{A,id}$*-KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, and address the problem of changing $\mathcal{K}$ with the insertion of a finite set F of ABox assertions. As in the previous section, we assume that both $\langle \mathcal{T}, \mathcal{A}\rangle$ and $\langle \mathcal{T}, F\rangle$ are satisfiable.

Theorem 1 tells us that, in principle, we can compute the KB resulting from the insertion of F into $\langle \mathcal{T}, \mathcal{A}\rangle$ by building all maximal subsets of $\mathcal{A}$ which are $\mathcal{T}$-consistent with F, and then computing their intersection. The main problem to be faced with this method is that, depending on the DL used, there can be an exponential number of maximal subsets $\mathcal{A}'$ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}' \cup \{f\}$ is $\mathcal{T}$-consistent[5].In particular, in *DL-Lite$_{A,id}$*, building all maximal subsets of $\mathcal{A}$ which are $\mathcal{T}$-consistent with F, and then computing their intersection is computationally costly. Fortunately, we show in the following that $\mathcal{K} \oplus^{\mathcal{T}}_{\cap} F$ can be computed without computing all maximal consistent subsets of $\mathcal{A}$ with F.

[5] Note that this cannot happen in those DLs of the *DL-Lite* family which do not admit the use of identification assertions (such as the DL studied in [6]).

To describe our method, we need some preliminary notions. A set V of ABox assertions is called a *$\mathcal{T}$-violation set for $t \in \mathcal{T} \setminus \mathcal{T}^+$* if $\langle \mathcal{T}^+ \cup \{t\}, V\rangle$ is unsatisfiable, while for every proper subset V' of V, $\langle \mathcal{T}^+ \cup \{t\}, V'\rangle$ is satisfiable. Any set V of ABox assertions that is a $\mathcal{T}$-violation set for a $t \in \mathcal{T} \setminus \mathcal{T}^+$ is simply called a *$\mathcal{T}$-violation set.*

We know from Theorem 1 that the ABox $\mathcal{A}'$ accomplishes the insertion of F from $\langle \mathcal{T}, \mathcal{A}\rangle$ minimally if and only if $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}') = \mathcal{A}'' \cup \mathsf{cl}_{\mathcal{T}}(F)$, where $\mathcal{A}''$ is a maximal $\mathcal{T}$-closed subset of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\mathcal{A}'' \cup F$ is $\mathcal{T}$-consistent. Since we must compute the intersection of all such ABoxes $\mathcal{A}'$, it is sufficient to compute those assertions in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ that are not in the intersection, and remove them from $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{F})$. All the assertions in $\mathsf{cl}_{\mathcal{T}}(F)$ are obviously in the intersection of the ABoxes $\mathcal{A}'$. As for the ABox assertions in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$, it is easy to see that one such assertion α is not in the intersection of the ABoxes $\mathcal{A}'$ if and only if there exists a maximal subset Σ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{T}$-consistent, and Σ does not contain α.

Taking into account the above observation, the next theorem is the key to our solution.

Theorem 6. *Let α be an assertion in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$. There exists a maximal subset Σ of $\mathsf{cl}_{\mathcal{T}}(\mathcal{A})$ such that $\Sigma \cup F$ is $\mathcal{T}$-consistent, and Σ does not contain α if and only if there is a $\mathcal{T}$-violation set V in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ such that $\alpha \in V$, and $F \cup (V \setminus \{\alpha\})$ is $\mathcal{T}$-consistent.*

Theorem 6 suggests immediately the algorithm ComputeInsertion below for computing $\mathcal{K} \oplus_{\cap}^{\mathcal{T}} F$.

```
Input: a satisfiable $DL\text{-}Lite_{A,id}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, a finite set of ABox assertions
        $F$ such that $\langle \mathcal{T}, F\rangle$ is satisfiable
Output: a $DL\text{-}Lite_{A,id}$ KB.
begin
    $F' = \emptyset$;
    foreach $\alpha \in \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus \mathsf{cl}_{\mathcal{T}}(F)$ do
        if $\exists$ a $\mathcal{T}$-violation set $V$ in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$ s.t. $\alpha \in V$ and
            $\langle \mathcal{T}, F \cup (V \setminus \{\alpha\})\rangle$ is satisfiable
        then $F' \leftarrow F' \cup \{\alpha\}$;
    return $\langle \mathcal{T}, F \cup \mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \setminus F'\rangle$;
end
```

Algorithm 2: *ComputeInsertion*($\langle \mathcal{T}, \mathcal{A}\rangle, \mathcal{F}$)

Algorithm ComputeInsertion requires to compute all $\mathcal{T}$-violation sets in $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. It can be shown that this can be done by computing the results of suitable conjunctive queries posed to $\mathsf{cl}_{\mathcal{T}}(\mathcal{A}) \cup \mathsf{cl}_{\mathcal{T}}(F)$. Such queries are built out of the negative inclusion assertions and the identification assertions $\mathcal{T}_{id}$ in $\mathcal{T}$, and essentially look for tuples that satisfy the negation of such assertions. From this observation, one can derive the following theorem.

Theorem 7. *ComputeInsertion($\langle \mathcal{T}, \mathcal{A}\rangle, \mathcal{F}$) terminates, and computes $\langle \mathcal{T}, \mathcal{A}\rangle \oplus_{\cap}^{\mathcal{T}} F$ in polynomial time with respect to $|\mathcal{T} \setminus \mathcal{T}_{id}|$, $|\mathcal{A}|$, and $|F|$, and in NP with respect to $|\mathcal{T}_{id}|$.*

6 Conclusions

We plan to continue our work along several directions. First, we aim at extending our approach to the problem of evolution of the whole KB, as opposed to the ABox only. Also, we will add the notion of protected part to our approach, to model situations where one wants to prevent changes on specific parts of the KB when applying insertions or deletions. Finally, we aim at studying the case where the KB contains other kinds of constraints, so as to capture the scenario where updates are expressed on a conceptual model used as a global schema in a data integration system [3]. In this context, one of the major challenges is to deal with the problem of pushing the updates to the data sources.

References

1. Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS'99*, pages 68–79, 1999.
2. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini. Accessing Data Integration Systems through Conceptual Schemas. *International Conference on Conceptual Modeling*, LNCS 2224, 270–284, ER 2001.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-based identification constraints in description logics. In *Proc. of KR 2008*, pages 231–241, 2008.
6. Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Evolution of *DL-Lite* knowledge bases. In *Proc. of ISWC 2010*, 2010.
7. Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On instance-level update and erasure in description logic ontologies. *J. of Logic and Computation, Special Issue on Ontology Dynamics*, 19(5):745–770, 2009.
8. Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proc. of PODS'83*, pages 352–365, 1983.
9. Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.
10. Hirofumi Katsuno and Alberto Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. of KR'91*, pages 387–394, 1991.
11. Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.
12. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proc. of KR 2006*, pages 46–56, 2006.
13. Zhe Wang, Kewen Wang, and Rodney W. Topor. A new approach to knowledge base revision in *DL-Lite*. In *Proc. of AAAI 2010*. AAAI Press, 2010.
14. Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

Non-Uniform Data Complexity of Query Answering in Description Logics

Carsten Lutz[1] and Frank Wolter[2]

[1] Department of Computer Science, University of Bremen, Germany
[2] Department of Computer Science, University of Liverpool, UK
clu@uni-bremen.de,Wolter@liverpool.ac.uk

1 Introduction

In recent years, the use of ontologies to access instance data has become increasingly popular. The general idea is that an ontology provides a vocabulary or conceptual model for the application domain, which can then be used as an interface for querying instance data and to derive additional facts. In this emerging area, called ontology-based data access (OBDA), it is a central research goal to identify ontology languages for which query answering scales to large amounts of instance data. Since the size of the data is typically very large compared to the size of the ontology and the size of the query, the central measure for such scalability is provided by *data complexity*—the complexity of query answering where only the data is considered to be an input, but both the query and the ontology are fixed.

In description logic (DL), ontologies take the form of a TBox, instance data is stored in an ABox, and the most important class of queries are conjunctive queries (CQs). A fundamental observation regarding this setup is that, for expressive DLs such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$, the complexity of query answering is coNP-complete [12] and thus intractable (when speaking of complexity, we *always* mean data complexity). The most popular strategy to avoid this problem is to replace $\mathcal{ALC}$ and $\mathcal{SHIQ}$ with less expressive DLs that *are Horn* in the sense that they can be embedded into the Horn fragment of first-order (FO) logic and have minimal models that can be exploited for PTIME query answering. Horn DLs in this sense include, for example, logics from the $\mathcal{EL}$ and DL-Lite families as well as Horn-$\mathcal{SHIQ}$, a large fragment of $\mathcal{SHIQ}$ for which CQ-answering is still in PTIME [12]. While CQ-answering in Horn-$\mathcal{SHIQ}$ and the $\mathcal{EL}$ family of DLs is also hard for PTIME, the problem has even lower complexity in DL-Lite. In fact, the design goal of DL-Lite was to achieve *FO-rewritability*, i.e., that any CQ q and TBox $\mathcal{T}$ can be rewritten into an FO query q' such that the answers to q w.r.t. $\mathcal{T}$ coincide with the answers that a standard database system produces for q' [6]. Achieving this goal requires CQ-answering to be in AC^0.

It thus seems that the data complexity of query answering in a DL context is well-understood. However, all results discussed above are on the *level of logics*, i.e., each result concerns a class of TBoxes that is defined syntactically through expressibility in a certain logic, but no attempt is made to identify more structure *inside* these classes. The aim of this paper is to advocate a fresh look on the subject, by taking a novel approach. Specifically, we advocate a *non-uniform* study of the complexity of query answering by considering data complexity on the *level of individual TBoxes*. For a TBox $\mathcal{T}$, we say that *CQ-answering w.r.t. $\mathcal{T}$ is in* PTIME if for every CQ q, there is a PTIME algorithm that, given an ABox $\mathcal{A}$, computes the answers to q in $\mathcal{A}$ w.r.t. $\mathcal{T}$. In a similar way, we can define coNP-hardness and FO-rewritability on the TBox level. The non-uniform perspective allows us to investigate more fine-grained questions regarding the data complexity of query answering such as: given an expressive DL $\mathcal{L}$ such as $\mathcal{ALC}$ or $\mathcal{SHIQ}$, how can one characterize those $\mathcal{L}$-TBoxes $\mathcal{T}$ for which CQ-answering is in PTIME? How can we do the same for FO-rewritability? Is there a dichotomy for the complexity of query answering w.r.t. TBoxes formulated in $\mathcal{L}$, such as: for any $\mathcal{L}$-TBox $\mathcal{T}$, CQ-answering w.r.t. $\mathcal{T}$ is either in PTIME or coNP-hard?

In this paper, we consider TBoxes formulated in the expressive DL $\mathcal{ALCFI}$, answer some of the above questions, and take some steps towards others. Our main results are:

1. there is a dichotomy between PTIME and coNP-complete for CQ-answering w.r.t. $\mathcal{ALC}$-TBoxes if, and only if, Feder and Vardi's dichotomy conjecture that "constraint satisfaction problems (CSPs) with finite template are in PTIME or NP-complete" [10] is true; the same holds for $\mathcal{ALCI}$-TBoxes;
2. there is no dichotomy between PTIME and coNP-complete for CQ-answering w.r.t. $\mathcal{ALCF}$-TBoxes, unless PTIME $=$ NP; moreover, PTIME-complexity of CQ answering and many related problems are undecidable for $\mathcal{ALCF}$.
3. there is a dichotomy between PTIME and coNP-complete for CQ-answering w.r.t. $\mathcal{ALCFI}$-TBoxes of depth one, i.e., TBoxes where concepts have role depth ≤ 1;
4. FO-rewritability is decidable for Horn-$\mathcal{ALCFI}$-TBoxes of depth two and all Horn-$\mathcal{ALCF}$-TBoxes;

It should be noted that there has been steady progress regarding the dichotomy conjecture of Feder and Vardi over the last fifteen years and though the problem is still open, a solution does not seem completely out of reach [4, 5]. Our proof of Point 1 is based on a novel connection between CSPs and query answering w.r.t. $\mathcal{ALCI}$-TBoxes that can be exploited to transfer numerous results from the CSP world to query answering w.r.t. $\mathcal{ALCI}$-TBoxes and related problems. For example, together with [16, 5] we obtain the following results on 'FO-rewritability of ABox consistency':

5. Given an $\mathcal{ALCI}$-TBox $\mathcal{T}$, it can be decided in NEXPTIME whether there is an FO-sentence $\varphi_\mathcal{T}$ such that for all ABoxes $\mathcal{A}$, $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ viewed as an FO-structure satisfies $\varphi_\mathcal{T}$. Moreover, such a sentence $\varphi_\mathcal{T}$ exists iff ABox consistency w.r.t. $\mathcal{T}$ can be decided in non-uniform AC^0. Finally, if no such sentence $\varphi_\mathcal{T}$ exists, then ABox consistency w.r.t. $\mathcal{T}$ is LOGSPACE-hard (under FO-reductions).

To prove our results, we introduce some new notions that are relevant for studying the questions raised and prove some additional results of general interest. A central such notion is *materializability* of a TBox $\mathcal{T}$, which formalizes the existence of minimal models as known from Horn-DLs. We show that, in the case of TBoxes of depth one, materializability characterizes PTIME CQ-answering, which allows us to establish Point 2 above. For TBoxes of unrestricted depth, non-materializability still provides a sufficient condition for coNP-hardness of CQ-answering. We also develop the notion of *unraveling tolerance* of a TBox $\mathcal{T}$, which provides a sufficient condition for query answering to be in PTIME. The resulting upper bound strictly generalizes the known result that CQ-answering in Horn-$\mathcal{ALCFI}$ is in PTIME. Our framework also allows to formally establish some common intuitions and beliefs held in the context of CQ-answering in description logics. For example, we show that for any $\mathcal{ALCFI}$-TBox $\mathcal{T}$, CQ-answering is in PTIME iff answering positive existential queries is in PTIME iff answering $\mathcal{ELI}$-instance queries is in PTIME and likewise for FO-rewritability. Another observation in this spirit is that an $\mathcal{ALCFI}$-TBox is materializable (has minimal models) iff it is convex (a notion related to the entailment of disjunctions).

Most proofs in this paper are deferred to the (appendix of the) long version, which is available at http://www.csc.liv.ac.uk/~frank/publ/publ.html.

2 Preliminaries

We use standard notation for the syntax and semantics of $\mathcal{ALCFI}$ and other well-known DLs. Our TBoxes are finite sets of concept inclusions $C \sqsubseteq D$, where C and D are potentially compound concepts, and functionality assertions $\mathsf{func}(r)$, where r is a potentially inverse role. ABoxes are finite sets of assertions $A(a)$ and $r(a,b)$ with A a concept *name* and r a role name. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names used in the ABox $\mathcal{A}$ and sometimes write $r^-(a,b) \in \mathcal{A}$ instead of $r(b,a) \in \mathcal{A}$. For the interpretation of individual names, we make the unique name assumption.

A *first-order query (FOQ)* $q(\boldsymbol{x})$ is a first-order formula with free variables $\boldsymbol{x}$ constructed from atoms $A(t)$, $r(t,t')$, and $t = t'$ (where t, t' range over individual names and variables) using negation, conjunction, disjunction, and existential quantification. The variables in $\boldsymbol{x}$ are the *answer variables* of q. A FOQ without answer variables is *Boolean*. We say that a tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is an *answer to* $q(\boldsymbol{x})$ *in an interpretation* $\mathcal{I}$ if $\mathcal{I} \models q[\boldsymbol{a}]$, where $q[\boldsymbol{a}]$ results from replacing the answer variables $\boldsymbol{x}$ in $q(\boldsymbol{x})$ with $\boldsymbol{a}$. A tuple $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$ is a *certain answer to* $q(\boldsymbol{x})$ *in* $\mathcal{A}$ *given* $\mathcal{T}$, in symbols $\mathcal{T}, \mathcal{A} \models q(\boldsymbol{a})$, if $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$. Set $\mathsf{cert}_{\mathcal{T}}(q,\mathcal{A}) = \{\boldsymbol{a} \mid \mathcal{T}, \mathcal{A} \models q(\boldsymbol{a})\}$. A *positive existential query (PEQ)* $q(\boldsymbol{x})$ is a FOQ without negation and equality and a *conjunctive query (CQ)* is a positive existential query without disjunction. If C is an $\mathcal{ELI}$-concept and $a \in \mathsf{N_I}$, then $C(a)$ is an *$\mathcal{ELI}$-query (ELIQ)*. *$\mathcal{EL}$-queries (ELQs)* are defined analogously. Note that $\mathcal{ELI}$-queries and $\mathcal{EL}$-queries are always Boolean. In what follows, we sometimes slightly abuse notation and use FOQ to denote the set of all first-order queries, and likewise for CQ, PEQ, ELIQ, and ELQ.

Definition 1. *Let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox. Let $\mathcal{Q} \in \{CQ, PEQ, ELIQ, ELQ\}$. Then*

- $\mathcal{Q}$-answering w.r.t. $\mathcal{T}$ is in PTIME *if for every $q(\boldsymbol{x}) \in \mathcal{Q}$, there is a polytime algorithm that computes, given an ABox $\mathcal{A}$, the answer* $\mathsf{cert}_{\mathcal{T}}(q,\mathcal{A})$*;*
- $\mathcal{Q}$-answering w.r.t. $\mathcal{T}$ is coNP-hard *if there is a Boolean $q \in \mathcal{Q}$ such that, given an ABox $\mathcal{A}$, it is* coNP*-hard to decide whether $\mathcal{T}, \mathcal{A} \models q$;*
- $\mathcal{T}$ *is* FO-rewritable for $\mathcal{Q}$ *iff for every $q(\boldsymbol{x}) \in \mathcal{Q}$ one can effectively construct an FO-formula $q'(\boldsymbol{x})$ such that for every ABox $\mathcal{A}$,* $\mathsf{cert}_{\mathcal{T}}(q,\mathcal{A}) = \{\boldsymbol{a} \mid \mathcal{I}_{\mathcal{A}} \models q'(\boldsymbol{a})\}$*, where $\mathcal{I}_{\mathcal{A}}$ denotes $\mathcal{A}$ viewed as an interpretation.*

The above notions of complexity are rather robust under changing the query language: as we show next, neither the PTIME bounds nor FO-rewritability depend on whether we consider PEQs, CQs, or ELIQs.

Theorem 1. *For all $\mathcal{ALCFI}$-TBoxes $\mathcal{T}$, the following equivalences hold:*

1. *CQ-answering w.r.t. $\mathcal{T}$ is in* PTIME *iff PEQ-answering w.r.t. $\mathcal{T}$ is in* PTIME *iff ELIQ-answering w.r.t. $\mathcal{T}$ is in* PTIME*;*
2. *$\mathcal{T}$ is FO-rewritable for CQ iff it is FO-rewritable for PEQ iff it is FO-rewritable for ELIQ.*

If $\mathcal{T}$ is an $\mathcal{ALCF}$-TBox, then we can replace ELIQ in Points 1 and 2 with ELQ.

The proof is based on Theorems 2 and 3 below. Theorem 1 allows us to (sometimes) speak of the 'complexity of query answering' without reference to a query language.

3 Materializability

An important tool we use for analyzing the complexity of query answering is the notion of materializability of a TBox $\mathcal{T}$, which means that computing the certain answers to any query q and ABox $\mathcal{A}$ w.r.t. $\mathcal{T}$ reduces to evaluating q in a single model of $\mathcal{A}$ and $\mathcal{T}$.

Definition 2. *Let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox and $\mathcal{Q} \in \{CQ, PEQ, ELIQ, ELQ\}$. $\mathcal{T}$ is $\mathcal{Q}$-*materializable *if for every ABox $\mathcal{A}$ that is consistent w.r.t. $\mathcal{T}$, there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ such that $\mathcal{I} \models q[\boldsymbol{a}]$ iff $\mathcal{T}, \mathcal{A} \models q(\boldsymbol{a})$ for all $q(\boldsymbol{x}) \in \mathcal{Q}$ and $\boldsymbol{a} \subseteq \mathsf{Ind}(\mathcal{A})$.*

We show that PEQ, CQ, and ELIQ-materializability coincide (and for $\mathcal{ALC}$-TBoxes, all these also coincide with ELQ-materializability). Materializability is also equivalent to the following disjunction property (sometimes also called *convexity*): a TBox $\mathcal{T}$ has the *ABox disjunction property* if for all ABoxes $\mathcal{A}$ and ELIQs $C_1(a_1), \ldots, C_n(a_n)$, from $\mathcal{T}, \mathcal{A} \models C_1(a_1) \vee \ldots \vee C_n(a_n)$ it follows that $\mathcal{T}, \mathcal{A} \models C_i(a_i)$, for some $i \leq n$.

Theorem 2. *Let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox. The following equivalences hold: $\mathcal{T}$ is PEQ-materializable iff $\mathcal{T}$ is CQ-materializable iff $\mathcal{T}$ is ELIQ-materializable iff $\mathcal{T}$ has the ABox disjunction property.*

If $\mathcal{T}$ is an $\mathcal{ALC}$-TBox, the above are equivalent to ELQ-materializability.

Because of Theorem 2, we sometimes use the term materializability without reference to a query language. We call an interpretation $\mathcal{I}$ that satisfies the condition formulated in Definition 2 for PEQs a *minimal model* of $\mathcal{T}$ and $\mathcal{A}$. Note that in many cases, only an infinite minimal models exists. For example, for $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$ and $\mathcal{A} = \{A(a)\}$ every minimal model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ comprises an infinite r-chain starting at $a^{\mathcal{I}}$. Every TBox that is equivalent to an FO Horn sentence (in the general sense of [7]) is materializable: to construct a minimal model for such a TBox $\mathcal{T}$ and some ABox $\mathcal{A}$, one can take the direct product of all at most countable models of $\mathcal{T}$ and $\mathcal{A}$ (for additional information on direct products in DLs, see [17]). Conversely, however, there are simple materializable TBoxes that are not equivalent to FO Horn sentences.

Example 1. Let $\mathcal{T} = \{\exists r.(A \sqcap \neg B \sqcap \neg E) \sqsubseteq \exists r.(\neg A \sqcap \neg B \sqcap \neg E)\}$. One can easily show that $\mathcal{T}$ is not preserved under direct products; thus, it is not equivalent to a Horn sentence. However, one can construct a minimal model $\mathcal{I}$ for $\mathcal{T}$ and any ABox $\mathcal{A}$ by taking the interpretation $\mathcal{I}_{\mathcal{A}}$ obtained by viewing $\mathcal{A}$ as an interpretation and then adding, for any $a \in \mathsf{Ind}(\mathcal{A})$ with $a \in (\exists r.(A \sqcap \neg B \sqcap \neg E))^{\mathcal{I}_\mathcal{A}}$, a fresh d_a such that $(a, d_a) \in r^{\mathcal{I}}$ and d_a is not in the extension of any concept name. PEQ-answering w.r.t. $\mathcal{T}$ is FO-rewritable since for any PEQ q, $\mathsf{cert}_\mathcal{T}(q, \mathcal{A})$ consists of precisely the answers to q in $\mathcal{I}_\mathcal{A}$ (i.e., no query rewriting is necessary). Thus, PEQ-answering w.r.t. $\mathcal{T}$ is also in PTIME.

We show that materializability is a necessary condition for query answering being in PTIME.

Theorem 3. *If an $\mathcal{ALCFI}$-TBox $\mathcal{T}$ ($\mathcal{ALCF}$-TBox $\mathcal{T}$) is not materializable, then ELIQ-answering (ELQ-answering) is* coNP*-hard w.r.t. $\mathcal{T}$.*

The proof uses the violation of the ABox disjunction property stated in Theorem 2 and generalizes the reduction of 2+2-SAT used in [19] to prove that instance checking in a variant of $\mathcal{EL}$ is coNP-hard.

Materializability is not a sufficient condition for query answering to be in PTIME. In fact, we show that for any non-uniform constraint satisfaction problem, there is a materializable $\mathcal{ALC}$-TBox for which Boolean CQ-answering has the same complexity, up to complementation of the complexity class. For two finite relational FO-structures $\mathcal{R}$ and $\mathcal{R}'$ over relation symbols Σ, we write $\mathsf{Hom}(\mathcal{R}', \mathcal{R})$ if there is a homomorphism from $\mathcal{R}'$ to $\mathcal{R}$. The non-uniform constraint satisfaction problem for $\mathcal{R}$, denoted by $\mathsf{CSP}(\mathcal{R})$, is the problem to decide, for every finite $\mathcal{R}'$ over Σ, whether $\mathsf{Hom}(\mathcal{R}', \mathcal{R})$. Numerous algorithmic problems, among them many NP-complete ones such as k-SAT and k-colourability of graphs, can be given in the form $\mathsf{CSP}(\mathcal{R})$. It is known that every problem of the form $\mathsf{CSP}(\mathcal{R})$ is polynomially equivalent to some $\mathsf{CSP}(\mathcal{R}')$ with $\mathcal{R}'$ a digraph [10]. Thus, in what follows we can restrict ourselves to considering CSPs of the form $\mathsf{CSP}(\mathcal{I})$, where $\mathcal{I}$ is a DL interpretation. A *signature* Σ is a set of concept and role names. The signature $\mathsf{sig}(\mathcal{T})$ of a TBox $\mathcal{T}$ is the set of concept and role names that occur in $\mathcal{T}$. A Σ-TBox is a TBox that uses symbols from Σ only. Similar notation is used for ABoxes, concepts, and interpretations. For an ABox $\mathcal{A}$, we denote by $\mathcal{A}^\Sigma$ the subset of $\mathcal{A}$ containing symbols from Σ only. We will often not distinguish between ABoxes and finite interpretations.

Theorem 4. *For every non-uniform constraint satisfaction problem $\mathsf{CSP}(\mathcal{I})$, one can compute in polytime a materializable $\mathcal{ALC}$-TBox $\mathcal{T}$ such that for all ABoxes $\mathcal{A}$,*

1. *$\mathsf{Hom}(\mathcal{A}^\Sigma, \mathcal{I})$, with $\Sigma = \mathsf{sig}(\mathcal{I})$, iff $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$;*
2. *for any Boolean CQ q, answering q w.r.t. $\mathcal{T}$ is polynomially reducible to the complement of $\mathsf{CSP}(\mathcal{I})$.*

The proof Theorem 4 relies on the existence of $\mathcal{ALC}$-concepts H whose value $H^\mathcal{I}$ in interpretations $\mathcal{I}$ cannot be detected directly using CQs, but which can be used in a TBox to influence the values $A^\mathcal{I}$ of concept names A and, therefore, have an indirect effect on the answers to CQs. From the viewpoint of CQ query answering, they thus behave similarly to second-order variables. More precisely, let, for a finite set V of indices, Z_v, r_v, s_v be concept and role names, respectively. Let

$$\mathcal{T}_V = \{\top \sqsubseteq \exists r_v.\top, \top \sqsubseteq \exists s_v.Z_v \mid v \in V\}, \quad H_v = \forall r_v.\exists s_v.\neg Z_v.$$

Lemma 1. *For any ABox $\mathcal{A}$ and sets $I_v \subseteq \mathsf{Ind}(\mathcal{A})$, $v \in V$, one can construct a minimal model $\mathcal{I}$ of $(\mathcal{T}_V, \mathcal{A})$ such that $H_v^\mathcal{I} = I_v$ for all $v \in V$. $\mathcal{T}_V$ is FO-rewritable for PEQ.*

To prove Theorem 4, one extends the TBox $\mathcal{T}_V$. Assume $\mathsf{CSP}(\mathcal{I})$ is given. Let $V = \Delta^{\mathcal{I}}$ and assume, for simplicity, that $\mathsf{sig}(\mathcal{I}) = \{r\}$. Define

$$\mathcal{T} = \mathcal{T}_V \cup \{H_v \sqcap \exists r.H_w \sqsubseteq \bot \mid v, w \in V, (v,w) \notin r^{\mathcal{I}}\} \cup \{H_v \sqcap H_w \sqsubseteq \bot \mid v, w \in V, v \neq w\} \cup \{ \bigsqcap_{v \in V} \neg H_v \sqsubseteq \bot \}$$

Based on Lemma 1, it is possible to verify Points 1 and 2 of Theorem 4. For Point 2, it can be seen that for all Boolean CQs q and ABoxes $\mathcal{A}$, $(\mathcal{T}, \mathcal{A}) \models q$ iff $(\mathcal{T}_V, \mathcal{A}) \models q$ or not $\mathsf{Hom}(\mathcal{A}^{\Sigma}, \mathcal{I})$; since $\mathcal{T}_V$ is FO-rewritable, the former can be checked in PTime.

4 (Towards) Dichotomies

We start with a reduction of Boolean CQ-answering w.r.t. $\mathcal{ALCI}$-TBoxes to CSPs that yields, together with Theorem 4, a proof of Point 1 in the introduction: the dichotomy problem for CSPs is equivalent to the dichotomy problem for CQ answering w.r.t. $\mathcal{ALC}$- (and $\mathcal{ALCI}$-) TBoxes.

Theorem 5. *Let $\mathcal{T}$ be an $\mathcal{ALCI}$-TBox and $C(a)$ an ELIQ. Then one can construct, in time exponential in $|\mathcal{T}| + |C|$,*

1. *a Σ-interpretation $\mathcal{I}$, $\Sigma = (\mathsf{sig}(\mathcal{T}) \cup \mathsf{sig}(C)) \uplus \{P\}$, with P a concept name, such that for all ABoxes $\mathcal{A}$,*
   - *(a) there is a polynomial reduction of answering $C(a)$ w.r.t. $\mathcal{T}$ to the complement of $\mathsf{CSP}(\mathcal{I})$;*
   - *(b) there is a polynomial reduction from the complement of $\mathsf{CSP}(\mathcal{I})$ to Boolean CQ-answering w.r.t. $\mathcal{T}$;*
2. *a Σ-interpretation $\mathcal{I}$, $\Sigma = \mathsf{sig}(\mathcal{T})$, such that for every ABox $\mathcal{A}$, $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathsf{Hom}(\mathcal{A}^{\Sigma}, \mathcal{I})$.*

For Point 1, $\mathcal{I}$ is in fact the interpretation that is obtained by the standard type elimination procedure for $\mathcal{ALCI}$-TBoxes $\mathcal{T}$ and concepts C. More specifically, let S be the closure under single negation of all subconcepts of $\mathcal{T}$ and C. A *type* t is a maximal subset of S that is satisfiable w.r.t. $\mathcal{T}$. Then $\Delta^{\mathcal{I}}$ is the set of all types, $t \in A^{\mathcal{I}}$ iff $A \in t$, and $(t, t') \in r^{\mathcal{I}}$ iff $\forall r.D \in t$ implies $D \in t'$ and $\forall r^{-}.D \in t'$ implies $D \in t$. For the special concept name P, set $P^{\mathcal{I}} = \{t \mid C \notin t\}$. With the type elimination algorithm, $\mathcal{I}$ can be constructed in exponential time. The mentioned reductions are then as follows:

(a) $(\mathcal{T}, \mathcal{A}) \models C(a)$ iff not $\mathsf{Hom}(\mathcal{A}^{\Sigma}_{P(a)}, \mathcal{I})$, where $\mathcal{A}_{P(a)}$ results from $\mathcal{A}$ by adding $P(a)$ to $\mathcal{A}$ and removing all other assertions using P from $\mathcal{A}$;
(b) not $\mathsf{Hom}(\mathcal{A}^{\Sigma}, \mathcal{I})$ iff $(\mathcal{T}, \mathcal{A}) \models \exists v.(P(v) \wedge C(v))$.

Result 1 from the introduction can be derived as follows. Let $\mathsf{CSP}(\mathcal{I})$ be an NP-intermediate CSP, i.e., a CSP that is neither in PTime nor NP-hard. Take the TBox $\mathcal{T}$ from Theorem 4. By Point 1 of that theorem and since consistency of ABoxes w.r.t. $\mathcal{T}$ can trivially be reduced to the complement of answering Boolean CQs w.r.t. $\mathcal{T}$, CQ-answering w.r.t. $\mathcal{T}$ is not in PTime. By Point 2, CQ-answering w.r.t. $\mathcal{T}$ is not coNP-hard either. Conversely, let $\mathcal{T}$ be a TBox for which CQ-answering w.r.t. $\mathcal{T}$ is neither in

PTIME nor coNP-hard. Then by Theorem 1 and since every ELIQ is a CQ, the same holds for ELIQ-answering w.r.t. $\mathcal{T}$. Thus, there is a concrete ELIQ $C(a)$ such that answering $C(a)$ w.r.t. $\mathcal{T}$ is coNP-intermediate. Let $\mathcal{I}$ be the interpretation constructed in Point 1 of Theorem 5 for $\mathcal{T}$ and $C(a)$. By Point 1a, $\mathsf{CSP}(\mathcal{I})$ is not in PTIME; by Point 1b, it is not NP-hard either.

Result 5 from the introduction can be derived as follows. It is proved in [16, 5] that the problem to decide whether the class of structures $\{\mathcal{I}' \mid \mathsf{Hom}(\mathcal{I}', \mathcal{I})\}$ is FO-definable is NP-complete. We obtain a NEXPTIME upper bound since the template $\mathcal{I}$ associated with $\mathcal{T}$ can be constructed in exponential time. The claims for AC^0 and LOGSPACE follow in the same way from other results in [16, 5].

We now develop a condition on TBoxes, called unraveling tolerance, that is sufficient for PTIME CQ-answering and strictly generalizes Horn-$\mathcal{ALCFI}$, the $\mathcal{ALCFI}$-fragment of Horn-$\mathcal{SHIQ}$. For the case of TBoxes of depth one, we obtain a PTIME/coNP dichotomy result. The notion of unraveling tolerance is based on an unraveling operation on ABoxes, in the same spirit as the well-known unraveling of an interpretation into a tree interpretation. This is inspired by (i) the observation that, in the proof of Theorem 3, the non-tree-shape of ABoxes is essential; and (ii) by Theorem 5 together with the known fact the non-uniform CSPs are tractable when restricted to tree-shaped input structures. The *unraveling* $\mathcal{A}_u$ of an ABox $\mathcal{A}$ is the following ABox:

- the individual names $\mathsf{Ind}(\mathcal{A}_u)$ of $\mathcal{A}_u$ are sequences $b_0 r_0 b_1 \cdots r_{n-1} b_n$, $b_0, \dots, b_n \in \mathsf{Ind}(\mathcal{A})$ and $r_0, \dots, r_{n-1}$ (possibly inverse) roles such that for all $i < n$, we have $r_i(b_i, b_{i+1}) \in \mathcal{A}$ and $b_{i+1} \neq b_{i-1}$ (whenever $i > 0$);
- for each $C(b) \in \mathcal{A}$ and $\alpha = b_0 r_0 b_1 \cdots r_{n-1} b_n \in \mathsf{Ind}(\mathcal{A}_u)$ with $b_n = b$, we have $C(\alpha) \in \mathcal{A}_u$;
- for each $b_0 r_0 b_1 \cdots r_{n-1} b_n \in \mathsf{Ind}(\mathcal{A}_u)$, we have $r_{n-1}(b_{n-1}, b_n) \in \mathcal{A}_u$.

For all $\beta = b_0 r_0 \cdots r_{n-1} b_n \in \mathsf{Ind}(\mathcal{A}_u)$, we write $\mathsf{tail}(\beta)$ to denote b_n. Note that the condition $b_{i+1} \neq b_{i-1}$ is needed to ensure that functional roles can still be interpreted in a functional way after unraveling, despite the UNA.

Definition 3. *A TBox $\mathcal{T}$ is* unraveling tolerant *if for all ABoxes $\mathcal{A}$ and ELIQs q, we have that $\mathcal{T}, \mathcal{A} \models q$ implies $\mathcal{T}, \mathcal{A}_u \models q$.*

It is not hard to prove that the converse direction '$\mathcal{T}, \mathcal{A}_u \models q$ implies $\mathcal{T}, \mathcal{A} \models q$' is true for *all* $\mathcal{ALCFI}$-TBoxes. We now show that the class of unraveling tolerant $\mathcal{ALCFI}$-TBoxes generalizes Horn-$\mathcal{ALCFI}$. This is based on the original and most general definition of Horn-$\mathcal{SHIQ}$ in [12] and thus also captures weaker variants as used e.g. in [13, 9]. The TBox in Example 1, which is unraveling tolerant but not a Horn-$\mathcal{ALCFI}$-TBox, demonstrates that the generalization is strict.

Lemma 2. *Every Horn-$\mathcal{ALCFI}$-TBox is unraveling tolerant.*

It is interesting to note that unraveling tolerance implies materializability. We shall see that the converse is, in general, not true.

Lemma 3. *Every unraveling-tolerant $\mathcal{ALCFI}$-TBox is materializable.*

We now show that unraveling tolerance yields a class of $\mathcal{ALCFI}$-TBoxes for which query answering is in PTIME. By Lemma 2 and since we actually exhibit a *uniform* algorithm for query answering w.r.t. unraveling tolerant TBoxes, this also reproves the known PTIME upper bound for CQ-answering in Horn-$\mathcal{ALCFI}$ [9]. This result is not a consquence of Theorem 4 and known results for CSPs since we capture full $\mathcal{ALCFI}$.

Theorem 6. *If an $\mathcal{ALCFI}$-TBox $\mathcal{T}$ is unraveling tolerant, then PEQ-answering w.r.t. $\mathcal{T}$ is in* PTIME.

To see that unraveling tolerance does not capture all $\mathcal{ALCFI}$-TBoxes for which query answering is in PTIME, we can invoke Theorem 4. For example, taking a CSP for 2-colorability, we obtain a TBox $\mathcal{T}$ for which CQ-answering is in PTIME and such that an ABox $\mathcal{A}$ with $\mathsf{sig}(\mathcal{A}) = \{r\}$ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is 2-colorable. Thus, $\mathcal{A}, \mathcal{T} \models X(a)$, X a fresh concept name, iff $\mathcal{A}$ is not 2-colorable. It follows that $\mathcal{T}$ is not unraveling tolerant. We conjecture that it is possible to generalize Theorem 6 to larger classes of TBoxes by relaxing the operation of ABox unraveling such that it yields ABoxes of bounded treewidth instead of tree-shaped ABoxes. Such a generalization would still not capture 2-colorability.

We now turn to TBoxes of depth one. The central observation is that for this special case, we can prove a converse of Lemma 3.

Lemma 4. *Every materializable $\mathcal{ALCFI}$-TBox of depth one is unraveling tolerant.*

This brings us into the position where we can establish the announced dichotomy result for $\mathcal{ALCFI}$-TBoxes of depth one. If such a TBox $\mathcal{T}$ is materializable, then Lemma 4 and Theorem 6 yield that PEQ-answering w.r.t. $\mathcal{T}$ is in PTIME. Otherwise, ELIQ-answering w.r.t. $\mathcal{T}$ is coNP-complete by Theorem 3. We thus obtain the following.

Theorem 7 (Dichotomy). *For every $\mathcal{ALCFI}$-TBox $\mathcal{T}$ of depth one, one of the following is true:*

- *$\mathcal{Q}$-answering w.r.t. $\mathcal{T}$ is in* PTIME *for any $\mathcal{Q} \in \{PEQ, CQ, ELIQ\}$;*
- *$\mathcal{Q}$-answering w.r.t. $\mathcal{T}$ is* coNP*-complete for any $\mathcal{Q} \in \{PEQ, CQ, ELIQ\}$.*

5 Deciding FO-Rewritability

The results of this section are based on the observation that for materializable TBoxes of depth one, FO-rewritability for CQ follows from FO-rewritability for *atomic* concepts, i.e., concept names and $\bot$. We say that an atomic concept A is *FO-rewritable w.r.t. a TBox $\mathcal{T}$ and a signature Σ* if there exists an FO-formula φ_A such that for all Σ-ABoxes $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$: $\mathcal{T}, \mathcal{A} \models A(a)$ iff $\mathcal{I}_\mathcal{A} \models \varphi_A[a]$. Clearly, if $\mathcal{T}$ is FO-rewritable for CQ, then every atomic concept is FO-rewritable w.r.t. $\mathcal{T}$ and any signature. For materializable TBoxes of depth one, the converse is also true.

Lemma 5. *A materializable $\mathcal{ALCFI}$-TBox of depth one is FO-rewritable for CQs iff all atomic concepts are FO-rewritable w.r.t. $\mathcal{T}$ and* $\mathsf{sig}(\mathcal{T})$.

Based on Lemma 5, we can use Theorem 5 and results from [16] to obtain the following result, in a similar (but slightly more involved) way as in the proof of Result 5 from the introduction.

Theorem 8. *FO-rewritability for CQs is decidable in* NEXPTIME, *for any of the following classes of TBoxes: materializable $\mathcal{ALCI}$-TBoxes of depth one, Horn-$\mathcal{ALC}$-TBoxes, and Horn-$\mathcal{ALCI}$-TBoxes of depth two.*

Theorem 5 does not apply to DLs with functional roles. To analyze FO-rewritability in the presence of functional roles, we associate with every materializable TBox $\mathcal{T}$ of depth one a monadic datalog program $\Pi_{\mathcal{T}}$ such that $\mathcal{T}$ and $\Pi_{\mathcal{T}}$ give the same answers to queries $A(a)$, A atomic. We then show that $\mathcal{T}$ is FO-rewritable if, and only if, $\Pi_{\mathcal{T}}$ is equivalent to a non-recursive datalog program. The latter property is known as *boundedness* of a datalog program and has been studied extensively for fixpoint logics [3, 18] and datalog programs [8]. Using existing decidability results for boundedness, we can then establish a counterpart of Theorem 8 for the case of $\mathcal{ALCFI}$.

For our purposes, a monadic datalog program Π consists of rules $A(x) \leftarrow X$, where A is a concept name and X is a finite set consisting of assertions of the form $B(x)$, $r(x_1, x_2)$, and inequalities $x_1 \neq x_2$, where B is a concept name, r a role, and x, x_1, x_2 range over variables. Inequalities are required to model functional roles. We also use a special unary predicate $\perp$ and rules $\perp(x) \leftarrow X$ stating that X is inconsistent. For an ABox $\mathcal{A}$, we denote by $\Pi^i(\mathcal{A})$ the set of all assertions $A(a)$ that can be derived using i applications of rules from Π to $\mathcal{A}$. We set $\Pi^\infty(\mathcal{A}) = \bigcup_{i\geq 0} \Pi^i(\mathcal{A})$.

Definition 4 (Boundedness). *Let Π be a datalog program and Σ a signature. An atomic concept A* is bounded in Π for Σ-ABoxes *if there exists a $k > 0$ such that for all Σ-ABoxes $\mathcal{A}$ and all $a \in \mathsf{sig}(\mathcal{A})$: $A(a) \in \Pi^\infty(\mathcal{A})$ iff $A(a) \in \Pi^k(\mathcal{A})$.*

Let $\mathcal{T}$ be a materializable TBox of depth one. A Σ-neighbourhood ABox (Σ-NH) consists of a Σ-ABox $\mathcal{A}$ with a distinguished individual name f such that $\mathcal{A}$ consists of assertions of the form $r(f, a)$ with r a role and $a \neq f$ and $A(b)$ such that

- for each $b \neq f$ with $b \in \mathsf{Ind}(\mathcal{A})$ there is exactly one r such that $r(f, b) \in \mathcal{A}$;
- if $r(f, b_1)$ and $r(f, b_2) \in \mathcal{A}$ and $b_1 \neq b_2$, then there exists $A(b_1) \in \mathcal{A}$ with $A(b_2) \notin \mathcal{A}$ or vice versa.

The ABox $\mathcal{A}$ in which each individual b is replaced by a variable x_b is denoted by $\mathcal{A}^x$. Now define a monadic datalog program associated with $\mathcal{T}$, where $\Sigma = \mathsf{sig}(\mathcal{T})$:

$$\begin{aligned}
\Pi_{\mathcal{T}} = \{A(x_a) \leftarrow \mathcal{A}^x \mid \mathcal{A} \text{ is a } \Sigma\text{-NH}, a \in \mathsf{Ind}(\mathcal{A}), A \in \Sigma, (\mathcal{T}, \mathcal{A}) \models A(a)\} \cup \\
\{\perp(x) \leftarrow \mathcal{A}^x \mid \mathcal{A} \text{ is a } \Sigma\text{-NH that is not consistent w.r.t. } \mathcal{T}\} \cup \\
\{\perp(x) \leftarrow r(y, y_1), r(y, y_2), y_1 \neq y_2 \mid \mathsf{func}(r) \in \mathcal{T}\} \cup \\
\{A(x) \leftarrow \perp(x) \mid A \in \Sigma\}.
\end{aligned}$$

The following lemma states that $\Pi_{\mathcal{T}}$ behaves as intended.

Lemma 6. *For every materializable $\mathcal{ALCFI}$-TBox $\mathcal{T}$ of depth one, every $A \in \mathsf{sig}(\mathcal{T})$, every ABox $\mathcal{A}$, and every $a \in \mathsf{Ind}(\mathcal{A})$, $(\mathcal{T}, \mathcal{A}) \models A(a)$ iff $A(a) \in \Pi^\infty_{\mathcal{T}}(\mathcal{A})$. Moreover, $\perp(a) \in \Pi^\infty_{\mathcal{T}}(\mathcal{A})$ iff $\mathcal{A}$ is not consistent w.r.t. $\mathcal{T}$.*

Using unfolding tolerance of materializable TBoxes of depth one, one can show the following equivalence for FO-rewritability and boundedness.

Lemma 7. *For every materializable $\mathcal{ALCFI}$-TBox $\mathcal{T}$ of depth one and signature Σ: an atomic concept A is bounded in $\Pi_{\mathcal{T}}$ for Σ-ABoxes iff A is FO-rewritable w.r.t. $\mathcal{T}$ and Σ.*

Unfortunately, decidability results for boundedness of monadic datalog programs are not directly applicable to $\Pi_{\mathcal{T}}$ since they assume programs without inequalities [8, 11]. However, using unfolding tolerance, one can employ instead recent decidability results on boundedness of least fixed points over trees [18] to obtain the following theorem.

Theorem 9. *FO-rewritability for CQs is decidable, for any of the following classes of TBoxes: materializable $\mathcal{ALCFI}$-TBoxes of depth one, Horn-$\mathcal{ALCF}$-TBoxes, and Horn-$\mathcal{ALCFI}$-TBoxes of depth two.*

6 Non-Dichotomy and Undecidability in $\mathcal{ALCF}$

The aim of this section is to show that the addition of functional roles significantly complicates the problems studied in the previous sections. More precisely, we show that (i) for CQ-answering w.r.t. $\mathcal{ALCF}$-TBoxes, there is no dichotomy between PTIME and coNP unless PTIME = NP; and (ii) CQ-answering in PTIME is undecidable for $\mathcal{ALCF}$-TBoxes, and likewise for coNP-hardness, materializability and FO-rewritability. Point (i) is a consequence of the following result.

Theorem 10. *For every language L in* coNP*, there is an $\mathcal{ALCF}$-TBox $\mathcal{T}$ and query* $\mathsf{rej}(a)$, rej *a concept name, such that the following holds:*

1. *there exists a polynomial reduction of deciding $v \in L$ to answering* $\mathsf{rej}(a)$ *w.r.t. $\mathcal{T}$;*
2. *for every ELIQ q, answering q w.r.t. $\mathcal{T}$ is polynomially reducible to deciding $v \in L$.*

Ladners theorem [15] states that unless PTIME = NP, coNP intermediate problems exist. Suppose to the contrary of Point (i) that for every $\mathcal{ALCF}$-TBox $\mathcal{T}$, CQ answering w.r.t. $\mathcal{T}$ is in PTIME or coNP-hard. Take a coNP-intermediate language L and let $\mathcal{T}$ be the TBox from Theorem 10. By Point 1 of the theorem, CQ-answering w.r.t. $\mathcal{T}$ is not in PTIME. Thus it must be coNP-hard. By Theorem 1 and since a dichotomy for CQ-answering w.r.t. $\mathcal{T}$ also implies a dichotomy for ELIQ-answering w.r.t. $\mathcal{T}$, ELIQ-answering w.r.t. $\mathcal{T}$ is also coNP-hard. By Point 2 of Theorem 10, this is impossible.

The proof of Theorem 10 combines the 'hidden' concepts H_v from the proof of Theorem 4 with ideas from a proof in [1] which establishes undecidability of a certain *query emptiness* problem in $\mathcal{ALCF}$. Using a similar strategy, we establish the undecidability results announced as Point (ii) above, summarized by the following theorem.

Theorem 11. *For $\mathcal{ALCF}$-TBoxes $\mathcal{T}$, the following problems are undecidable (Points 1 and 2 are subject to the side condition that* PTIME $\neq$ NP*):*

1. *CQ-answering w.r.t. $\mathcal{T}$ is in* PTIME*;*
2. *CQ answering w.r.t. $\mathcal{T}$ is* coNP*-hard;*
3. *$\mathcal{T}$ is materializable.*

In the appendix, we also prove that FO-rewritability for CQ is undecidable in $\mathcal{ALCF}$, for a slightly modified definition of FO-rewritability that only considers *consistent* ABoxes.

7 Conclusions

We have have introduced non-uniform data complexity of query answering w.r.t. description logic TBoxes and proved that it enables a more fine-grained analysis than the standard approach. Many questions remain. In particular, the newly established CSP-connection should be exploited further. We believe that the techniques introduced in this paper can be extended to richer DLs such as $\mathcal{SHIQ}$.

Acknowledgments. C. Lutz was supported by the DFG SFB/TR 8 "Spatial Cognition".

References

1. F. Baader, M. Bienvenu, C. Lutz, and F. Wolter. Query and predicate emptiness in description logics. In *Proc. of KR2010*. AAAI Press, 2010.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI05*, pages 364–369. Professional Book Center, 2005.
3. J. Barwise and Y. N. Moschovakis. Global inductive definability. *J. Symb. Log.*, 43(3):521–534, 1978.
4. A. A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. of FOCS02*, pages 649–658, 2002.
5. A. A. Bulatov, A. A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints*, pages 93–124, 2008.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Autom. Reasoning*, 39(3):385–429, 2007.
7. C. C. Chang and H. J. Keisler. *Model Theory*, volume 73 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1990.
8. S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proc. of STOC88*, pages 477–490, 1988.
9. T. Eiter, G. Gottlob, M. Ortiz, and M. Simkus. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proc. of JELIA08*, volume 5293 of LNCS, pages 166–179. Springer, 2008.
10. T. Feder and M. Y. Vardi. Monotone monadic snp and constraint satisfaction. In *Proc. of STOC93*, pages 612–622, 1993.
11. H. Gaifman, H. G. Mairson, Y. Sagiv, and M. Y. Vardi. Undecidable optimization problems for database logic programs. In *Proc. of LICS87*, pages 106–115, 1987.
12. U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
13. Y. Kazakov. Consequence-driven reasoning for horn SHIQ ontologies. In *Proc. of IJCAI09*, pages 2040–2045, 2009.
14. M. Krötzsch. Efficient inferencing for OWL EL. In *Proc. of JELIA10*, volume 6341 of LNCS, pages 234–246. Springer, 2010.
15. R. E. Ladner. On the structure of polynomial time reducibility. *JACM*, 22(1):155171, 1975.
16. B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4), 2007.
17. C. Lutz, R. Piro, and F. Wolter. Description logic tboxes: Model-theoretic characterizations and rewritability. In *Proc. of IJCAI11*, 2011.
18. M. Otto, A. Blumensath, and M. Weyer. Decidability results for the boundedness problem. Technical report, TU Darmstadt, 2010.
19. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intell. Inf. Sys.*, 2:265–278, 1993.

Practical ABox cleaning in DL-Lite (progress report)

Giulia Masotti, Riccardo Rosati, Marco Ruzzi

Dipartimento di Informatica e Sistemistica “Antonio Ruberti”
Sapienza Università di Roma
Via Ariosto 25, I-00185 Roma, Italy

1 Introduction

One of the most important current issues in Description Logic (DL) ontology management is dealing with inconsistency, that is, the presence of contradictory information in the ontology [7]. It is well-known that the classical semantics of DLs is not *inconsistency-tolerant*, i.e., it does not allow for using in a meaningful way any piece of information in an inconsistent ontology. On the other hand, the size of ontologies used by real applications is scaling up, and ontologies are increasingly merged and integrated into larger ontologies: the probability of creating inconsistent ontologies is consequently getting higher and higher.

In this paper we focus on ABox inconsistency, i.e., the case of inconsistent KBs where the TBox is consistent while the ABox is inconsistent with the TBox, i.e., a subset of the assertions in the ABox contradicts a TBox assertion (or a subset of the TBox). In particular, we are interested in defining a form of automatic *ABox cleaning*, i.e., given $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we want to identify an ABox $\mathcal{A}'$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent and $\mathcal{A}'$ is “as close as possible” to $\mathcal{A}$.

The kind of ABox cleaning we adopt is formally based on inconsistency-tolerant semantics, which overcome the limitations of the classical DL semantics in inconsistency management. In particular, we consider inconsistency-tolerant semantics for general DLs recently proposed in [4], called *IAR semantics* and *$ICAR$ semantics*, for which reasoning has been studied in the context of the Description Logics of the *DL-Lite* family. The notion of ABox repair in the IAR semantics is very simple: the ABox repair of a DL ontology is the intersection of all the maximal subsets of the ABox that are consistent with the TBox. The notion of ABox repair in the $ICAR$ semantics is a variant of the IAR semantics that is based on a notion of “equivalence under consistency” of ABoxes inconsistent with respect to a given TBox. In [4] it was proved that computing the ABox repair of a *DL-Lite$_A$* ontology is tractable both under IAR semantics and $ICAR$ semantics.

We argue that the results of [4] are very important from the practical viewpoint, for the following reasons: (i) they provide (to the best of our knowledge) the first formally grounded notion of ABox cleaning. In other words, IAR and $ICAR$ are the first inconsistency-tolerant semantics that allow for expressing ABox repairs in terms of a single ABox; (ii) they identify (to the best of our knowledge) the first tractable inconsistency-tolerant semantics in DLs. This paper starts from the above results, and tries to provide an experimental validation that ABox cleaning based on the above semantics is actually feasible. More precisely, we provide the following contributions:

(1) We present effective techniques for ABox cleaning in *DL-Lite*$_A$ under IAR and $ICAR$ semantics. To this aim, we present the Quonto ABox Cleaner (QuAC), which implements, within the Quonto system,[1] techniques for the computation of the ABox repair of a *DL-Lite*$_A$ knowledge base under the above semantics. QuAC constitutes (to the best of our knowledge) the first implementation of a tractable ABox cleaning algorithm for DL ontologies. Moreover, since Quonto delegates the management of the ABox to a relational database system (DBMS), all modifications of the ABox are delegated to the DBMS through SQL queries and updates. This potentially allows for handling and cleaning very large ABoxes.
(2) We report on the experimental analysis that we are actually conducting using QuAC. Our first results are allowing us to understand the actual impact, w.r.t. the efficiency of ABox cleaning, of the different aspects involved in the computation of the ABox repair, and the limits and possibilities of the approach implemented in QuAC.

The paper that is closer to our work is [3], which also presents a technique for ABox cleaning in DL ontologies. However, there are two main differences with our approach: (i) [3] considers the very expressive DL $\mathcal{SHIN}$, in which all the semantics considered by our approach are intractable ([6]); (ii) the two approaches are based on different semantics: in particular, the ABox cleaning algorithm of [3] computes a consistent subset of the ABox which in general is uncomparable with the ABox repair defined by the IAR semantics (and the $ICAR$ semantics).

The rest of the paper is organized as follows. In Section 2, we give some preliminaries, and in particular we introduce *DL-Lite*$_A$ and the definition of the IAR and the $ICAR$ semantics. In Section 3, we present detailed algorithms for ABox cleaning in *DL-Lite*$_A$. In Section 4 we present the QuAC system and report on the experiments we are currently conducting with QuAC. Finally, in Section 5 we conclude the paper.

2 Preliminaries

2.1 The DL *DL-Lite*$_A$

In this paper we consider DL ontologies (knowledge bases) specified in *DL-Lite*$_A$, a member of the *DL-Lite* family of tractable Description Logics [2,1], which is at the basis of OWL 2 QL, one of the profile of OWL 2, the official knowledge base specification language of the World-Wide-Web Consortium (W3C). *DL-Lite*$_A$ distinguishes concepts from *value-domains*, which denote sets of (data) values, and roles from *attributes*, which denote binary relations between objects and values. Concepts, roles, attributes, and value-domains in this DL are formed according to the following syntax:

$$\begin{array}{ll}
B \longrightarrow A \mid \exists Q \mid \delta(U) & E \longrightarrow \rho(U) \\
C \longrightarrow B \mid \neg B & F \longrightarrow \top_D \mid T_1 \mid \cdots \mid T_n \\
Q \longrightarrow P \mid P^- & V \longrightarrow U \mid \neg U \\
R \longrightarrow Q \mid \neg Q &
\end{array}$$

In such rules, A, P, and U respectively denote an atomic concept (i.e., a concept name), an atomic role (i.e., a role name), and an attribute name, P^- denotes the inverse of an atomic role, whereas B and Q are called basic concept and basic role, respectively.

[1] `http://www.dis.uniroma.it/~quonto`

Furthermore, $\delta(U)$ denotes the *domain* of U, i.e., the set of objects that U relates to values; $\rho(U)$ denotes the *range* of U, i.e., the set of values that U relates to objects; $\top_D$ is the universal value-domain; $T_1, \ldots, T_n$ are n pairwise disjoint unbounded value-domains.

A *DL-Lite$_A$* knowledge base (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the TBox and $\mathcal{A}$ the ABox. The TBox $\mathcal{T}$ is a finite set of assertions of the form

$$B \sqsubseteq C \qquad Q \sqsubseteq R \qquad E \sqsubseteq F \qquad U \sqsubseteq V \qquad (\mathsf{funct}\ Q) \qquad (\mathsf{funct}\ U)$$

From left to right, the first four assertions respectively denote inclusions between concepts, roles, value-domains, and attributes. In turn, the last two assertions denote functionality on roles and on attributes. In fact, in *DL-Lite$_A$* TBoxes we further impose that roles and attributes occurring in functionality assertions cannot be specialized (i.e., they cannot occur in the right-hand side of inclusions). Let B_1 and B_2 be basic concepts, and let Q_1 and Q_2 be basic roles. We call *positive inclusions (PIs)* assertions of the form $B_1 \sqsubseteq B_2$, and of the form $Q_1 \sqsubseteq Q_2$, whereas we call *negative inclusions (NIs)* assertions of the form $B_1 \sqsubseteq \neg B_2$ and $Q_1 \sqsubseteq \neg Q_2$.

A *DL-Lite$_A$* ABox $\mathcal{A}$ is a finite set of membership assertions (ABox assertions) of the forms $A(a)$, $P(a, b)$, and $U(a, v)$, where A, P, and U are as above, a and b belong to Γ_O, the subset of Γ_C containing object constants, and v belongs to Γ_V, the subset of Γ_C containing value constants, where $\{\Gamma_O, \Gamma_V\}$ is a partition of Γ_C.

The semantics of a *DL-Lite$_A$* knowledge base is given in terms of first-order logic (FOL) interpretations in the usual way. An interpretation $\mathcal{I}$ satisfying a knowledge base $\mathcal{K}$ a called a *model* for $\mathcal{K}$. In the following *Mod*($\langle \mathcal{T}, \mathcal{A} \rangle$) will indicate the set of models of the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. A knowledge base $\mathcal{K}$ is satisfiable if it has at least a model, otherwise it is called unsatisfiable. Given an assertion α (which is either a TBox or ABox assertion), we write $\mathcal{K} \models \alpha$ if α is satisfied in every model for $\mathcal{K}$.

Given a TBox $\mathcal{T}$ and an ABox $\mathcal{A}'$, $\mathcal{A}'$ is called a *minimal conflict set for* $\mathcal{T}$ if the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$ is unsatisfiable and, for every ABox $\mathcal{A}''$ such that $\mathcal{A}'' \subset \mathcal{A}'$, the KB $\langle \mathcal{T}, \mathcal{A}'' \rangle$ is satisfiable. A minimal conflict set for $\mathcal{T}$ is called *unary* if its cardinality (that is, the number of assertions it contains) is 1 and is called *binary* if its cardinality is 2.

2.2 Inconsistency-tolerant semantics for DLs

In this section we recall the inconsistency-tolerant semantics for general DL knowledge bases defined in [4].[2] We assume that, for a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{T}$ is satisfiable, whereas $\mathcal{A}$ may be inconsistent with $\mathcal{T}$, i.e., the set of models of $\mathcal{K}$ may be empty.

AR-semantics The first notion of repair that we consider, called *AR-repair*, is a very natural one: a repair is a maximal subset of the ABox that is consistent with the TBox. Thus, an AR-repair is obtained by throwing away from $\mathcal{A}$ a minimal set of assertions to make it consistent with $\mathcal{T}$.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An AR-*repair *of $\mathcal{K}$ is a set $\mathcal{A}'$ of membership assertions such that: (i)$\mathcal{A}' \subseteq \mathcal{A}$; (ii) Mod($\langle \mathcal{T}, \mathcal{A}' \rangle$) $\neq \emptyset$; (iii) there does not exist*

[2] Due to space limitations, we refer the reader to [4] for introductory examples illustrating these semantics.

$\mathcal{A}''$ such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ and $Mod(\langle \mathcal{T}, \mathcal{A}'' \rangle) \neq \emptyset$. The set of AR-repairs for $\mathcal{K}$ is denoted by AR-Rep$(\mathcal{K})$. Moreover, we say that a first-order sentence ϕ is AR-entailed *by $\mathcal{K}$, written $\mathcal{K} \models_{AR} \phi$, if $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$ for every $\mathcal{A}' \in$ AR-Rep$(\mathcal{K})$.*

CAR-semantics We start by formally introducing a notion of "equivalence under consistency" for inconsistent KBs.

Given a KB $\mathcal{K}$, let $\mathcal{S}_K$ denote the signature of $\mathcal{K}$, i.e., the set of concept, role, and individual names occurring in $\mathcal{K}$. Given a signature $\mathcal{S}$, we denote with $HB(\mathcal{S})$ the *Herbrand Base of* $\mathcal{S}$, i.e. the set of ABox assertions (ground atoms) that can be built over the signature $\mathcal{S}$. Then, given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we define the *consistent logical consequences of* $\mathcal{K}$ as the set $clc(\mathcal{K}) = \{\alpha \mid \alpha \in HB(\mathcal{S}_K)$ and there exists $\mathcal{A}' \subseteq \mathcal{A}$ such that $Mod(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$ and $\langle \mathcal{T}, \mathcal{A}' \rangle \models \alpha\}$. Finally, we say that two KBs $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}, \mathcal{A}' \rangle$ are *consistently equivalent (C-equivalent)* if $clc(\langle \mathcal{T}, \mathcal{A} \rangle) = clc(\langle \mathcal{T}, \mathcal{A}' \rangle)$.

We argue that the notion of C-equivalence is very reasonable in settings in which the ABox (or at least a part of it) has been "closed" (in a complete or partial way) with respect to the TBox, e.g., when (some or all) the ABox assertions that are entailed by the ABox and the TBox have been added to the original ABox. This may happen, for example, when the ABox is obtained by integrating different (and locally consistent) sources, since some of these sources might have been locally closed with respect to some TBox axioms: this is very likely, for instance, if a source is an RDF graph with RDFS predicates, since many RDF systems materialize in the RDF graph the implicit triples due to the RDFS predicates.

In settings where C-equivalence makes sense, the AR-semantics is not suited to handle inconsistency. In fact, we would expect two C-equivalent KBs to produce the same logical consequences under inconsistency-tolerant semantics. Unfortunately, the AR-semantics does not have this property. A simple example is the following: let $\mathcal{T} = \{student \sqsubseteq young,\ student \sqsubseteq \neg worker\}$ and let $\mathcal{A} = \{student(mary), worker(mary)\}$, $\mathcal{A}' = \{student(mary), worker(mary), young(mary)\}$. It is immediate to verify that if $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, then $clc(\mathcal{K}) = clc(\mathcal{K}') = \mathcal{A}'$, thus $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent, however $\mathcal{K}' \models_{AR} young(mary)$ while $\mathcal{K} \not\models_{AR} young(mary)$.

To overcome the above problem, the CAR-semantics has been defined in [4], through a modification of the AR-semantics.[3]

Definition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. A* CAR-repair *for $\mathcal{K}$ is a set $\mathcal{A}'$ of membership assertions such that $\mathcal{A}'$ is an AR-repair of $\langle \mathcal{T}, clc(\mathcal{K}) \rangle$. The set of CAR-repairs for $\mathcal{K}$ is denoted by CAR-Rep$(\mathcal{T}, \mathcal{A})$. Moreover, we say that a first-order sentence ϕ is* CAR-entailed *by $\mathcal{K}$, written $\mathcal{K} \models_{CAR} \phi$, if $\langle \mathcal{T}, \mathcal{A}' \rangle \models \phi$ for every $\mathcal{A}' \in$ CAR-Rep$(\mathcal{K})$.*

Going back to the previous example, it is immediate to see that, since $\mathcal{K}$ and $\mathcal{K}'$ are C-equivalent, the set of CAR-repairs (and hence the set of CAR-models) of $\mathcal{K}$ and $\mathcal{K}'$ coincide. As the above example shows, there are sentences entailed by a KB under CAR-semantics that are not entailed under AR-semantics. Conversely, it is shown in

[3] The definition provided here of the CAR-semantics is a slight simplification of the one appearing in [4]: this modification, however, does not affect any of the computational results presented in [4].

[4] that the AR-semantics is a sound approximation of the CAR-semantics, i.e., for every KB $\mathcal{K}$ and every FOL sentence ϕ, $\mathcal{K} \models_{AR} \phi$ implies $\mathcal{K} \models_{CAR} \phi$.

IAR-semantics and ICAR-semantics We then recall the IAR-semantics and $ICAR$-semantics, which are sound approximations of the AR-semantics and the CAR-semantics, respectively [4].

Definition 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. Then: (i) The* IAR-repair *for $\mathcal{K}$, denoted by IAR-Rep($\mathcal{K}$) is defined as IAR-Rep($\mathcal{K}$) $= \bigcap_{\mathcal{A}' \in AR\text{-}Rep(\mathcal{K})} \mathcal{A}'$. (ii) The* $ICAR$-repair *for $\mathcal{K}$, denoted by ICAR-Rep($\mathcal{K}$) is defined as ICAR-Rep($\mathcal{K}$) $= \bigcap_{\mathcal{A}' \in CAR\text{-}Rep(\mathcal{K})} \mathcal{A}'$. (iii) We say that a first-order sentence ϕ is* IAR-entailed *(respectively,* $ICAR$-entailed*) by $\mathcal{K}$, and we write $\mathcal{K} \models_{IAR} \phi$ (respectively, $\mathcal{K} \models_{ICAR} \phi$), if $\langle \mathcal{T}, IAR\text{-}Rep(\mathcal{K}) \rangle \models \phi$ (respectively, $\langle \mathcal{T}, ICAR\text{-}Rep(\mathcal{K}) \rangle \models \phi$).*

Example 1. Let us consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where the TBox $\mathcal{T}$ is the following:

$$\mathcal{T} = \{A \sqsubseteq C, B \sqsubseteq C, \exists R \sqsubseteq B, \exists R^- \sqsubseteq D,\ A \sqsubseteq \neg B, (\mathsf{funct}\ R)\}$$

and the ABox $\mathcal{A}$ is $\mathcal{A} = \{A(a), B(a), C(a), R(a,b)\}$. Such a KB is unsatisfiable, due to the presence of the assertions $A(a)$ and $B(a)$ which violate the disjointness assertion in $\mathcal{T}$. The following are the standard AR-repairs of $\mathcal{A}$:

$$\mathcal{A}_{AR}{}^1 = \{B(a), C(a), R(a,b)\}, \quad \mathcal{A}_{AR}{}^2 = \{A(a), C(a)\}$$

Then, we have $clc(A) = \{A(a), B(a), C(a), R(a,b), D(a)\}$. Therefore, the CAR-repair of $\mathcal{A}$ are as follows:

$$\mathcal{A}_{CAR}{}^1 = \{B(a), C(a), R(a,b), D(b)\}, \quad \mathcal{A}_{CAR}{}^2 = \{A(a), C(a), D(b)\}$$

Consequently, the IAR-repair and ICAR-repair are the following:

$$\mathcal{A}_{IAR} = \mathcal{A}_{AR}{}^1 \cap \mathcal{A}_{AR}{}^2 = \{C(a)\}, \ \ \mathcal{A}_{ICAR} = \mathcal{A}_{CAR}{}^1 \cap \mathcal{A}_{CAR}{}^2 = \{C(a), D(a)\}$$

Example 2. One might conjecture that the IAR semantics collapses into a simple ABox cleaning technique which deletes from the ABox all the assertions that participate in conflicts with the TBox. This is actually not the case, because, as explained in [4], the IAR-repair actually deletes only the assertions that participate in *minimal* conflict sets. Here is an example: given the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T} = \{A \sqsubseteq \neg\ \exists R,\ R \sqsubseteq \neg R^-\}$, $\mathcal{A} = \{A(a),\ R(a,a)\}$, the IAR-repair of $\mathcal{K}$ is $\{A(a)\}$. That is, the assertion $A(a)$ belongs to the IAR-repair even if it participates in the conflict set $\{A(a),\ R(a,a)\}$ caused by the concept disjointness $A \sqsubseteq \neg \exists\ R$: the reason is that such a conflict set is not minimal because of the unary conflict set $\{R(a,a)\}$ caused by the role disjointness $R \sqsubseteq \neg R^-$.

3 Algorithms for ABox cleaning

The technique for computing the ICAR-repair of a *DL-Lite*$_A$ ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is based on the idea of deleting from $\mathcal{A}$ all the membership assertions participating in *minimal* conflict sets for $\mathcal{T}$. As shown in [4], this task is relatively easy (in particular, tractable) in *DL-Lite*$_A$ because the following property holds: for every *DL-Lite*$_A$ TBox $\mathcal{T}$, all the minimal conflict sets for $\mathcal{T}$ are either unary conflict sets or binary conflict sets.

This property is actually crucial for tractability of reasoning under IAR and $ICAR$ semantics. As shown in [6] this property is not shared by other tractable DLs (e.g. $\mathcal{EL}_\perp$), in which the size of minimal conflict sets is not bound by a constant but depends on the size of the ABox.

We now present detailed algorithms for computing the IAR-repair and the $ICAR$-repair of a *DL-Lite*$_A$ ontology. These algorithms exploits the techniques presented in [4], whose aim was only to provide PTIME upper bounds for the problem of computing such repairs. In particular, the present algorithms specify efficient ways of detecting minimal conflict sets and computing consistent logical consequences. Instead, the previous techniques check all unary and binary subsets of the ABox for these purposes.

In the following, we call *annotated ABox assertion* an expression ξ of the form $\langle\alpha,\gamma\rangle$ where α is an ABox assertion and γ is a value in the set $\{\mathit{cons}, \mathit{ucs}, \mathit{bcs}\}$. Furthermore, we call *annotated ABox* a set of annotated ABox assertions. The intuition behind an annotated ABox assertion ξ is that its annotation γ expresses whether the associated ABox expression α does not participate in any minimal conflict set (*cons*) or participates in a unary conflict set (*ucs*) or to a binary conflict set (*bcs*).

The following algorithm *QuAC-ICAR* computes the $ICAR$-repair of a *DL-Lite*$_A$ KB. For ease of exposition, the algorithm does not report details on the treatment of attributes, which are actually handled in a way analogous to roles.

```
**Algorithm** *QuAC-ICAR*($\mathcal{K}$)
**input**: *DL-Lite*$_A$ KB $\mathcal{K} = \langle\mathcal{T},\mathcal{A}\rangle$,   **output**: $ICAR$-repair of $\mathcal{K}$
**begin**
// STEP 1: create annotated ABox $\mathcal{A}_{ann}$
  $\mathcal{A}_{ann} = \emptyset$;
  **for each** $\alpha \in \mathcal{A}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \langle\alpha, \mathit{cons}\rangle$;
// STEP 2: detect unary conflict sets in $\mathcal{A}_{ann}$
  **for each** concept name $A$ s.t. $\mathcal{T} \models A \sqsubseteq \neg A$ **do**
      **for each** $\xi = \langle A(a), \mathit{cons}\rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle A(a), \mathit{ucs}\rangle\}$;
  **for each** role name $R$ s.t. $\mathcal{T} \models R \sqsubseteq \neg R$ **do**
      **for each** $\xi = \langle R(a,b), \mathit{cons}\rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle R(a,b), \mathit{ucs}\rangle\}$;
  **for each** role name $R$ s.t. $\mathcal{T} \models R \sqsubseteq \neg R^-$ or $\mathcal{T} \models \exists R \sqsubseteq \neg\exists R^-$ **do**
      **for each** $\xi = \langle R(a,a), \mathit{cons}\rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle R(a,a), \mathit{ucs}\rangle\}$;
// STEP 3: compute consistent logical consequences in $\mathcal{A}_{ann}$
  **for each** inclusion $A \sqsubseteq B$ with $A, B$ atomic concepts such that $\mathcal{T} \models A \sqsubseteq B$ **do**
      **for each** $\langle A(a), \gamma\rangle \in \mathcal{A}_{ann}$ such that $\gamma \neq \mathit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle B(a), \mathit{cons}\rangle\}$;
  **for each** inclusion $\exists R \sqsubseteq A$ with $A$ atomic concept such that $\mathcal{T} \models \exists R \sqsubseteq A$ **do**
      **for each** $\langle R(a,b), \gamma\rangle \in \mathcal{A}_{ann}$ such that $\gamma \neq \mathit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle A(a), \mathit{cons}\rangle\}$;
  **for each** inclusion $\exists R^- \sqsubseteq A$ with $A$ atomic concept such that $\mathcal{T} \models \exists R^- \sqsubseteq A$ **do**
      **for each** $\langle R(a,b), \gamma\rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \mathit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle A(b), \mathit{cons}\rangle\}$;
  **for each** inclusion $R \sqsubseteq S$ with $R, S$ atomic roles such that $\mathcal{T} \models R \sqsubseteq S$ **do**
      **for each** $\langle R(a,b), \gamma\rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \mathit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle S(a,b), \mathit{cons}\rangle\}$;
  **for each** inclusion $R^- \sqsubseteq S$ with $R, S$ atomic roles such that $\mathcal{T} \models R^- \sqsubseteq S$ **do**
      **for each** $\langle R(b,a), \gamma\rangle \in \mathcal{A}_{ann}$ and $\gamma \neq \mathit{ucs}$ **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} \cup \{\langle S(a,b), \mathit{cons}\rangle\}$;
// STEP 4: detect binary conflict sets in $\mathcal{A}_{ann}$
  **for each** disjointness $A \sqsubseteq \neg B$ with $A, B$ atomic concepts
      such that $\mathcal{T} \models A \sqsubseteq \neg B$ **do**
      **for each** pair $\xi_1 = \langle A(a), \gamma_1\rangle, \xi_2 = \langle B(a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
```

```
            such that $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle B(a), bcs\rangle\}$;
    **for each** disjointness $A \sqsubseteq \neg\exists R$ with $A$ atomic concept
        such that $\mathcal{T} \models A \sqsubseteq \neg\exists R$ **do**
        **for each** pair $\xi_1 = \langle A(a), \gamma_1\rangle, \xi_2 = \langle R(a,b), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle R(a,b), bcs\rangle\}$;
    **for each** disjointness $A \sqsubseteq \neg\exists R^-$ with $A$ atomic concept
        such that $\mathcal{T} \models A \sqsubseteq \neg\exists R$ **do**
        **for each** pair $\xi_1 = \langle A(a), \gamma_1\rangle, \xi_2 = \langle R(b,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle A(a), bcs\rangle, \langle R(b,a), bcs\rangle\}$;
    **for each** disjointness $R \sqsubseteq \neg S$ with $R, S$ atomic roles
        such that $\mathcal{T} \models R \sqsubseteq \neg S$ **do**
        **for each** pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle S(a,b), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle S(a,b), bcs\rangle\}$;
    **for each** disjointness $R \sqsubseteq \neg S^-$ with $R, S$ atomic roles
        such that $\mathcal{T} \models R \sqsubseteq \neg S^-$ **do**
        **for each** pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle S(b,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $\gamma_1, \gamma_2 \neq$ *ucs*
            **do** $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle S(b,a), bcs\rangle\}$;
    **for each** functionality assertion $(\mathsf{funct}\ R) \in \mathcal{T}$ with $R$ atomic role **do**
        **for each** pair $\xi_1 = \langle R(a,b), \gamma_1\rangle, \xi_2 = \langle R(a,c), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $b \neq c$ and $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(a,b), bcs\rangle, \langle R(a,c), bcs\rangle\}$;
    **for each** functionality assertion $(\mathsf{funct}\ R^-) \in \mathcal{T}$ with $R$ atomic role **do**
        **for each** pair $\xi_1 = \langle R(b,a), \gamma_1\rangle, \xi_2 = \langle R(c,a), \gamma_2\rangle \in \mathcal{A}'_{ann}$
            such that $b \neq c$ and $\gamma_1, \gamma_2 \neq$ *ucs* **do**
            $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle R(b,a), bcs\rangle, \langle R(c,a), bcs\rangle\}$;
// STEP 5: extract the ICAR repair from $\mathcal{A}_{ann}$
    $\mathcal{A}' = \emptyset$;
    **for each** $\langle \alpha, cons\rangle \in \mathcal{A}_{ann}$ **do** $\mathcal{A}' = \mathcal{A}' \cup \{\alpha\}$;
    **return** $\mathcal{A}'$
**end**
```

The algorithm *QuAC-ICAR* consists of five steps which can be informally described as follows.

step 1 *copy of $\mathcal{A}$ into an annotated ABox $\mathcal{A}_{ann}$.* In this step, the value of the annotation is initialized to *cons* for all ABox assertions.

step 2 *detection of the unary conflict sets in $\mathcal{A}_{ann}$.* For every assertion of the form $\xi = \langle \alpha, \mathit{cons}\rangle$, such that $\{\alpha\}$ is a unary conflict set for $\mathcal{T}$, $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi\} \cup \{\langle \alpha, \mathit{ucs}\rangle\}$, i.e., the annotation relative to α is changed to *ucs*. Unary conflict sets are actually detected through TBox reasoning, by looking at empty concepts and roles in $\mathcal{T}$, as well as asymmetric roles, i.e., roles disjoint with their inverse.

step 3 *computation of the consistent logical consequences in $\mathcal{A}_{ann}$.* Here, the task is to compute all ABox assertions that are entailed by $\mathcal{T}$ together with any $\mathcal{T}$-consistent subset of $\mathcal{A}$. In *DL-Lite$_A$*, this actually corresponds to computing the ABox assertions that are entailed by $\mathcal{T}$ together with the ABox obtained from $\mathcal{A}$ by deleting all unary conflict sets for $\mathcal{T}$. Hence, what the algorithms does is computing the ABox assertions that are logical consequence of $\mathcal{T}$ and of the assertions of $\mathcal{A}_{ann}$ which have not been annotated as unary conflict sets in the previous step.

step 4 *detection of the binary conflict sets in $\mathcal{A}_{ann}$*. For every pair of assertions of the form $\xi_1 = \langle \alpha_1, \gamma_1 \rangle, \xi_2 = \langle \alpha_2, \gamma_2 \rangle$ such that $\gamma_1 \neq \mathit{ucs}$ and $\gamma_2 \neq \mathit{ucs}$ and $\{\alpha,\beta\}$ is a binary conflict set for $\mathcal{T}$, $\mathcal{A}_{ann} = \mathcal{A}_{ann} - \{\xi_1, \xi_2\} \cup \{\langle \alpha, \mathit{bcs} \rangle, \langle \beta, \mathit{bcs} \rangle\}$, i.e., the annotation relative to α and β is changed to *bcs*. As in the case of unary conflict sets, to find binary conflict sets the algorithm looks for disjoint concepts and roles in $\mathcal{T}$, as well as functional roles.

step 5 *extraction of the ICAR-repair from $\mathcal{A}_{ann}$*. The $ICAR$-repair can be now simply extracted from the annotated ABox $\mathcal{A}_{ann}$, by eliminating both unary conflict sets and binary conflict sets. Therefore, for every assertion of the form $\langle \alpha, \mathit{cons} \rangle$ in $\mathcal{A}_{ann}$, α is copied into the (non-annotated) ABox $\mathcal{A}'$ which is finally returned by the algorithm.

The algorithm *QuAC-IAR* is very similar to *QuAC-ICAR*: the only difference is that it does not execute step 3, i.e., computation of consistent logical consequences. Correctness of the above algorithms can be proved starting from the results in [4].

Theorem 1. *Let $\mathcal{K}$ be a DL-Lite$_A$ KB and let $\mathcal{A}'$ be the ABox returned by QuAC-ICAR($\mathcal{K}$). Then, $\mathcal{A}'$ = ICAR-Rep($\mathcal{K}$). Moreover, let $\mathcal{A}''$ be the ABox returned by QuAC-IAR($\mathcal{K}$). Then, $\mathcal{A}''$ = IAR-Rep($\mathcal{K}$).*

4 Implementation and experiments

We have implemented the above algorithms *QuAC-ICAR* and *QuAC-IAR* within the Quonto system, in a module called QuAC (the Quonto ABox Cleaner). Essentially, QuAC is a Java implementation of the above algorithms where operations on the involved ABoxes are delegated to a relational database system (DBMS). In fact, in the Quonto architecture, the management of the ABox is delegated to a DBMS: therefore, all the operations on ABox assertions of the algorithms for computing repairs are executed in QuAC by the DBMS used by Quonto, through appropriate SQL scripts.

We are currently experimenting QuAC in order to answer several open questions, among which:

- the computational cost of the various steps of the algorithm *QuAC-IAR* and *QuAC-ICAR*;
- the scalability of such algorithms;
- measuring the difference in terms of computational costs of the IAR semantics and the $ICAR$ semantics;
- the impact of the “degree of inconsistency” of the ABox on the computational cost of the algorithms.

The tables reported in Figure 1 and Figure 2 present some of the experimental results that we have obtained so far. The TBox used in the experiments has 30 concept names, 20 role names, 10 attribute names, and about 200 TBox assertions. The various ABoxes used have been automatically generated.

The first table presents the experimental results for a version of Quonto that uses a main memory database (H2) to handle the ABox, while the second table presents the same results when Quonto uses a standard (disk-resident) database (PostgreSQL). The results have been conducted on a Pentium i7 (2.67 GHz) CPU with 6GB RAM under Windows 7 (64 bit) operating system. We have also executed the same tests using the MySQL DBMS, with results analogous to those obtained with PostgreSQL.

In the tables, the first column reports the number of assertions in the ABox, while the second column reports the percentage of ABox assertions that participate in minimal conflict sets for the considered TBox. Moreover:

- Δ_1 denotes the time to create the annotated ABox;
- Δ_2^{IAR} denotes the time to detect unary and binary conflict sets;
- Δ_3^{IAR} denotes the time to extract the IAR-repair from the annotated ABox;
- Δ_2^{ICAR} denotes the time to detect unary conflict sets, compute consistent logical consequences and detect binary conflict sets;
- Δ_3^{ICAR} denotes the time to extract the $ICAR$-repair from the annotated ABox;
- Δ^{IAR} is the total time to compute the IAR-repair, i.e., $\Delta_1 + \Delta_2^{IAR} + \Delta_3^{IAR}$;
- Δ^{ICAR} is the total time to compute the $ICAR$-repair, i.e., $\Delta_1 + \Delta_2^{ICAR} + \Delta_3^{ICAR}$;
- all times are expressed in milliseconds.

ABox size	% incons.	Δ_1	Δ_2^{IAR}	Δ_3^{IAR}	Δ^{IAR}	Δ_2^{ICAR}	Δ_3^{ICAR}	Δ^{ICAR}
1,000	1%	188	296	109	593	749	250	1,187
1,000	5%	188	358	78	624	749	250	1,187
1,000	10%	188	296	94	578	749	266	1,203
3,000	1%	359	670	251	1,280	1,997	266	2,622
3,000	5%	359	795	234	1,388	1,997	251	2,607
3,000	10%	359	717	126	1,202	1,997	282	2,638
10,000	1%	515	874	141	1,530	3,495	1,424	5,434
10,000	5%	515	781	171	1,467	3,495	1,376	5,386
10,000	10%	515	982	172	1,669	3,495	1,156	5,166
30,000	1%	812	3,075	422	4,309	22,635	3,559	27,006
30,000	5%	812	3,355	418	4,585	22,635	2,498	25,945
30,000	10%	812	3,417	344	4,573	22,635	2,748	26,195

Fig. 1. Results for main memory database (H2)

The above experimental results show that:

(i) with both the main memory DB and the disk-resident DB, the computation of the IAR-repair (Δ^{IAR} column) seems really scalable, while the computation of the $ICAR$-repair suffers from the additional step of computing logical consequences, which is computationally very expensive: its cost actually dominates the cost of all the other steps;

(ii) the percentage of inconsistency, i.e., the fraction of ABox assertions that participate in minimal conflict sets, does not have a significant impact on the execution time of both algorithms;

ABox size	% incons.	Δ_1	Δ_2^{IAR}	Δ_3^{IAR}	Δ^{IAR}	Δ_2^{ICAR}	Δ_3^{ICAR}	Δ^{ICAR}
$1,000$	1%	718	516	5,117	6,351	1,358	6,314	7,672
$1,000$	5%	718	515	5,258	6,491	1,358	6,070	7,428
$1,000$	10%	718	531	4,680	5,929	1,358	5,929	7,287
$3,000$	1%	1,840	688	5,444	7,972	4,011	8,747	12,758
$3,000$	5%	1,840	750	5,366	7,956	4,011	7,317	11,328
$3,000$	10%	1,840	797	5,304	7,941	4,011	8,284	12,295
$10,000$	1%	5,850	1,171	10,235	17,256	16,990	19,115	36,105
$10,000$	5%	5,850	1,499	10,297	17,646	16,990	19,424	36,414
$10,000$	10%	5,850	1,561	9,923	17,334	16,990	17,926	34,916
$30,000$	1%	16,255	3,823	20,702	40,780	134,286	65,959	200,245
$30,000$	5%	16,255	4,790	20,999	42,044	134,286	61,170	195,456
$30,000$	10%	16,255	5,539	20,281	42,075	134,286	63,736	198,022

Fig. 2. Results for PostgreSQL

(iii) using the main memory DB, most of the computation time for the IAR-repair is devoted to the detection of minimal conflict sets (i.e., Δ_2^{IAR}); conversely, using the disk-resident DB, a very large percentage of the execution time (always more than 80%) is devoted to the generation of the annotated ABox and to the extraction of the IAR-repair. This is of course due to the fact that such steps require to create and write a large number of records on the disk. On the other hand, RAM size is a bottleneck for the main memory DB (we were not able to process ABoxes with 100,000 assertions).

5 Ongoing and future work

As above observed, most of the execution time of the algorithm *QuAC-IAR* using a disk-resident DB is due to the creation of the annotated ABox (step 1) and to the creation of the IAR-repair (step 5). Thus, avoiding these steps would dramatically improve the efficiency of this algorithm.

To this aim, we observe that both the above steps could be completely skipped if the database schema used for representing the ABox would present an additional attribute for storing annotations in every relation (the usual DB representation of an ABox uses a unary relation for every concept and a binary relation for every role). This corresponds to the idea of directly using an annotated ABox instead of a standard ABox in the system. In this case, the computation of the IAR-repair could only consist of the steps 2, 3 and 4 of the algorithm *QuAC-IAR*. However, the choice of using an annotated ABox instead of an ABox affects query answering, since the queries evaluated on an annotated ABox should be able to filter out the assertions whose annotation is equal to *cons*.

We are currently experimenting whether this choice is actually feasible. Below we present a table showing the evaluation time of four queries of increasing complexity on the ABoxes considered in the previous section (in particular, the ABoxes with 5% inconsistent assertions). We show the cost of both evaluating the query on the IAR-repair (represented as a standard ABox) and directly on the annotated ABox (with the further selection condition on the annotations).

ABox size	query	query ans. on IAR-repair (nsec)	query ans. on annotated ABox (nsec)	difference (msec)	difference (%)
1,000	q1	123,577	105,868	-17	-17%
1,000	q2	216,740	226,750	10	4%
1,000	q3	1,179,561	295,275	-884	-299%
1,000	q4	421,161	600,174	179	30%
3,000	q1	138,591	229,060	90	39%
3,000	q2	210,581	355,716	145	41%
3,000	q3	1,348,179	490,842	-857	-175%
3,000	q4	507,396	653,685	146	22%
10,000	q1	164,384	339,932	175	52%
10,000	q2	267,172	499,696	232	47%
10,000	q3	1,347,024	592,475	-754	-127%
10,000	q4	491,612	664,465	172	26%
30,000	q1	199,417	724,521	525	72%
30,000	q2	398,448	905,074	506	56%
30,000	q3	1,519,493	944,726	-574	-61%
30,000	q4	485,067	1,096,021	610	56%

These first experimental results show that, in Quonto, evaluating queries on the annotated ABox often seems computationally not much harder (and sometimes even easier) than evaluating them on the standard ABox. Therefore, a more detailed experimental analysis is needed in order to understand the conditions under which it could be convenient to only work with an annotated representation of the ABox.

Finally, it would be very interesting to compare the performance of QuAC with a query rewriting approach. Indeed, techniques for the perfect rewriting of unions of conjunctive queries over $DL\text{-}Lite_A$ KBs under both IAR and $ICAR$ semantics have been recently defined [5]. Such techniques are able to reduce query answering over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ to answering a FOL query over the ABox $\mathcal{A}$. So, the ABox is not repaired at all by this approach: rather, the ABox repair is virtually considered during query answering through a suitable reformulation of the query. We plan to implement such query rewriting techniques, with the aim of comparing such an approach with the approach of QuAC.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
3. J. Dolby, J. Fan, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, J. W. Murdock, K. Srinivas, and C. A. Welty. Scalable cleanup of information extraction data using ontologies. In *ISWC/ASWC*, pages 100–113, 2007.
4. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of RR 2010*, 2010.
5. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Query rewriting for inconsistent DL-Lite ontologies. In *Proc. of RR 2011*, 2011. To appear.
6. R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. of IJCAI 2011*, 2011. To appear.
7. Z. Wang, K. Wang, and R. W. Topor. A new approach to knowledge base revision in *DL-Lite*. In *Proc. of AAAI 2010*. AAAI Press, 2010.

Practical Epistemic Entailment Checking in $\mathcal{SROIQ}$

Anees Mehdi and Sebastian Rudolph

Institute AIFB, Karlsruhe Institute of Technology, DE
{anees.mehdi,sebastian.rudolph}@kit.edu

Abstract. In this paper, we present a reasoner capable of epistemic inferences in $\mathcal{SROIQ}$ knowledge bases. We first identify some counter intuitive effects of imposing the traditional semantics in epistemic extensions of expressive description logics (DLs). Thus, we provide a revised *downward compatible* semantics with a more intuitive behavior in such cases. Based on the new semantics, we present a reduction method for epistemic queries to standard DL reasoning. This enables us to deploy state-of-the-art DL reasoners for such non-standard inferences. Additionally, we provide an implementation of our approach and present first evaluation results.

1 Introduction

In the early 1980s, Hector J. Levesque questioned the adequateness of the query language in knowledge formalisms [6]. He proposed the idea of embedding the epistemic operator $\mathbf{K}$ into a query language, thereby achieving the capabilities of knowledge base introspection. A similar line of research was initiated by R. Reiter in the context of databases [9]. Due to the extended reasoning capabilities, epistemic extensions have also been investigated (cf. e.g. [3, 2, 4]) in the context of Description Logics (DLs, cf. [1]).

To see the usefulness of the $\mathbf{K}$ operator for epistemic querying, consider the following example. Assume we want to query for "known white wines that are not known to be produced in a French region" which can be solved by performing instance retrieval w.r.t. the epistemic DL concept $\mathbf{K}\,\mathit{WhiteWine} \sqcap \neg\exists\mathbf{K}\mathit{locatedIn}.\{\mathit{FrenchRegion}\}$. This query will not only retrieve the wines that are explicitly excluded from being French wines but also those for which there is just no evidence that they are French (neither directly nor indirectly via deduction). For a knowledge base containing $\{\mathit{WhiteWine}(\mathit{MountadamRiesling}), \mathit{locatedIn}(\mathit{MountadamRiesling}, \mathit{AustralianRegion})\}$ as a subset, the query would yield *MountadamRiesling* as a result, whereas the same query without epistemic operators would produce an empty result. Moreover, by adding additional information such as *MountadamRiesling* being located in a French region, the answer to the epistemic query would also become empty, which illustrates that introducing the epistemic operator into a logic brings about non-monotonicity.

Another typical use case for epistemic querying is integrity constraint checking: testing whether the axiom

$$\mathsf{K}\,\mathit{Wine} \sqsubseteq \exists \mathsf{K}\mathit{hasSugar}.\{\mathit{Dry}\} \sqcup \exists \mathsf{K}\mathit{hasSugar}.\{\mathit{OffDry}\} \sqcup \exists \mathsf{K}\mathit{hasSugar}.\{\mathit{Sweet}\}$$

is entailed allows to check whether for every named individual in the knowledge base that is known to be a wine it is also known (i.e. it can be logically derived from the ontology) what degree of sugar it has. Note that this cannot be taken for granted even if $\mathit{Wine} \sqsubseteq \exists \mathit{hasSugar}.\{\mathit{Dry}\} \sqcup \exists \mathit{hasSugar}.\{\mathit{OffDry}\} \sqcup \exists \mathit{hasSugar}.\{\mathit{Sweet}\}$ is stated in (or can be derived from) the ontology.

These examples illustrate an obvious added value of epistemic extensions of description logics in practical applications. However former research – focused on extending tableaux algorithms for less expressive languages – has not paced up with the development of reasoners for very expressive DLs. In fact, as we will discuss in the course of this paper, some expressive features like nominal concepts require special care when combined with the idea of introspection by epistemic operators.

This paper investigates the epistemic extension of the very expressive DL $\mathcal{SROIQ}$ [5]. When applying a semantics along the lines of [4] to $\mathcal{SROIQ}$ we observe effects that clearly contradict natural requirements for epistemic reasoning (that we call backward compatibility). This directly leads to the question for an altered semantics that "behaves well" also for $\mathcal{SROIQ}$. We introduce such a semantics and show that it complies with the proposed requirements. With the more adequate semantics at hand, we then turn to the question of efficient algorithms for the specific problem of answering queries to classical (i.e., K-free) $\mathcal{SROIQ}$ ontologies. We solve this problem by providing a sound and complete reduction from epistemic querying to standard DL reasoning; our approach reduces occurrences of the K operator to intermediate calls to a standard DL reasoner. Employing this technique, existing reasoners for non-epistemic DLs can be reused in a black-box fashion for the task of answering epistemic queries. Based on this algorithm, we implemented a reasoner capable of answering epistemic queries to $\mathcal{SROIQ}$ knowledge bases. For the complete proofs and technical details, we refer the interested reader to the accompanying technical report [7].

2 Epistemic DLs and the Classical Semantics

We consider $\mathcal{SROIQ}$ as the underlying DL (for details see [5]). The extension of $\mathcal{SROIQ}$ with the epistemic operator $\mathbf{K}$, denoted by $\mathcal{SROIQK}$, allows $\mathbf{K}$ to appear in front of concept or role expressions. We call a $\mathcal{SROIQK}$-role an *epistemic role* if $\mathbf{K}$ occurs in it. An epistemic role is *simple* if it is of the form $\mathbf{K}S$ where S is a simple $\mathcal{SROIQ}$-role.

Following the way epistemic semantics for DLs have been hitherto defined (see, e.g., [4]), the classical semantics of $\mathcal{SROIQK}$ is given as *possible world semantics* in terms of *epistemic interpretations*. Thereby the following two central assumptions are made. (1) *Common Domain Assumption*: all interpretations are defined over a fixed countably infinite domain Δ. (2) *Rigid Term Assumption*: for all interpretations, the mapping from individuals to domains elements is fixed (it is just the identity function). Due to these assumptions, we can w.l.o.g. stipulate $\Delta := N_I \cup \mathbb{N}$. Essentially, these assumptions are imposed in order to ensure that (sets of) domain elements can be referred to and dealt with uniformly in a cross-domain manner.
Next, we provide the definition of epistemic interpretations. The main difference to the non-epistemic case, is that we provide a "context" of relevant models which are inspected whenever the extension of an epistemic concept or role is to be determined.

Definition 1. An *epistemic interpretation* for $\mathcal{SROIQK}$ is a pair $(\mathcal{I}, \mathcal{W})$ where $\mathcal{I}$ is a $\mathcal{SROIQ}$-interpretation and $\mathcal{W}$ is a set of $\mathcal{SROIQ}$-interpretations, where $\mathcal{I}$ and all of $\mathcal{W}$ have the same infinite domain Δ with $N_I \subset \Delta$. The interpretation function $\cdot^{\mathcal{I},\mathcal{W}}$ is then defined as follows:

$$\begin{aligned}
a^{\mathcal{I},\mathcal{W}} &= a \quad \text{for } a \in N_I\\
X^{\mathcal{I},\mathcal{W}} &= X^{\mathcal{I}} \quad \text{for } X \in N_C \cup N_R \cup \{\top, \bot\}\\
\{a_1,\dots,a_n\}^{\mathcal{I},\mathcal{W}} &= \{a_1,\dots,a_n\}\\
(\mathbf{K}C)^{\mathcal{I},\mathcal{W}} &= \textstyle\bigcap_{\mathcal{J}\in\mathcal{W}}(C^{\mathcal{J},\mathcal{W}}) \qquad (\mathbf{K}R)^{\mathcal{I},\mathcal{W}} = \textstyle\bigcap_{\mathcal{J}\in\mathcal{W}}(R^{\mathcal{J},\mathcal{W}})\\
(C \sqcap D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \qquad (C \sqcup D)^{\mathcal{I},\mathcal{W}} = C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}}\\
(\neg C)^{\mathcal{I},\mathcal{W}} &= \Delta \setminus C^{\mathcal{I},\mathcal{W}}\\
(\exists R.\mathsf{Self})^{\mathcal{I},\mathcal{W}} &= \{x \mid (x,x) \in R^{\mathcal{I},\mathcal{W}}\}\\
(\exists R.C)^{\mathcal{I},\mathcal{W}} &= \{x \mid \exists y.(x,y) \in R^{\mathcal{I},\mathcal{W}} \land y \in C^{\mathcal{I},\mathcal{W}}\}\\
(\forall R.C)^{\mathcal{I},\mathcal{W}} &= \{x \mid \forall y.(x,y) \in R^{\mathcal{I},\mathcal{W}} \to y \in C^{\mathcal{I},\mathcal{W}}\}\\
(\leqslant nR.C)^{\mathcal{I},\mathcal{W}} &= \{x \mid \#\{y \in C^{\mathcal{I},\mathcal{W}} \mid (x,y) \in R^{\mathcal{I},\mathcal{W}}\} \leq n\}\\
(\geqslant nR.C)^{\mathcal{I},\mathcal{W}} &= \{x \mid \#\{y \in C^{\mathcal{I},\mathcal{W}} \mid (x,y) \in R^{\mathcal{I},\mathcal{W}}\} \geq n\}
\end{aligned}$$

where C and D are $\mathcal{SROIQK}$-concepts and R is a $\mathcal{SROIQK}$-role. $\Diamond$

From the above, one can see that $\mathbf{K}C$ is interpreted as the set of objects that are in the extension of C under every interpretation in $\mathcal{W}$. This also makes clear why the common domain and rigid term assumption have to be imposed; otherwise the respective extension intersections would be empty. Note that the rigid term assumption implies the unique name assumption (UNA), i.e., for any epistemic interpretation $\mathcal{I} \in \mathcal{W}$ and for any two distinct individual names a and b, we have that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

The notions of knowledge base, TBox, RBox and Abox, their respective axioms, and their interpretations can be extended from $\mathcal{SROIQ}$ to $\mathcal{SROIQK}$ in the obvious way.

An *epistemic model* for a $\mathcal{SROIQK}$-knowledge base Σ is a *maximal* non-empty set $\mathcal{W}$ of $\mathcal{SROIQ}$-interpretations such that $(\mathcal{I}, \mathcal{W})$ satisfies Σ for each $\mathcal{I} \in \mathcal{W}$. A $\mathcal{SROIQK}$-knowledge base Σ is said to be *satisfiable* if it has an epistemic model. The knowledge base Σ *(epistemically) entails* an axiom α (written $\Sigma \mid\models \alpha$), if for every epistemic model $\mathcal{W}$ of Σ, we have that for each $\mathcal{I} \in \mathcal{W}$, the epistemic interpretation $(\mathcal{I}, \mathcal{W})$ satisfies α. By definition, every $\mathcal{SROIQ}$-knowledge base is a $\mathcal{SROIQK}$-knowledge base. Note that a given $\mathcal{SROIQ}$-knowledge base Σ has up to isomorphism only one unique epistemic model which is the set of all models of Σ having infinite domain and satisfying the unique name assumption. We denote this model by $\mathcal{M}(\Sigma)$.

3 Problems with the Classical Semantics

Following the intuition that led to the introduction of the $\mathbf{K}$ operator as an extension of $\mathbf{K}$-free standard DL reasoning, a rather intuitive basic requirement to an epistemic DL semantics is arguably the following.

Definition 2. For a given DL $\mathcal{L}$, an epistemic DL semantics represented by an entailment relation $\models\!\!\approx$ is called *$\mathcal{L}$-backward-compatible* if it coincides with the (non-epistemic) standard semantics (represented by $\models$) on non-epistemic axioms, i.e. for an $\mathcal{L}$ knowledge base Σ and an $\mathcal{L}$ axiom α both of which not containing $\mathbf{K}$, we have $\Sigma \models\!\!\approx \alpha$ exactly if $\Sigma \models \alpha$. Moreover, $\models\!\!\approx$ is called *$\mathcal{L}$-UNA-backward-compatible*, if $\Sigma \models\!\!\approx \alpha$ exactly if $\Sigma \models \alpha$ under the unique name assumption. $\Diamond$

We can show that $\mid\models$ is $\mathcal{SRIQ}\backslash U$-UNA-backward-compatible, where $\mathcal{SRIQ}\backslash U$ denotes the description logic $\mathcal{SROIQ}$ without nominal concepts and the universal role. The main ingredient for this is the insight that for any finite interpretation of a given $\mathcal{SRIQ}\backslash U$ knowledge base, we can come up with an infinite interpretation such that both interpretations behave in exactly the same way in terms of satisfaction of axioms.

Lemma 3. *Let Σ be a $\mathcal{SRIQ}\backslash U$ knowledge base. For any interpretation $\mathcal{I}$ there is an interpretation $\mathcal{I}_\omega$ with infinite domain such that $\mathcal{I} \models \Sigma$ if and only if $\mathcal{I}_\omega \models \Sigma$.*

As a consequence, the restriction to infinite models imposed by the common domain assumption turns out to be not essential in the case of $\mathcal{SRIQ}\backslash U$. However, this situation changes drastically once nominals or the universal role are involved. To see this, consider the axioms $\top \sqsubseteq \{a, b, c\}$ or $\top \sqsubseteq \leqslant 3U.\top$. Each of these axioms considered as a knowledge base Σ has only models with at most three elements. Consequently, in both cases we have that Σ is unsatisfiable w.r.t. the classical epistemic semantics and consequently by *ex falso quodlibet* we, e.g., obtain $\Sigma \mathrel{|}\!\models \top \sqsubseteq \bot$ whereas we clearly have $\Sigma \not\models \top \sqsubseteq \bot$ even under the UNA. So we conclude that $\mathrel{|}\!\models$ is not UNA-backward-compatible for any description logic that features nominals or simultaneously number restrictions and the universal role; in particular, it is not $\mathcal{SROIQ}$-UNA-backward-compatible.

While the imposed UNA may be a deliberate decision, we believe that non-$\mathcal{SROIQ}$-UNA-backward-compatibility of classical epistemic entailment is not intended but rather a side effect of a semantics crafted for and probed against less expressive description logics; it contradicts the intuition behind the **K** operator. This motivates our quest for a more appropriate, "domain-flexible" epistemic semantics. In [8], another approach, based on first-order logic (FOL), has been presented which circumvents the described problem by treating the equality as a congruence relation with minimized extension. However, the solution we present is closer to the original DL setting as it extends the standard definition of DL interpretations.

4 A Revisited Semantics

In order to allow for the necessary flexibility, we need to relinquish the common domain assumption and the rigid term assumption in the epistemic semantics: The domains we consider in the possible worlds should be allowed to have arbitrary (yet non-empty) size and be composed of arbitrary elements. An individual name may refer to different elements in different possible worlds. Also, individuals denoted by different individual names may coincide in some worlds but not in others.

Still, due to the reasons discussed before, we have to find a substitute for the common domain and rigid term assumptions as otherwise, every epistemic role or concept would have the empty set as extension.

We solve the problem by introducing one abstraction layer that assigns *abstract individual names* to domain elements. These abstract individual names are elements from $N_I \cup \mathbb{N}$ and hence common to all interpretations, thus they can serve as the "common ground" for different interpretations with different domains. We require that every domain element is associated with at least one abstract name, however, we also allow for different names denoting the same domain element (thus allowing for the possibility of finite domains). This intuition leads to the definition of extended interpretations.

Thereby, an *extended $\mathcal{SROIQ}$-interpretation* $\mathfrak{I}$ is a tuple $(\Delta_\mathfrak{I}, \cdot^\mathfrak{I}, \varphi_\mathfrak{I})$ such that

- $(\Delta_\mathfrak{I}, \cdot^\mathfrak{I})$ is a standard DL interpretation,
- $\varphi_\mathfrak{I} : N_I \cup \mathbb{N} \twoheadrightarrow \Delta^\mathcal{I}$ is a surjective function from $N_I \cup \mathbb{N}$ onto $\Delta^\mathcal{I}$, such that for all $a \in N_I$ we have that $\varphi_\mathfrak{I}(a) = a^\mathfrak{I}$.

Note that the function $\varphi_\mathfrak{I}$ returns the actual interpretation of an individual, given its (abstract) name, under the interpretation $\mathfrak{I}$. We lift $\varphi_\mathfrak{I}$ to sets of names and let $\varphi_\mathfrak{I}^{-1}$ denote the corresponding inverse. Based on the notion of extended interpretations, we define an *extended epistemic interpretation* for $\mathcal{SROIQK}$ as a pair $(\mathfrak{I}, \mathfrak{W})$, where $\mathfrak{I}$ is an extended $\mathcal{SROIQ}$-interpretation and $\mathfrak{W}$ is a set of extended $\mathcal{SROIQ}$-interpretations. Similar to epistemic interpretations, we define an extended interpretation function $\cdot^{\mathfrak{I},\mathfrak{W}}$ as $\cdot^{\mathcal{I},\mathcal{W}}$ in Definition 1 with the following modifications:

$$\begin{aligned}(\mathbf{K}C)^{\mathfrak{I},\mathfrak{W}} &= \varphi_\mathfrak{I}\left(\textstyle\bigcap_{\mathfrak{J}\in\mathfrak{W}} \varphi_\mathfrak{J}^{-1}\left(C^{\mathfrak{J},\mathfrak{W}}\right)\right)\\ (\mathbf{K}R)^{\mathfrak{I},\mathfrak{W}} &= \varphi_\mathfrak{I}\left(\textstyle\bigcap_{\mathfrak{J}\in\mathfrak{W}} \varphi_\mathfrak{J}^{-1}\left(R^{\mathfrak{J},\mathfrak{W}}\right)\right)\end{aligned}$$

Again, we set $[(\mathbf{K}R)^-]^{\mathfrak{I},\mathfrak{W}} := (\mathbf{K}R^-)^{\mathfrak{I},\mathfrak{W}}$ for an epistemic role $(\mathbf{K}R)^-$.

The semantics of TBox, RBox and ABox axioms follows exactly that for the classical semantics. Here, instead of $\Vdash$, we use the symbol $\Vdash_e$, where e indicates that the relation is w.r.t. the extended semantics.

Like in case of the traditional (epistemic) semantics, we can define the notions of *extended epistemic modelhood* and the *satisfiability* of a $\mathcal{SROIQK}$-knowledge base by considering extended interpretations instead of the standard DL interpretations. Similarly, the *entailment* (under the new semantics) of an axiom from a knowledge base can be defined.

We now first note that the newly established semantics has the desired compatibility property.

Theorem 4. *$\Vdash_e$ is $\mathcal{SROIQ}$-backward-compatible.*

Proof sketch: First note that every satisfiable **K**-free knowledge base Σ has exactly one extended epistemic model

$$\mathfrak{M}(\Sigma) = \{(\Delta_{\mathfrak{J}}, \cdot^{\mathfrak{J}}, \varphi_{\mathfrak{J}}) \mid (\Delta_{\mathfrak{J}}, \cdot^{\mathfrak{J}}) \models \Sigma, \varphi\ _{\mathfrak{J}} = \cdot^{\mathfrak{J}}|_{N_I} \cup f, f{:}\mathbb{N} \twoheadrightarrow \Delta_{\mathfrak{J}}\}.$$

Hence we have $\Sigma \mathrel{\|\!\!\models_{\mathrm{e}}} \alpha$ exactly if every $\mathfrak{J} \in \mathcal{M}_{\mathrm{e}}(\Sigma)$ satisfies α, which (presuming α being **K**-free) is the case exactly if $\Sigma \models \alpha$. $\square$

Consequently, this new semantics is more adequate for very expressive DLs such as $\mathcal{SROIQ}$. Yet, as will be shown later in the paper, it is also generic in the sense that for $\mathcal{SRIQ}\backslash U$ knowledge bases it behaves similar to the (classical) epistemic interpretation introduced earlier. With this new semantics, we avoid the aforementioned problems arising from nominals and the universal role in the language of a knowledge base. Arguably, this makes the revisited semantics a more suitable and appropriate choice for **K**-extensions of expressive description logics, like $\mathcal{SROIQK}$.

5 Reducing Epistemic Querying to Standard DL Reasoning

We next introduce a novel technique for answering epistemic queries to $\mathcal{SROIQ}$ knowledge bases under the revised semantics. More precisely, we provide a way of checking whether a given knowledge base entails concept assertions, role assertions or concept subsumptions where the involved concepts and roles may contain **K**. Our method reduces this problem to a number of iterative entailment checks for **K**-free axioms. To justify the translation, we establish two lemmata (c.f., Lemma 25 and Lemma 27 in the technical report) that characterize possible instances of epistemic concepts and roles, respectively. The idea is that the extension of a concept that is preceded by **K** can only contain named individuals unless it comprises the whole domain. For roles we get a more intricate case distinction, however, it boils down to characterizing the set of "(inverse) role neighbors" of a fixed individual as the whole domain or a set of named individuals. As an "exceptional case" to this, we might get the diagonal of $\Delta^{\mathfrak{J}} \times \Delta^{\mathfrak{J}}$ as additional instances of an epistemic role.

Based on these characteristics of epistemic concepts and roles, we present a translation of epistemic concept expressions into equivalent **K**-free ones. Note that the translation itself requires to check entailment of (**K**-free) axioms, hence it is not strictly syntactical and it depends on the underlying knowledge base.

Definition 5. (Translation Function $[\![\cdot]\!]_{\Sigma}$)

Let Σ be a $\mathcal{SROIQ}$-knowledge base. For a $\mathcal{SROIQ}$ concept A and a $\mathcal{SROIQ}$ role R, let $\mathrm{trg}_{\Sigma}^{A,R}$ denote the nominal concept $\{a_1, \ldots, a_n\}$ containing all a_i for which $\Sigma \models A \sqsubseteq \exists R.\{a_i\}$ and let $\mathrm{trg}_{\Sigma}^{A,R} = \bot$ if there are no such a_i. We recursively define the function $[\![\cdot]\!]_{\Sigma}$ mapping $\mathcal{SROIQK}$ concept expressions to $\mathcal{SROIQ}$ concept expressions:

$$
\begin{aligned}
[\![C]\!]_{\Sigma} &= C && \text{if } C \text{ is from } N_I \cup \{\top, \bot\}, \text{ a nominal,} \\
&&& \text{or a } \mathbf{K}\text{-free self concept;} \\
[\![C_1 \sqcap C_2]\!]_{\Sigma} &= [\![C_1]\!]_{\Sigma} \sqcap [\![C_2]\!]_{\Sigma} \\
[\![C_1 \sqcup C_2]\!]_{\Sigma} &= [\![C_1]\!]_{\Sigma} \sqcup [\![C_2]\!]_{\Sigma} \\
[\![\neg C]\!]_{\Sigma} &= \neg [\![C]\!]_{\Sigma} \\
[\![\Xi R.D]\!]_{\Sigma} &= \Xi R.[\![D]\!]_{\Sigma} && \text{for } \Xi \in \{\forall, \exists, \geqslant n, \leqslant n\},\ R\ \mathbf{K}\text{-free} \\
[\![\mathbf{K}D]\!]_{\Sigma} &= \begin{cases} \top & \text{if } \Sigma \models [\![D]\!]_{\Sigma} \equiv \top \\ \{a \in N_I \mid \Sigma \models [\![D]\!]_{\Sigma}(a)\} & \text{otherwise} \end{cases} \\
[\![\exists \mathbf{K} S.\mathsf{Self}]\!]_{\Sigma} &= [\![\mathbf{K} \exists S.\mathsf{Self}]\!]_{\Sigma} \\
[\![\Xi \mathbf{K} R.D]\!]_{\Sigma} &= \Xi R.[\![D]\!]_{\Sigma} && \text{for } \Xi \in \{\forall, \exists, \geqslant n, \leqslant n\} \text{ and } \Sigma \models R \equiv U \\
[\![\forall \mathbf{K} P.D]\!]_{\Sigma} &= \neg [\![\exists \mathbf{K} P.\neg D]\!]_{\Sigma} \\
[\![\exists \mathbf{K} P.D]\!]_{\Sigma} &= \exists P.\big(\mathrm{trg}_{\Sigma}^{\top,P} \sqcap [\![D]\!]_{\Sigma}\big) \sqcup (\mathrm{trg}_{\Sigma}^{\top,P^-} \sqcap \exists P.[\![D]\!]_{\Sigma}) \\
&\quad \sqcup \textstyle\bigsqcup_{a \in N_I} (\{a\} \sqcap \exists P.(\mathrm{trg}_{\Sigma}^{\{a\},P} \sqcap [\![D]\!]_{\Sigma})) \quad \underbrace{\sqcup\ [\![D]\!]_{\Sigma}}_{\text{only if } \Sigma \models \top \sqsubseteq \exists P.\mathsf{Self}} \\
[\![\leqslant n \mathbf{K} P.D]\!]_{\Sigma} &= \neg [\![\geqslant (n+1) \mathbf{K} P.D]\!]_{\Sigma} \\
[\![\geqslant n \mathbf{K} P.D]\!]_{\Sigma} &= \geqslant n P.(\mathrm{trg}_{\Sigma}^{\top,P} \sqcap [\![D]\!]_{\Sigma}) \sqcup (\mathrm{trg}_{\Sigma}^{\top,P^-} \sqcap \geqslant n P.[\![D]\!]_{\Sigma}) \\
&\quad \sqcup \textstyle\bigsqcup_{a \in N_I} (\{a\} \sqcap \geqslant n P.(\mathrm{trg}_{\Sigma}^{\{a\},P} \sqcap [\![D]\!]_{\Sigma}\})) \\
&\quad \underbrace{\sqcup\, (\neg\{a \mid a \in N_I\} \sqcap [\![D]\!]_{\Sigma} \sqcap \geqslant (n-1) P.\big(\mathrm{trg}_{\Sigma}^{\top,P} \sqcap [\![D]\!]_{\Sigma}\big))}_{\text{only if } \Sigma \models \top \sqsubseteq \exists P.\mathsf{Self}}
\end{aligned}
$$

$\Diamond$

Observe that by definition, the result of applying this function to an epistemic concept indeed yields a concept not containing **K**. Moreover the following lemma, which can be proved by structural induction over the concept expression, ensures that the translation function preserves the concept extension.

Lemma 6. *Let Σ be a $\mathcal{SROIQ}$-knowledge base and C be a $\mathcal{SROIQK}$ concept. Then for any extended interpretation $\mathfrak{I} \in \mathfrak{M}(\Sigma)$, we have that $C^{\mathfrak{I},\mathfrak{M}(\Sigma)} = [\![C]\!]_{\Sigma}{}^{\mathfrak{I},\mathfrak{M}(\Sigma)}$.*

Consequently this lemma can be employed to prove our main result justifying our approach of deciding entailment of epistemic axioms based on non-epistemic standard reasoning.

Theorem 7. *For a $\mathcal{SROIQ}$-knowledge base Σ, $\mathcal{SROIQK}$ concept C, D, and an individual a, the following hold:*

1. $\Sigma \mathrel{|\!\!\models_{\mathrm{e}}} C(a)$ *if and only if* $\Sigma \models [\![C]\!]_\Sigma(a)$.
2. $\Sigma \mathrel{|\!\!\models_{\mathrm{e}}} C \sqsubseteq D$ *if and only if* $\Sigma \models [\![C]\!]_\Sigma \sqsubseteq [\![D]\!]_\Sigma$.

Finally, we are also able to establish the correspondence that the classical and the newly introduced semantics coincide, as far as epistemic querying on $\mathcal{SRIQ}\backslash U$ knowledge bases is concerned. This result further substantiates our claim that our semantics is a natural extension of the original intuition behind epistemic DLs.

Theorem 8. *Let* Σ *be a* $\mathcal{SRIQ}\backslash U$ *knowledge base,* C *and* D $\mathcal{SROIQK}$ *concepts, and* a *an individual name. Then, the following hold:*

1. $\Sigma \mathrel{|\!\!\models_{\mathrm{e}}} C(a)$ *under the unique name assumption if and only if* $\Sigma \mathrel{|\!\!\models} C(a)$.
2. $\Sigma \mathrel{|\!\!\models_{\mathrm{e}}} C \sqsubseteq D$ *under the unique name assumption if and only if* $\Sigma \mathrel{|\!\!\models} C \sqsubseteq D$.

This can be proved by providing a transformation function similar to $[\![\cdot]\!]_\Sigma$ for the classical semantics, proving its correctness and showing that it coincides with $[\![\cdot]\!]_\Sigma$ on $\mathcal{SRIQ}\backslash U$ knowledge bases.

6 A System

Based on the results established in the preceding section, we have implemented a preliminary prototype. The system takes an epistemic concept as input and translates it into an equivalent non-epistemic one according to Definition 5. A detailed system description is provided in the technical report. A running system has been uploaded and shared on googlecode[1]. For the purpose of testing, we consider two versions of the wine ontology[2] with 483 and 1127 individuals. As a measure, we consider the translation time of an epistemic concept to a non-epistemic equivalent one and the instance retrieval time of the translated concept. We consider different epistemic concepts. For each such concept C, we consider a non-epistemic concept obtained from C by dropping the $\mathbf{K}$-operators from it (see Table 1). Given a concept C, $\mathsf{t}_{(C)}$ and $|C_i|$ represent the time in seconds required to compute the instances and the number of instances computed for C_i. Finally for an epistemic concept EC_i, $\mathsf{t}_{\mathsf{T}(EC_i)}$ represents the time required to translate EC_i to its non-epistemic equivalent. Table 2 provides our evaluation results. From Table 2, the time

[1] http://code.google.com/p/epistemicdl/
[2] http://www.w3.org/TR/owl-guide/wine.rdf

Table 1. Concepts used for instance retrieval experiments.

C_1	$\exists \mathit{hasWineDescriptor}.\mathit{WineDescriptor}$
EC_1	$\exists \mathbf{K} \mathit{hasWineDescriptor}.\mathbf{K}\, \mathit{WineDescriptor}$
C_2	$\forall \mathit{hasWineDescriptor}.\mathit{WineDescriptor}$
EC_2	$\forall \mathbf{K} \mathit{hasWineDescriptor}.\mathbf{K}\, \mathit{WineDescriptor}$
C_3	$\exists \mathit{hasWineDescriptor}.\mathit{WineDescriptor} \sqcap \exists \mathit{madeFromFruit}.\mathit{WineGrape}$
EC_3	$\exists \mathbf{K} \mathit{hasWineDescriptor}.\mathbf{K}\, \mathit{WineDescriptor} \sqcap \exists \mathbf{K} \mathit{madeFromFruit}.\mathbf{K}\, \mathit{WineGrape}$
C_4	$\mathit{WhiteWine} \sqcap \neg \exists \mathit{locatedIn}.\{\mathit{FrenchRegion}\}$
EC_4	$\mathbf{K}\, \mathit{WhiteWine} \sqcap \neg \exists \mathbf{K} \mathit{locatedIn}.\{\mathit{FrenchRegion}\}$
C_5	$\mathit{Wine} \sqcap \neg \exists \mathit{hasSugar}.\{\mathit{Dry}\} \sqcap \neg \exists \mathit{hasSugar}.\{\mathit{OffDry}\} \sqcap \neg \exists \mathit{hasSugar}.\{\mathit{Sweet}\}$
EC_5	$\mathbf{K}\, \mathit{Wine} \sqcap \neg \exists \mathbf{K} \mathit{hasSugar}.\{\mathit{Dry}\} \sqcap \neg \exists \mathbf{K} \mathit{hasSugar}.\{\mathit{OffDry}\} \sqcap \neg \mathbf{K} \exists \mathit{hasSugar}.\{\mathit{Sweet}\}$

Table 2. Evaluation

Ontology	Concept	$\mathsf{t}_{(C_i)}$	$\|C_i\|$	Concept	$\mathsf{t}_{\mathsf{T}(EC_i)}$	$\mathsf{t}_{(EC_i)}$	$\|EC_i\|$
Wine 1	C_1	2.13	159	EC_1	46.98	0.04	3
	C_2	0.01	483	EC_2	0.18	0.00	0
	C_3	28.90	159	EC_3	79.43	6.52	3
	C_4	0.13	0	EC_4	95.60	107.82	72
	C_5	52.23	80	EC_5	60.78	330.49	119
Wine 2	C_1	8.51	371	EC_1	351.78	0.13	308
	C_2	0.30	1127	EC_2	0.127	0.00	0
	C_3	227.10	371	EC_3	641.24	19.58	7
	C_4	0.34	0	EC_4	865.04	840.97	168
	C_5	295.87	240	EC_5	381.41	2417.65	331

required to compute the number of instances is feasible; it is roughly in the same order of magnitude as for non-epistemic concepts. Note also that the runtime comparison between epistemic concepts EC_i and their non-epistemic counterparts C_i should be taken with a grain of salt as they are semantically different in general, as also indicated by the fact that there are cases where retrieval for the epistemic concept takes less time than for the non-epistemic version. As a general observation, we noticed that instances retrieval for an epistemic concept where a **K**-operator occurs within the scope of a negation, tends to require much time.

7 Conclusion and Outlook

We argued how the traditional semantics for epistemic DLs causes problems and thus suggested a revision to the semantics. We proved that this revised semantics solves the aforementioned problem while coinciding with the traditional semantics on less expressive DLs (up to $\mathcal{SRIQ}\backslash U$). Focusing on the new semantics, we provided a way of answering epistemic queries to $\mathcal{SROIQ}$ knowledge bases via a reduction to a series of standard reasoning steps. Finally, we presented an implementation allowing for epistemic querying in $\mathcal{SROIQ}$.

Avenues for future research include the following: First, we will investigate to what extent the methods described here can be employed for entailment checks on $\mathcal{SROIQK}$ knowledge bases, i.e., in cases where $\mathbf{K}$ occurs inside the knowledge base. In that case, stronger non-monotonic effects occur and the unique-epistemic-model property is generally lost. On the more practical side, we aim at further developing our initial prototype. We are confident that by applying appropriate optimizations such as caching strategies and syntactic query preprocessing a significant improvement in terms of runtime can be achieved. In the long run, we aim at demonstrating the added value of epistemic querying by providing an appropriate user-front-end and performing user studies. Furthermore, we will propose an extension of the current OWL standard by epistemic constructs in order to provide a common ground for future applications.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Donini, F., Nardi, D., Rosati, R.: Non-first-order features in concept languages. In: Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA'95). pp. 91–102. Springer (1995)
3. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: Adding epistemic operators to concept languages. In: Bernhard Nebel, Charles Rich, W.R.S. (ed.) Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). pp. 342–353. Morgan Kaufmann (1992)
4. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: An epistemic operator for description logics. Artificial Intelligence 100(1-2), 225–274 (1998)
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
6. Levesque, H.J.: Foundations of a functional approach to knowledge representation. Artificial Intelligence 23(2), 155–212 (1984)
7. Mehdi, A., Rudolph, S.: Revisiting semantics for epistemic extensions of description logics. Technical Report 3015, Institute AIFB, Karlsruhe Institute of Technology (2011), available at http://www.aifb.kit.edu/web/Techreport3015/en
8. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)
9. Reiter, R.: What should a database know? Journal of Logic Programming 14(1-2), 127–153 (1992)

Implementing completion-based inferences for the $\mathcal{EL}$-family

Julian Mendez and Andreas Ecke and Anni-Yasmin Turhan

TU Dresden, Institute for Theoretical Computer Science

Abstract. Completion algorithms for subsumption are investigated for many extensions of the description logic $\mathcal{EL}$. While for several of them subsumption is tractable, this is no longer the case, if inverse roles are admitted. In this paper we present an optimized version of the completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ [11], which is implemented in jCel. The completion sets computed during classification are a good substrate for implementing other reasoning services such as generalizations. We report on an extension of jCel that computes role-depth bounded least common subsumers and most specific concepts based on completion sets.

1 Introduction

The lightweight Description Logic (DL) $\mathcal{EL}$ and many of its extensions enjoy the nice property that computing concept subsumption and classification of ontologies written in these Description Logics is tractable [1]. Prominent bio-medical ontologies are expressed in extensions of $\mathcal{EL}$ for which reasoning can still be done in polynomial time. The Gene ontology (GO) is an $\mathcal{ELH}$ ontology and SNOMED is written in $\mathcal{EL}^+$. However, the GALEN ontology uses the DL $\mathcal{ELHI}f_{\mathcal{R}^+}$—a DL with inverse roles, which are known to make subsumption w.r.t. general ontologies ExpTime-complete [2]. While the polynomial time completion algorithms work on graph structures that are static and have simple labellings, the algorithm for $\mathcal{ELI}$ requires dynamic nodes sets and uses complex labels. In [11] a completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ has been devised. Since the node set generated by this method can grow exponentially, it is important to use a good completion strategy, that determines the next node label to which a completion rules is applicable. We present in this paper an optimized version of the algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ with such a completion strategy, which is implemented in the reasoner jCel.

Recently, the completion sets computed during classification have been employed to compute (approximations for) generalization inferences such as the least common subsumer (lcs) or most specific concept (msc). The lcs generalizes a collection of concept descriptions into a single concept description that is the least w.r.t. subsumption. The msc generalizes a description of an individual into a concept description. Intuitively, the msc delivers the most specific concept description that the input individual belongs to. Both of these services are useful for the building of knowledge bases. In the bio-medical field in particular the lcs is employed to define similarity measures between concept descriptions. Since for

	Syntax	Semantics
conjunction	$C \sqcap C$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restr.	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
role inclusion	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
functional role	$f(r)$	$\forall d_1 \in \Delta_{\mathcal{I}} : \mid \{d_2 \in \Delta_{\mathcal{I}} \mid (d_1,d_2) \in r^{\mathcal{I}}\} \mid \leq 1$
inverse role	r^-	$\{(d_1,d_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d_2,d_1) \in r^{\mathcal{I}}\}$
transitive role	$r \circ r \sqsubseteq r$	$\{(d_1,d_2),(d_2,d_3)\} \subseteq r^{\mathcal{I}} \to (d_1,d_3) \subseteq r^{\mathcal{I}}$

Table 1. $\mathcal{ELHI}f_{\mathcal{R}^+}$-concept and role constructors.

general $\mathcal{EL}$-TBoxes neither the lcs nor the msc need to exist, an algorithm for role-depth bounded lcs and -msc was devised in [8]. These algorithms are now implemented for $\mathcal{ELH}$ on top of jCel.

2 Preliminaries

Starting from two disjoint sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept* and *role names*, respectively, $\mathcal{ELHI}f_{\mathcal{R}^+}$*-concept descriptions* are built using concept and role constructors shown in Table 1 and the *top-concept* ($\top$). The DL $\mathcal{EL}$ is the $\mathcal{ELHI}f_{\mathcal{R}^+}$-fragment that only allows for the concept constructors conjunction and existential restrictions. $\mathcal{ELH}$ extends $\mathcal{EL}$ by role inclusion statements.

The semantics of $\mathcal{ELHI}f_{\mathcal{R}^+}$ is defined by interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concepts. The interpretation function is extended to complex concept descriptions and roles as described in the last column of Table 1.

A TBox is a set of *concept inclusion axioms* of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ *satisfies* the concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. A concept C is *subsumed by* a concept D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$) if, for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$.

Let $\mathsf{N_I}$ be a set of individual names. An $\mathcal{EL}$*-ABox* is a set of assertions of the form $C(a), r(a,b)$, where C is an $\mathcal{EL}$-concept description, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Finally, an individual $a \in \mathsf{N_I}$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ (written $\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. *ABox realization* is to compute for each individual a in $\mathcal{A}$ the set of named concepts from $\mathcal{K}$ that have a as an instance.

3 Completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$

Classification of TBoxes is the computation of all subsumption relations between all named concepts of a TBox. For several extensions of $\mathcal{EL}$ classification can be performed in polynomial time [1, 2]. These classification algorithms typically proceed in three steps:

NR-1	$C \equiv D$	$\rightsquigarrow$	$C \sqsubseteq D, D \sqsubseteq C$
NR-2	$C_1 \sqcap \cdots \sqcap \hat{C} \sqcap \cdots \sqcap C_n \sqsubseteq D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, C_1 \sqcap \cdots \sqcap A \sqcap \cdots \sqcap C_n \sqsubseteq D$
NR-3	$\exists r'.\hat{C} \sqsubseteq D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, \exists r'.A \sqsubseteq D$
NR-4	$\hat{C} \sqsubseteq \exists r'.D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A, A \sqsubseteq \exists r'.D$
NR-5	$B \sqsubseteq \exists r'.\hat{C}$	$\rightsquigarrow$	$B \sqsubseteq \exists r'.A, A \sqsubseteq \hat{C}$
NR-6	$D \sqsubseteq C_1 \sqcap C_2$	$\rightsquigarrow$	$D \sqsubseteq C_1, D \sqsubseteq C_2$
NR-7	$C \sqsubseteq \exists r^-.D$	$\rightsquigarrow$	$C \sqsubseteq \exists u.D$, $u \sqsubseteq r^-$, $r^- \sqsubseteq u$
NR-8	$\exists r^-.C \sqsubseteq D$	$\rightsquigarrow$	$\exists u.C \sqsubseteq D$, $u \sqsubseteq r^-$, $r^- \sqsubseteq u$

where r: role; r': (inverse) role; C, C_i, D: concept descriptions; $\hat{C}$, $\hat{D}$: complex concept descriptions; B: concept name; A: *fresh* concept name; u: *fresh* role name.

Table 2. Normalization rules.

1. normalization of the TBox
2. apply completion rules to the *completion graph*
3. read off subsumptions relations from the saturated completion graph

The basic completion algorithm represents the completion graph by two kinds of *completion sets*: $S(C)$ and $S(C, r)$ for each concept name C and role name r from the TBox. The sets contain concept names from the TBox and $\top$. The sets $S(C)$ represent the labelled nodes, while the sets $S(C, r)$ represent the edges of the completion graph. The idea of the classification algorithm is that completion rules make implicit subsumption relationships explicit. In fact, the following invariants hold:

– $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
– $D \in S(C, r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.

For extensions of $\mathcal{EL}$ that also offer inverse roles, testing subsumption is not polynomial, but it is ExpTime-complete [2]. In [11] Vu has devised a completion algorithm for $\mathcal{ELHI}f_{\mathcal{R}^+}$ (and some of its sublanguages). In contrast to the basic completion algorithm, this one works on completion graphs with more complex nodes. Moreover, the set of nodes grows *dynamically* during completion. We describe now an optimized version of Vu's algorithm given in [7].

Normalization. An $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBox $\mathcal{T}$ is in *normal form* if all concept inclusions have one of the following forms, where A_1, A_2, B are concept names:

$$A_1 \sqsubseteq B, \quad A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B, \quad A_1 \sqsubseteq \exists r.A_2 \quad \text{or} \quad \exists r.A_1 \sqsubseteq B.$$

Each $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBox can be transformed into this normal form by applying the rules shown Table 2, where the axioms on the left-hand side are replaced by the axiom(s) on the right-hand side. The implicit information on (functional) roles is made explicit by applying the following *saturation rules* to the TBox:

$$\begin{array}{rcl@{\qquad}rcl} r \sqsubseteq s & \leadsto & r^- \sqsubseteq s^- & r \sqsubseteq s \;,\; s \sqsubseteq t & \leadsto & r \sqsubseteq t \\ r \circ r \sqsubseteq r & \leadsto & r^- \circ r^- \sqsubseteq r^- & r \sqsubseteq s \;,\; f(s) & \leadsto & f(r) \end{array}$$

In addition, auxiliary role names are added to $\mathsf{N_R}$ to allow a mapping where each role s has an inverse role r^- such that $s \equiv r^-$. In this way, the algorithm applies the completion rules to role names and inverse roles.

Completion rules. Once the TBox is normalized and saturated, the completion sets are initialized and the completion rules are applied. Based on the two sets $\varXi := \{\exists r.A \mid r \in \mathsf{N_R}, A \in \mathsf{N_C}\}$ and $\varOmega := \{(A,\psi) \mid A \in \mathsf{N_C}, \psi \subseteq \varXi\}$ the completion sets are defined as

- $V \subseteq \varOmega$
- $S \subseteq \{(x, A) \mid x \in \varOmega, A \in \mathsf{N_C}\}$
- $R \subseteq \{(r, x, y) \mid r \in \mathsf{N_R}, x, y \in \varOmega\}$.

For the completion graph, the set V is the set of nodes, S is a node labeling and $\varOmega$ is the set of edges. The elements in S are called *S-entries*, the elements in R are called *R-entries*, the elements in V are referred to as *nodes*. The completion process satisfies the following *invariants*:

- if $((A,\varphi), C) \in S$, then $(A \sqcap \bigsqcap_{E\in\varphi} E) \sqsubseteq_{\mathcal{T}} C$
- if $(r, (A,\varphi), (B,\psi)) \in R$, then $(A \sqcap \bigsqcap_{E\in\varphi} E) \sqsubseteq_{\mathcal{T}} \exists r.(B \sqcap \bigsqcap_{E\in\psi} E)$

where each E is of the form $\exists r.X$. Furthermore, after completion we have that $A \sqsubseteq_{\mathcal{T}} B$ if and only if $((A, \emptyset), B) \in S$. The algorithm initializes the sets as follows:

- $V := \{(A, \emptyset) \mid A \in \mathsf{N_C}\}$,
- $S := \{((A, \emptyset), A) \mid A \in \mathsf{N_C}\} \cup \{((A, \emptyset), \top) \mid A \in \mathsf{N_C}\}$,
- $R := \emptyset$

and applies the completion rules. The optimized completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$ are presented in Table 3. The underlined elements are membership checks for S and R. These conditions are relevant for the strategy of completion.

As a consequence of the normal form presented here, CR-2 may have several conjuncts on the left-hand side of a normalized GCI. This simple optimization reduces the number of auxiliary symbols.

In [7] it was shown that the rules in Table 3 are equivalent to those in [11].

Completion strategy. The completion rules do not define any order of application to the S- and R entries. In fact, finding the element of the completion sets and axioms from the TBox to which a rule is applicable fast is crucial for the performance of the reasoner. The idea of *sets* is that it collects newly generated entries, which are not present in the set yet, to be tested for applicability of completion rules. This approach has already been employed in Cel, see [**?**]. In Cel each node has an associated queue with entries to be tested. This idea is now transferred to dynamic node sets and the set of completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$.

To prepare Q the initialization of the algorithm is slightly modified:

```
CR-1 if $A \sqsubseteq B \in \mathcal{T}$, $\underline{(x, A) \in S}$ then $S' := S \cup \{(x, B)\}$

CR-2 if $A_1 \sqcap \ldots \sqcap A_i \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$,
       $(x, A_1) \in S$, ..., $\underline{(x, A_i) \in S}$, ..., $(x, A_n) \in S$
     then $S' := S \cup \{(x, B)\}$

CR-3 if $A \sqsubseteq \exists r.B \in \mathcal{T}$, $\underline{(x, A) \in S}$
     then if $f(r)$
       then $v := (\top, \{\exists r^-.A\})$
         if $v \notin V$ then $V := V \cup \{v\}$,  $S' := S \cup \{(v, B)\} \cup \{(v, \top)\}$,
                              $R' := R \cup \{(r, x, v)\}$
       else $y := (B, \emptyset)$
         $R' := R \cup \{(r, x, y)\}$

CR-4 if $\exists s.A \sqsubseteq B \in \mathcal{T}$, $\underline{(r, x, y) \in R}$, $\underline{(y, A) \in S}$, $r \sqsubseteq_{\mathcal{T}} s$
     then $S' := S \cup \{(x, B)\}$

CR-5 if $s \circ s \sqsubseteq s \in \mathcal{T}$, $\underline{(r_1, x, y) \in R}$, $\underline{(r_2, y, z) \in R}$, $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$
     then $R' := R \cup \{(s, x, z)\}$

CR-6 if $\exists s^-.A \sqsubseteq B \in \mathcal{T}$, $r \sqsubseteq_{\mathcal{T}} s$, $\underline{(r, x, y) \in R}$, $\underline{(x, A) \in S}$, $(y, B) \notin S$, $y = (B', \psi)$
     then $v := (B', \psi \cup \{\exists r^-.A\})$
       if $v \notin V$ then $V := V \cup \{v\}$,   $S' := S \cup \{(v, k) \mid (y, k) \in S\}$
       $S' := S \cup \{(v, B)\}$,  $R' := R \cup \{(r, x, v)\}$

CR-7 if $\exists s^-.A \sqsubseteq B \in \mathcal{T}$, $\underline{(r_2, x, y) \in R}$, $x = (A', \varphi)$, $y = (B', \psi)$,
       $r \circ r \sqsubseteq r \in \mathcal{T}$, $r_1 \sqsubseteq_{\mathcal{T}} r$, $r_2 \sqsubseteq_{\mathcal{T}} r$, $\exists r_1^-.A \in \varphi$, $r \sqsubseteq_{\mathcal{T}} s$
     then $v := (B', \psi \cup \{\exists r^-.A\})$
       if $v \notin V$ then $V := V \cup \{v\}$,  $S' := S \cup \{(v, k) \mid (y, k) \in S\}$
       $S' := S \cup \{(v, B)\}$,  $R' := R \cup \{(r_2, x, v)\}$

CR-8 if $A \sqsubseteq \exists r_2^-.B \in \mathcal{T}$, $\underline{(r_1, x, y) \in R}$, $\underline{(y, A) \in S}$,  $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$, $f(s^-)$
     then $S' := S \cup \{(x, B)\}$

CR-9 if $\underline{(r_1, x, y) \in R}$, $(r_2, x, z) \in R$, $r_1 \sqsubseteq_{\mathcal{T}} s$, $r_2 \sqsubseteq_{\mathcal{T}} s$,
       $y = (\top, \psi)$, $z = (\top, \varphi)$, $y \neq z$, $f(s)$
     then $v := (\top, \psi \cup \varphi)$
       if $v \notin V$ then $V := V \cup \{v\}$
       $S' := S \cup \{(v, k) \mid (y, k) \in S\} \cup \{(v, k) \mid (z, k) \in S\}$,  $R' := R \cup \{(r_1, x, v)\}$
```

Table 3. Optimized completion rules for $\mathcal{ELHI}f_{\mathcal{R}^+}$.

- $S := \emptyset$, $R := \emptyset$
- $Q := \{((A, \emptyset), A) \mid A \in \mathsf{N_C}\} \cup \{((A, \emptyset), \top) \mid A \in \mathsf{N_C}\}$

The sets S' and R' represent the next step of sets S and R, respectively. Their new elements are added to Q' in the algorithm shown in Table 4.

We say a completion rule is *sensitive* to changes in a set, if the precondition of that rule mentions that set. In Table 3 the relevant entries are underlined. For example, CR-1 is sensitive to changes in S only, CR-7 is sensitive to changes

```
1. $S, R, Q := \emptyset$
2. **for each** concept name $A$, add $((A, \emptyset), A)$ and $((A, \emptyset), \top)$ to $Q$
3. **while** $Q \neq \emptyset$
4.     take one element $e$ from $Q$ and remove it from $Q$
5.     **if** $e$ is an $S$-entry
6.         let $Q'$ be the result of applying all the $S$-rules to $e$
7.     **else if** $e$ is an $R$-entry
8.         let $Q'$ be the result of applying all the $R$-rules to $e$
9.     $Q := Q \cup ((Q' \setminus S) \setminus R)$
```

Table 4. General algorithm.

in R only, and CR-4 is sensitive to changes in S and R. According to the kind of entry they are sensitive to, the completion rules are members of the *chain of rules* the process S-entries or R-entries.

A conceptual scheme of the algorithm is presented in Table 4. The processor takes entries from Q, changes sets S and R, and informs the corresponding chain of rules of these changes. This procedure is repeated until Q is empty, i.e. no rules are applicable.

3.1 Implementation in jCel

jCel[1] is implemented in Java. The object-oriented design of the completion algorithm brings a very low coupling, since each rule can be changed separately. Thus jCel can easily be adapted to new sets of completion rules. For implementation-dependant technical details (e.g. data structures) see [7].

Besides classification for $\mathcal{ELHI}f_{\mathcal{R}^+}$-TBoxes, jCel also implements realization of $\mathcal{ELH}$-ABoxes.

3.2 Experiments with jCel

The experiments were run on a computer with two Intel(R) Core(TM)2 Duo E8500 processors running at 3.16 GHz and 4 GB of main memory.

Experiments classifying $\mathcal{ELHI}f_{\mathcal{R}^+}$ ontologies. The full version of GALEN is still one of the most challenging ontologies, since hardly any reasoner can classify it. Two GALEN ontologies were considered: the original version of GALEN (GALEN-A), and the newer version of GALEN (GALEN-B), which were used in [?] to test Cel. Table 6 lists their sizes in terms of concepts etc.

For GALEN-A, jCel took 1093 s and the reasoner CB less than 1 s. In case of GALEN-B, the current version of jCel could not finish classification due to lack of memory. CB classified this ontology in 5 s.

[1] The reasoner jCel and its source code is available at http://jcel.sourceforge.net.

ontology	#axioms	#norm. ax.	#concepts	#roles
GALEN-A	8140	12930	2748	413
GALEN-B	61787	95789	23143	950

Table 5. Ontologies using $\mathcal{ELHI}f_{\mathcal{R}^+}$.

ontology	logic	#axioms	#norm. ax.	#concepts	#roles
NCI	$\mathcal{EL}$	74662	47080	27652	70
GO	$\mathcal{EL}_{\mathcal{R}^+}$	49363	28900	20465	1
FMA	$\mathcal{EL}_{\mathcal{R}^+}$	150282	119570	75139	2
SNOMED CT	$\mathcal{ELH}$	962796	1127193	378569	61
NotGalen	$\mathcal{ELH}_{\mathcal{R}^+}$	7540	15089	2748	413
CELGalen	$\mathcal{ELH}_{\mathcal{R}^+}$	60637	102742	23141	950

Table 6. Ontologies using $\mathcal{ELH}_{\mathcal{R}^+}$.

ontology	entries	jCel 0.13.0	Cel Plug-in 0.5.0	quotient
NCI	346887	8.9 s	10.2 s	0.87
GO	154489	4.4 s	3.5 s	1.26
FMA	9576858	149.0 s	2388.0 s	0.06
SNOMED CT	143039451	1108.0 s	705.0 s	1.57
NotGalen	224565	2.9 s	5.2 s	0.56
CELGalen	6836237	52.0 s	134.0 s	0.39

Table 7. Compared times of classification between jCel and Cel.

Experiments in $\mathcal{ELH}_{\mathcal{R}^+}$. In Table 6 we compare the sizes of the different test ontologies to be classified with the polynomial completion algorithm (with static node set).

The execution times of jCel were compared with the Cel system. Cel is one of the fastest reasoners for reasoning in the $\mathcal{EL}$-family of DLs and is known to deliver correct results [6, 4]. The inferred concept hierarchy was identical in classifications of both reasoners. The measured run-times are shown in Table 7.

To sum up, jCel's performance is comparable to state of the art reasoners and, in case of Cel sometimes even better.

4 Completion based generalization

The classification and the realization algorithm of jCel can be employed to compute generalizations. We define these inferences now.

Definition 1. *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{ELH}$-KB and $C_1, \ldots, C_n$ $\mathcal{ELH}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{ELH}$-concept description C is the* role-depth bounded $\mathcal{ELH}$-least common subsumer *of $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$ and role-depth k (written k-lcs$(C_1, \ldots, C_n)$) i ff*

1. *role-depth$(C) \leq k$,*
2. *$C_i \sqsubseteq_{\mathcal{T}} C$ for all $1 \leq i \leq n$, and*

3. *for each $\mathcal{ELH}$-concept description D with role-depth$(D) \leq k$ it holds that, $C_i \sqsubseteq_\mathcal{T} D$ for all $1 \leq i \leq n$ implies $C \sqsubseteq_\mathcal{T} D$.*

Let a be an individual in $\mathcal{A}$ and again $k \in \mathbb{N}$. The $\mathcal{ELH}$-concept description C is the role-depth bounded $\mathcal{ELH}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-msc(a)) i ff*

1. *role-depth$(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for each $\mathcal{ELH}$-concept description D with role-depth$(D) \leq k$ holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_\mathcal{T} D$.*

Completion-based subsumption algorithms classify $\mathcal{ELH}$-TBoxes by explicitly deriving all subsumptions relationships between named concept and storing them in completion sets. The latter can be used to compute the *k-lcs* of concept descriptions. A completion-based realization algorithm can be used to compute the *k-msc* of an individual from its completion sets.

The algorithm for computing *k-lcs* and *k-msc* from completion sets is given in [8]. The idea for *k-lcs* algorithm is: the lcs for two $\mathcal{EL}$-concept descriptions (w.r.t. an empty TBox) can be computed as the product of their corresponding description trees [3]. However, with respect to a general TBox, we can construct the *k-lcs* of two $\mathcal{ELH}$-concept descriptions as follows:

1. assign the input concept descriptions new names
2. classify the augmented TBox
3. for the subgraph of the completion graph reachable from the nodes representing the newly introduced names by paths of length $\leq k$: do cross-product construction w.r.t. the node labels and edges.

The proof of the correctness for the *k-lcs*-algorithm for relies on the invariants discussed in Section 3.

If the completion sets for ABox realization are computed, one can compute the *k-msc* of an individual a simply by traversing the subgraph of the completion graph reachable from a by paths of length up to k and conjoining the node labels.

Since the completion sets are containing *all* subsumers of a named concept, the concept descriptions resulting from traversing subgraphs of the completion graph and collecting the node labels are very redundant. For a person editing the resulting concept description this is clearly undesirable. We devise a simplification heuristic that is similar to the (equivalent) minimal rewritings proposed in [3] for $\mathcal{EL}$-concept descriptions. For general TBoxes the Algorithm 1 yields equivalent and smaller, but not necessarily minimal concept descriptions.

Implementation of the generalization inferences in Gel. Our system Gel implements in Java the methods presented here. Gel accesses jCel's internal data structures directly to compute the *k-lcs* or the *k-msc*. These two reasoning methods and the above described simplification are implemented in Gel in a straight-forward way.

Algorithm 1 Simplification of the resulting concept description.

```
Procedure simplify $(C, S)$
Input: $C$: $\mathcal{EL}$ concept description; $S$: set of completion sets
Output: simplify($C$): a simplified concept description equivalent to $C$
 1: Let $C$ be of the form $A_1 \sqcap \ldots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \ldots \sqcap \exists r_m.D_m$ with $A_i \in N_C$
 2: $Conj := \{A_i \mid i \in \{1, \ldots, n\}\}$
 3: $ExRes := \{\exists r_j.D_j \mid j \in (1 \ldots m)\}$
 4: for all $A_i, E$ with $A_i \in Conj$ and $E \in Conj \cup ExRes$ do
 5:     if $E \neq A_i$ and $E \sqsubseteq_{\mathcal{T}} A_i$ then
 6:         $R := Conj \setminus \{A_i\}$
 7:     end if
 8: end for
 9: for all $\{E, D\} \subseteq ExRes$ do
10:     if $E \neq D$ and $E \sqsubseteq_{\mathcal{T}} D$ then
11:         $R := R \cup (ExRes \setminus \{D\})$
12:     end if
13: end for
14: for all $\exists r_j.D_j \in R$ do
15:     $R := (R \setminus \{\exists r_j.D_j\}) \cup \{\exists r_j.\text{simplify}(D_j, S)\}$
16: end for
17: return $\bigsqcap_{E \in R} E$
```

Our system Gel is available as a plug-in for the ontology editor Protégé and an API for the role-depth bounded lcs and -msc is planned. The former system sonic [10] implemented the lcs and msc as well, but allowed only for acyclic, unfoldable TBoxes.

Evaluation. For the evaluation of the generalization algorithms, we used two different ontologies. The earlier mentioned NotGalen described in Table 6 is a version of the medical ontology Galen stripped-down to $\mathcal{ELH}$. This ontology does not contain individuals, but its deep concept hierarchy makes it a good test ontology for the *k-lcs*. As test concepts for the *k-lcs* we selected sibling concepts from the concept hierarchy with common ancestors other than $\top$ and with many (comparable) existential restrictions. In total, we selected 20 such concept tuples from NotGalen.

We also used the Sweet[2] ontology, the *Semantic Web for Earth and Environmental Terminology* by NASA. This ontology was converted to $\mathcal{ELH}$ by replacing all value restrictions with existential restrictions and dropping all axioms not expressible in $\mathcal{ELH}$. Sweet does contain individuals and a rich relational structure and was used as a test ontology for the *k-msc*. It has 4276 concept names and 2069 individuals. We selected those individuals from Sweet that appear in many role assertions. In total, we selected 18 individuals from Sweet.

All tests were run on an Intel(R) Core(TM) i5-2400 under Oracle Java 6SE 64bit. For each computation of the *k-lcs* or *k-msc* we measured the concept size

[2] http://sweet.jpl.nasa.gov/sweet/

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
construction time (ms)	19	572	3567	7604	289778
simplification time (ms)	< 1	3	15	40	107
expanded concept size	185	3458	15478	33667	119296
simplified concept size	5	15	27	38	42

Table 8. Average concept size and run-time for the *k-lcs* of concepts from NotGalen.

	k	2-ary lcs	3-ary lcs	4-ary lcs	5-ary lcs
construction time (ms)	1	1	7	37	114
	2	23	249	972	3752
	3	284	1954	4465	24427
	4	1801	5253	8355	45374
	5	7008	12412	114452	2665440
expanded concept size	1	178	214	199	217
	2	3608	5974	2171	2313
	3	14198	26997	12859	15300
	4	35656	34958	25793	31634
	5	104768	133089	123924	178831

Table 9. Average concept size and run-time for the *k-lcs* of n concepts from NotGalen.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
construction time (ms)	< 1	< 1	1	2	3
simplification time (ms)	< 1	< 1	< 1	< 1	1
expanded concept size	100	275	498	918	2261
simplified concept size	8	9	10	10	11

Table 10. Average concept size and run-time for the *k-msc* of individuals from Sweet.

of the resulting concept description and after simplification and the run-time (after classification / realization) for construction of the *k-lcs* or *k-msc* and of its simplification. The Table 8 and 10 show the results for the *k-msc* and *k-lcs*, respectively. The concept construction time and expanded concept size for different numbers of input concepts to the *k-lcs* are shown in Table 9.

For the *k-lcs* the resulting run-times were quite high, whereas classification of NotGalen took only around 670 ms. Computation of the *k-msc* was always quite fast—especially compared to the realization time of 5.7 s for the Sweet ontology.

For both *k-lcs* and *k-msc* we found the expanded concept size (and thus the construction time) to grow exponentially with the role-depth bound k. The concept size of the simplified concepts, however, is growing much slower.

Interestingly, for the *k-msc* the resulting concept was the *exact* most specific concept for most individuals for a role-depth of only 2 or 3 — the resulting concept did not change for higher k. Only 3 of the 18 individuals had a msc with maximum role-depth of 5.

Table 9 shows how the run-time of the *k-lcs* grows drastically with the number of input concepts, whereas the concept size stays more or less constant. This is the case because the product construction in the *k-lcs* is more expensive for higher n. Instead of directly computing the *k-lcs* for n concepts, one can also apply the binary *k-lcs* to the first two concepts and then successively compute the *k-lcs* of the result with the next concept. Surprisingly, the accumulated construction time for this method to yield the 4-lcs of 5 concepts was in average 1043.8 ms—much faster than the direct computation time of around 45 s.

To sum up, the computation of the fully expanded concept description is very time-consuming. This is especially true for the product construction of the *k-lcs*. To apply some of the simplification steps already during the construction of the result should help the generalization algorithms to scale better.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proc. of the OWLED Workshop*, 2008.
3. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, 1999.
4. K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web Journal*, pages 1–17, 2011. DOI: 10.3233/SW-2011-0034.
5. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-09)* 2009.
6. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In *Proc. of the 2009 Description Logic Workshop (DL 2009)*, vol. 477 of *CEUR*, 2009.
7. J. Mendez. A classification algorithm for $\mathcal{ELIH}f_{\mathcal{R}+}$. Master's thesis, Technische Universität Dresden, 2011.
8. R. Peñaloza and A.-Y. Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In *Proc. of the 8th European Semantic Web Conf. (ESWC'11)*, LNCS. Springer, 2011.
9. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.* PhD thesis, Technische Universität Dresden, 2009.
10. A.-Y. Turhan and C. Kissig. Sonic — Non-standard inferences go OilEd. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-04)*, vol. 3097 of *LNCS*. Springer, 2004.
11. Q. H. Vu. Subsumption in the description logic $\mathcal{ELHI}fR^+$ w.r.t. general TBoxes. Master's thesis, Technische Universität Dresden, 2008.

Uniform Interpolation in General $\mathcal{EL}$ Terminologies

Nadeschda Nikitina

Karlsruhe Institute of Technology, Karlsruhe, Germany
nikitina@kit.edu

Abstract. Recently, different forgetting approaches for knowledge bases expressed in different logics were proposed. For $\mathcal{EL}$ terminologies containing each atomic concept at most once on the left-hand side, an approach has been proposed based on sufficient, but not necessary acyclicity conditions. In this paper, we propose an approach for computing a uniform interpolant for general $\mathcal{EL}$ terminologies. We first show that a uniform interpolant of any $\mathcal{EL}$ terminology w.r.t. any signature always exists in $\mathcal{EL}$ enriched with least and greatest fixpoint constructors and show how it can be computed by reducing the problem to the computation of Most General Subconcepts and Most Specific Superconcepts for atomic concepts. Then, we give the exact conditions for the existence of a uniform interpolant in $\mathcal{EL}$ and show how it can be obtained using our algorithms.

1 Introduction

The importance of non-standard reasoning services supporting knowledge engineers in modelling a particular domain or in understanding existing models by visualizing implicit dependencies between concepts and roles was pointed out by the research community [3], [5]. An example of such reasoning services supporting knowledge engineers in different activities is the uniform interpolation. In particular for the understanding and the development of complex knowledge bases, e.g., those consisting of *general concept inclusions* (GCIs), the appropriate tool support would be beneficial. However, the existing approach [7] to uniform interpolation in $\mathcal{EL}$ is restricted to terminologies containing each atomic concept at most once on the left-hand side of concept inclusions and additionally satisfying sufficient, but not necessary acyclicity conditions. Lutz et al.[10] propose an approach to uniform interpolation in expressive description logics such as $\mathcal{ALC}$ w.r.t. general terminologies by encoding $\mathcal{ALC}$ TBoxes as concepts, which, however does not solve the problem of uniform interpolation in $\mathcal{EL}$.

Clearly, the existence of the results for such reasoning problems is closely related to the notion of fixpoint semantics. For instance, Baader [2] shows that the structurally similar problems of computing Least Common Subsumer and Most Specific Concept can always be solved in cyclic classical TBoxes w.r.t. to greatest fixpoint semantics. Similar results were obtained for general $\mathcal{EL}$ TBoxes with descriptive semantics[9] , however extended with the greatest fixpoint constructor ($\mathcal{EL}_\nu$). In this paper, we extend the above results by showing that uniform interpolants preserving all $\mathcal{EL}$ consequences of general $\mathcal{EL}$ terminologies w.r.t. an arbitrary signature can always be expressed in an extension of $\mathcal{EL}$ with least fixpoint and greatest fixpoint constructors μ,ν as well as the disjunction used only on the left-hand side of concept inclusions. We extend the previous approach [7] and propose the algorithms for computing such uniform interpolants for general $\mathcal{EL}$ terminologies based on the notion of *most general subconcepts* and *most specific superconcepts*.

In the usual application scenarios it is rather useful to obtain uniform interpolants expressed in the DL of the original terminology instead of introducing additional language constructs. Therefore, in addition to the above algorithms, we derive the existence criteria for uniform interpolants in $\mathcal{EL}$ (i.e., expressed without the above extension) and show how such a uniform interpolant can be obtained using our algorithms. Full proofs are available in the extended version of this paper [11].

2 Preliminaries

Let N_C and N_R be countably infinite and mutually disjoint sets of concept symbols and role symbols. An $\mathcal{EL}$ concept C is defined as

$$C ::= A|\top|C \sqcap C|\exists r.C$$

where A and r range over N_C and N_R, respectively. In the following, we use symbols A, B to denote atomic concepts and C, D to denote arbitrary concepts. A *terminology* or *TBox* consists of *concept inclusion* axioms $C \sqsubseteq D$ and *concept equivalence* axioms $C \equiv D$ used as a shorthand for $C \sqsubseteq D$ and $D \sqsubseteq C$. While knowledge bases in general can also include a specification of individuals with the corresponding concept and role assertions (ABox), in this paper we abstract from ABoxes and concentrate on TBoxes. The signature of an $\mathcal{EL}$ concept C or an axiom α, denoted by sig(C) or sig(α), respectively, is the set of concepts and role symbols occurring in it. The signature of a TBox $\mathcal{T}$, in symbols sig($\mathcal{T}$), is analogously $N_C \cup N_R$. In what follows, we denote the set $N_C \cup \{\top\}$ as N_C^+.

Before introducing the fixpoint operators, we recall the semantics of the above introduced DL constructs, which is defined by the means of interpretations. An interpretation $\mathcal{I}$ is given by the domain $\Delta^\mathcal{I}$ and a function $\cdot^\mathcal{I}$ assigning each concept $A \in N_C$ a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$ and each role $r \in N_R$ a subset $r^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation of $\top$ is fixed to $\Delta^\mathcal{I}$. The interpretation of an arbitrary $\mathcal{EL}$ concept is defined inductively, i.e., $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$ and $(\exists r.C)^\mathcal{I} = \{x \mid (x, y) \in r^\mathcal{I}, y \in C^\mathcal{I}\}$. An interpretation $\mathcal{I}$ satisfies an axiom $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. $\mathcal{I}$ is a model of a TBox, if it satisfies all of its axioms. We say that a TBox $\mathcal{T}$ entails an axiom α, if α is satisfied by all models of $\mathcal{T}$. In combination with fixpoint constructors, we will additionally use *concept disjunction* $C \sqcup D$, the semantics of which is defined by $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$.

We now introduce the logics $\mathcal{EL}_{\mu(\sqcup),\nu}$, a fragment of monadic second order logics that we use to compute uniform interpolants of general $\mathcal{EL}$ TBoxes. $\mathcal{EL}_{\mu(\sqcup),\nu}$ is an extension of $\mathcal{EL}$ by the two fixpoint constructors, $\nu X.C_\nu$ [9] and $\mu X.C_\mu$ [4]. X is an element of the countably infinite set of concept variables N_V and C_ν, C_μ are constructed as follows:

$$C_\nu ::= X|A|\top|\nu X.C_\nu|C_\nu \sqcap C_\nu|\exists r.C_\nu$$

$$C_\mu ::= X|A|\top|\mu X.C_\mu|C_\mu \sqcup C_\mu|C_\mu \sqcap C_\mu|\exists r.C_\mu$$

where A ranges over atomic concepts and X ranges over N_V. All $\mathcal{EL}_\nu$ concepts and all $\mathcal{EL}_{\mu(\sqcup)}$ concepts are closed C_ν and C_μ expression, i.e., all concept variables are bound by the corresponding fixpoint constructor. Note that we define $\mathcal{EL}_\nu$ concepts and all $\mathcal{EL}_{\mu(\sqcup)}$ concepts in such a way, that no concept can contain both fixpoint constructors, i.e., we do not combine the two constructors within concepts. The semantics of the fixpoint constructors is defined using a mapping ϑ of concept variables to subsets of $\Delta^{\mathcal{I}}$. For an $\mathcal{EL}_{\mu(\sqcup),\nu}$ concept C and $W \subseteq \Delta^{\mathcal{I}}$, we denote a replacement of X by W as $C^{\mathcal{I},\vartheta[X \to W]}$. The semantics of $\mathcal{EL}_{\mu(\sqcup),\nu}$ concepts is defined by

$$(\nu X.C)^{\mathcal{I},\vartheta} = \bigcup\{W \subseteq \Delta^{\mathcal{I}} | W \subseteq C^{\mathcal{I},\vartheta[X \to W]}\}$$

$$(\mu X.C)^{\mathcal{I},\vartheta} = \bigcap\{W \subseteq \Delta^{\mathcal{I}} | C^{\mathcal{I},\vartheta[X \to W]} \subseteq W\}.$$

In order to allow for more succinct concept expressions, we use an extended version of the fixpoint constructs allowing for mutual recursion [12], [9]. The extended constructors have the form $\nu_i X_1...X_n.C_{\nu,1},...,C_{\nu,n}$ and $\mu_i X_1...X_n.C_{\mu,1},...,C_{\mu,n}$ with $1 \leq i \leq n$. The semantics is defined as follows: $(\nu_i X_1...X_n.C_1,...,C_n)^{\mathcal{I},\vartheta} = \bigcup\{W_i\}$ and $(\mu_i X_1...X_n.C_1,...,C_n)^{\mathcal{I},\vartheta} = \bigcap\{W_i\}$ such that there are $W_1,...,W_{i-1},W_{i+1},...,W_n$ with respectively $W_j \subseteq C_j^{\mathcal{I},\vartheta[X_1 \to W_1,...,X_n \to W_n]}$ and $C_j^{\mathcal{I},\vartheta[X_1 \to W_1,...,X_n \to W_n]} \subseteq W_j$ for $1 \leq j \leq n$.

3 TBox Inseparability and Uniform Interpolation

Intuitively, two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are inseparable w.r.t. a signature Σ if they have the same Σ consequences, i.e., consequences whose signature is a subset of Σ. Depending on the particular application requirements, the expressivity of those Σ consequences can vary from subsumption queries and instance queries to conjunctive queries. In this paper, we investigate forgetting based on concept inseparability of general $\mathcal{EL}$ terminologies defined analogously to previous work on inseparability, e.g., [8] or [7], as follows:

Definition 1. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two general $\mathcal{EL}$ TBoxes and Σ a signature. $\mathcal{T}_1$ and $\mathcal{T}_2$ are concept-inseparable w.r.t. Σ, in symbols $\mathcal{T}_1 \equiv^c_\Sigma \mathcal{T}_2$, if for all $\mathcal{EL}$ concepts C, D with $sig(C) \cup sig(D) \subseteq \Sigma$ holds $\mathcal{T}_1 \models C \sqsubseteq D$, iff $\mathcal{T}_2 \models C \sqsubseteq D$.*

Given a signature Σ and a TBox $\mathcal{T}$, the aim of uniform interpolation or forgetting is to determine a TBox $\mathcal{T}'$ with sig$(\mathcal{T}') \subseteq \Sigma$ such that $\mathcal{T} \equiv^c_\Sigma \mathcal{T}'$. $\mathcal{T}'$ is also called a *Uniform Interpolant (UI)* of $\mathcal{T}$ w.r.t. Σ. We call $\overline{\Sigma} = \text{sig}(\mathcal{T}) \setminus \Sigma$ the *forgotten* signature. In practise, the uniform interpolants are required to be finite. Therefore, in this paper, we investigate the existence of such uniform interpolants in $\mathcal{EL}$, i.e., uniform interpolants expressible by a finite set of finite axioms using only the language constructs of $\mathcal{EL}$. As demonstrated by the following example, in the presence of cyclic concept inclusions, a finite UI in $\mathcal{EL}$ might not exist for a particular $\mathcal{T}$ and a particular Σ.

Example 1. Forgetting the concept A in the TBox $\mathcal{T} = \{A' \sqsubseteq A, A \sqsubseteq A'', A \sqsubseteq \exists r.A, \exists s.A \sqsubseteq A\}$ results in an infinite chain of consequences $A' \sqsubseteq \exists r.\exists r.\exists r....A''$ and $\exists s.\exists s.\exists s....A' \sqsubseteq A''$ containing nested existential quantifiers of unbounded maximal depth.

Such infinite chain of consequences can be easily expressed in a finite way using the greatest fixpoint constructor ν and the least fixpoint constructor μ, thereby resulting in the inclusion axioms $A' \sqsubseteq \nu X.(A'' \sqcap \exists r.X)$ and $\mu X.(A' \sqcup \exists s.X) \sqsubseteq A''$. In the following, we show that for any $\mathcal{EL}$ TBox $\mathcal{T}$ and any signature Σ, a corresponding UI of $\mathcal{T}$ w.r.t. Σ in $\mathcal{EL}_{\mu(\sqcup),\nu}$ can always be computed. For this purpose, we reduce the problem of computing UI to the problem of computing *most general subconcepts* $\mathtt{MGS}(\Sigma, \mathcal{T}, A)$ and *most specific superconcepts* $\mathtt{MSS}(\Sigma, \mathcal{T}, A)$ for each concept $A \in \mathrm{sig}(\mathcal{T})$.

Definition 2. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox and Σ a signature. Further, let $A \in N_C$. A set of $\mathcal{EL}$ concepts C_i for $1 \le i \le n$ is $\mathtt{MSS}(\mathcal{T}, \Sigma, A)$, if:*

- *$sig(C_i) \subseteq \Sigma$ for all i,*
- *$\mathcal{T} \models \bigsqcup C_i \sqsubseteq A$ and $\mathcal{T} \not\models A \sqsubseteq \bigsqcup C_i$,*
- *for all $\mathcal{EL}$ concepts D with $sig(D) \subseteq \Sigma$ holds:*
  $\mathcal{T} \models D \sqsubseteq A$, iff $\mathcal{T} \models D \sqsubseteq \bigsqcap C_i$.

A set of $\mathcal{EL}$ concepts C_i for $1 \le i \le n$ is $\mathtt{MGS}(\mathcal{T}, \Sigma, A)$ if the following conditions are fulfilled:

- *$sig(C_i) \subseteq \Sigma$ for all i,*
- *$\mathcal{T} \models A \sqsubseteq \bigsqcap C_i$ and $\mathcal{T} \not\models \bigsqcap C_i \sqsubseteq A$*
- *for all $\mathcal{EL}$ concepts D with $sig(D) \subseteq \Sigma$ holds:*
  $\mathcal{T} \models A \sqsubseteq D$, iff $\mathcal{T} \models \bigsqcap C_i \sqsubseteq D$.

We denote $\mathtt{MSS}(A)$ and $\mathtt{MGS}(A)$ expressed in logic L by $\mathtt{MSS}^L(A)$ and $\mathtt{MGS}^L(A)$. If $\mathtt{MGS}(\mathcal{T}, \Sigma, A)$ consists of several incomparable disjuncts C_i, it cannot be expressed by an $\mathcal{EL}$ concept. However, in the following, it will come into notice that this is not further problematic for the computation of UI, since the disjunction appears only on the left-hand side and can therefore be expressed by the means of several inclusion axioms. If the TBox $\mathcal{T}$ and the signature Σ do not change, we omit them and simply write $\mathtt{MSS}(A)$ and $\mathtt{MGS}(A)$. For the remainder of this paper, we fix $\mathcal{T}$ to be a general $\mathcal{EL}$ TBox and Σ a signature.

4 Normalization

In order to simplify the computation of MGS and MSS, we apply the following normalization thereby restricting the syntactic form of $\mathcal{T}$. Analogously to the normalization employed in other approaches ([1], [6], [7]), we decompose complex axioms into syntactically simple ones. The decomposition is realized recursively by replacing expressions $B_1 \sqcap ... \sqcap B_n$ and $\exists r.B$ with fresh concept symbols until each axiom in $\mathcal{T}$ is one of $\{A \sqsubseteq B, A \equiv B_1 \sqcap ... \sqcap B_n, A \equiv \exists r.B\}$, where $A, B, B_i \in N_C \cup \{\top\}$ and $r \in N_R$. For this purpose, we introduce a minimal required set of fresh concept symbols $A' \in N_D$ and the corresponding definition axioms $(A' \equiv C)$ for each of them. In what follows, we assume that knowledge bases are normalized and refer to $N_C \cup N_D$ as N_C. Since concept symbols in N_D are fresh, they do not appear in Σ and are therefore elements of the forgotten signature $\overline{\Sigma}$. Further, we assume that equivalent concept symbols have

Algorithm 1 computing $\texttt{MGS}_{\texttt{core}}(A)$ for an $\mathcal{EL}$ TBox $\mathcal{T}$ and a signature Σ

```
1: $M \leftarrow \bigcup \texttt{MGS}_{\text{cond}}(D, A), D \in N_C$ such that $\mathcal{T} \models D \sqsubseteq A, A \neq D$
2: for all $A \equiv \bigsqcap_{1 \leq i \leq n} B_i \in \mathcal{T}$ do
3:     $M \leftarrow M \cup \{\bigsqcap_{C \in \texttt{REDUCE}_{\texttt{MSS}}(\{C_i | 1 \leq i \leq n\})} C | (C_1, ..., C_n) \in \texttt{MGS}_{\text{cond}}(B_1, A) \times ... \times \texttt{MGS}_{\text{cond}}(B_n, A)\}$
4: end for
5: for all $A \equiv \exists r.B \in \mathcal{T}$ with $r \in \Sigma$ do
6:     $M \leftarrow M \cup \{\exists\, r.C | C \in \texttt{MGS}_{\text{cond}}(B, A)\}$
7: end for
8: return $\texttt{REDUCE}_{\texttt{MGS}}(M)$
```

Algorithm 2 computing $\texttt{MSS}_{\texttt{core}}(A)$ for an $\mathcal{EL}$ TBox $\mathcal{T}$ and a signature Σ

```
1: $M \leftarrow \bigcup \texttt{MSS}_{\text{cond}}(D, A), D \in N_C^+$ such that $\mathcal{T} \models A \sqsubseteq D, A \neq D$
2: for all $A \equiv \bigsqcap_{1 \leq i \leq n} B_i \in \mathcal{T}$ do
3:     $M \leftarrow M \cup \{C | C \in \texttt{MSS}_{\text{cond}}(B_i, A)\}$
4: end for
5: for all $A \equiv \exists r.B \in \mathcal{T}$ with $r \in \Sigma$ do
6:     $M \leftarrow M \cup \{\exists\, r. \bigsqcap_{C \in \texttt{MSS}_{\text{cond}}(B,A)} C\}$
7: end for
8: return $\texttt{REDUCE}_{\texttt{MSS}}(M)$
```

been replaced by a single representative of the corresponding equivalence class.[1] The following lemma summarizes the properties of normalized TBoxes.

Lemma 1. *Any $\mathcal{T}$ can be extended into a normalized TBox $\mathcal{T}'$ and each model of $\mathcal{T}$ can be extended into a model of $\mathcal{T}'$.*

Proof Sketch. All introduced concepts in N_D are defined in terms of concepts with $\text{sig}(C) \subseteq \text{sig}(\mathcal{T})$, therefore each model of $\mathcal{T}$ can be extended into a model of $\mathcal{T}'$. □

5 Computing MGS and MSS for Acyclic TBoxes

Given an acyclic, normalized $\mathcal{EL}$ TBox $\mathcal{T}$ and a signature Σ, Algorithms 1 and 2 compute for each $A \in N_C$ the elements of $\texttt{MGS}(A)$ and $\texttt{MSS}(A)$, respectively. The algorithms are derived from a Gentzen-style proof system and proceed along the definitions for A in $\mathcal{T}$ as well as the inferred inclusions between atomic concepts involving A. The computation is indirectly recursive and consists of the procedure $\texttt{MGS}_{\texttt{core}}$ ($\texttt{MSS}_{\texttt{core}}$) containing the core computation and procedure $\texttt{MGS}_{\text{cond}}$ ($\texttt{MSS}_{\text{cond}}$) realizing the termination of the computation: if the first parameter of $\texttt{MGS}_{\text{cond}}$ ($\texttt{MSS}_{\text{cond}}$) – a concept B – is in Σ, it returns B itself, which is the basecase of the computation; otherwise, it calls in turn $\texttt{MGS}_{\texttt{core}}(B)$ ($\texttt{MSS}_{\texttt{core}}(B)$). Thereby, $\texttt{MGS}_{\text{cond}}$ ($\texttt{MSS}_{\text{cond}}$) ensures that $\texttt{MGS}$ and $\texttt{MSS}$ only contain Σ-concepts. To avoid confusion, we denote $\texttt{MGS}(A)$ and $\texttt{MSS}(A)$ obtained using this simple definition of $\texttt{MGS}_{\text{cond}}$ ($\texttt{MSS}_{\text{cond}}$) by $\texttt{MGS}_{\text{acyc}}(A)$ and $\texttt{MSS}_{\text{acyc}}(A)$.

[1] The elimination of equivalent symbols does not affect the correctness or completeness of the uniform interpolation, since the removed symbols can easily be included into the resulting TBox.

Definition 3. *Let $\mathcal{T}$ an acyclic $\mathcal{EL}$ TBox and $A \in N_C$.* $\mathtt{MGS}_{acyc}(A) = \mathtt{MGS}_{\mathtt{core}}(A)$ *and* $\mathtt{MSS}_{acyc}(A) = \mathtt{MSS}_{\mathtt{core}}(A)$*.*

While this separation of concerns between $\mathtt{MGS}_{\mathtt{core}}(A)$ ($\mathtt{MSS}_{\mathtt{core}}(A)$) and $\mathtt{MGS}_{\mathtt{cond}}(B, A)$ ($\mathtt{MSS}_{\mathtt{cond}}(B, A)$) appears not necessary in the simple case of acyclic TBoxes, in the next section we extend the computation to the general case of GCIs by adding further conditions to $\mathtt{MGS}_{\mathtt{cond}}(B, A)$ ($\mathtt{MSS}_{\mathtt{cond}}(B, A)$) without changing the core computation presented in Algorithms 1 and 2. In particular, the role of second parameter of $\mathtt{MGS}_{\mathtt{cond}}$ will become clear.

The functions $\mathtt{REDUCE}_{\mathtt{MGS}}$ and $\mathtt{REDUCE}_{\mathtt{MSS}}$ eliminate redundancy within the computed results, which is not just an optimization, but will also play an important role when deciding the existence of a uniform interpolant in $\mathcal{EL}$. The two functions are defined as follows:

Definition 4. *Let $M = \{C_i | 0 \le i \le n\}$ be a set of $\mathcal{EL}$ concepts and $\mathcal{T}_e = \{\}$.*

$$\mathtt{REDUCE}_{\mathtt{MSS}}(M) = \{C_i \in M | \text{ there is no } C_j \in M \text{ such that } \mathcal{T}_e \models C_j \sqsubseteq C_i\}$$

$$\mathtt{REDUCE}_{\mathtt{MGS}}(M) = \{C_i \in M | \text{ the is no } C_j \in M \text{ such that } \mathcal{T}_e \models C_i \sqsubseteq C_j\}$$

Both, $\mathtt{REDUCE}$ and $\mathtt{REDUCE}_C$, can be easily realized using standard reasoning procedures in $\mathcal{EL}_{\mu(\sqcup),\nu}$, which is known to be decidable in *ExpTime* [4]. It is easy to see that, in case of an acyclic TBox $\mathcal{T}$, both algorithms terminate, while, in case of cyclic terminologies, the algorithms do not need to terminate. In the next section, we extend the computation to be applicable to general TBoxes and ensure the termination also for this case.

6 MGS and MSS for General TBoxes

As already mentioned, the computation based on the simple version of $\mathtt{MGS}_{\mathtt{cond}}(B, A)$ and $\mathtt{MSS}_{\mathtt{cond}}(B, A)$ does not always terminate in case of general TBoxes such as the TBox in Example 1. In order to exactly specify the case, in which Algorithms 1 and 2 do not terminate, we use the following graph structures representing the possible flow of computation of $\mathtt{MGS}_{\mathtt{core}}$ and $\mathtt{MSS}_{\mathtt{core}}$ for a particular TBox $\mathcal{T}$ (independent from a particular signature):

Definition 5. *The* MSS- *and* MGS-*graphs* $\mathcal{A}_{\mathtt{MSS}}(\mathcal{T})$ *and* $\mathcal{A}_{\mathtt{MGS}}(\mathcal{T})$ *are defined as*

– $\mathcal{A}_{\mathtt{MSS}}(\mathcal{T}) = (\Gamma_{\mathtt{MSS}}, Q, E_{\mathtt{MSS}})$ *with the set of edge labels* $\Gamma_{\mathtt{MSS}} = N_R \cup \{\sqsubseteq\}$ *, the set of states* $Q = N_C$ *and the set of edges* $E_{\mathtt{MSS}} = \{(A, r, B) | A \equiv \exists r.B \in \mathcal{T}\} \cup \{(A, \sqsubseteq, B) | \mathcal{T} \models A \sqsubseteq B, A \neq B\}$*, where* $A, B \in Q$ *and* $r \in \Gamma_{\mathtt{MSS}}$*.*

– $\mathcal{A}_{\mathtt{MGS}}(\mathcal{T}) = (\Gamma_{\mathtt{MGS}}, Q, E_{\mathtt{MGS}})$ *with the set of edge labels* $\Gamma_{\mathtt{MGS}} = N_R \cup \{\sqsupseteq, \sqcap\}$*, the set of states* $Q = N_C$ *and the set of edges* $E_{\mathtt{MGS}} = \{(A, r, B) | A \equiv \exists r.B \in \mathcal{T}\} \cup \{(A, \sqsupseteq, B) | \mathcal{T} \models A \sqsupseteq B, A \neq B\} \cup \{(A, \sqcap, B) | A \equiv B \sqcap C \in \mathcal{T}$ *for any conjunction* C *of elements from* $Q\}$*, where* $A, B \in Q$ *and* $r \in \Gamma_{\mathtt{MGS}}$*.*

The two graphs can be constructed in linear time after the classification of the normalized TBox is finished. For $X \in \{\mathtt{MGS}, \mathtt{MSS}\}$, we denote the set of the paths in $\mathcal{A}_X(\mathcal{T}, \Sigma)$ from A to B as $L_X(A, B)$ and the set of the intersection-free paths from node

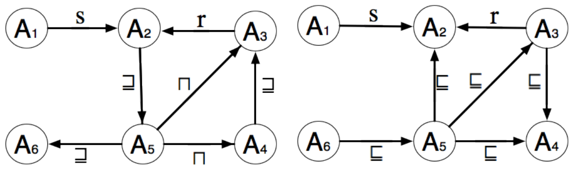


Fig. 1. MGS-graph (left) and MSS-graph (right) of $\mathcal{T}$.

A to itself as $L^1_X(A, A)$, i.e., such paths not contain any node except for A more than once. As illustrated by the example below, cyclic paths in $\mathcal{A}_{\mathtt{MSS}}(\mathcal{T})$ and $\mathcal{A}_{\mathtt{MGS}}(\mathcal{T})$ do not necessarily coincide.

Example 2. The corresponding MGS- and MSS-graphs of the TBox $\mathcal{T} = \{A_1 \equiv \exists s.A_2, A_3 \equiv \exists r.A_2, A_3 \sqsubseteq A_4, A_5 \equiv A_3 \sqcap A_4, A_5 \sqsubseteq A_2, A_5 \sqsubseteq A_6\}$ are shown in Fig. 1.

Given $\mathcal{A}_{\mathtt{MSS}}(\mathcal{T})$ and $\mathcal{A}_{\mathtt{MGS}}(\mathcal{T})$, we can easily determine for a particular signature Σ, whether the computation of the UI by the means of Algorithms 1 and 2 with the simple version of $\mathtt{MGS}_{\mathtt{cond}}(B, A)$ and $\mathtt{MSS}_{\mathtt{cond}}(B, A)$ terminates: if neither $\mathcal{A}_{\mathtt{MSS}}(\mathcal{T})$ nor $\mathcal{A}_{\mathtt{MGS}}(\mathcal{T})$ contain cyclic paths formed only by concepts from $\overline{\Sigma}$. Note that other cycles do not lead to a non-termination, since $\mathtt{MGS}_{\mathtt{cond}}(B, A) = \{B\}$ for any $B \in \Sigma$ and $A \in N_C$, i.e., the computation terminates. We denote such cyclic intersection-free paths from A containing only concepts from $\overline{\Sigma}$ by $L^{1,\overline{\Sigma}}_X(A, A)$ and the sets of concepts involved in such cycles by $\overline{\Sigma}_{C,\mathtt{MGS}} = \{A | A \in \overline{\Sigma}, L^{1,\overline{\Sigma}}_{\mathtt{MGS}}(A, A) \neq \emptyset\}$ and $\overline{\Sigma}_{C,\mathtt{MSS}} = \{A | A \in \overline{\Sigma}, L^{1,\overline{\Sigma}}_{\mathtt{MSS}}(A, A) \neq \emptyset\}$.

In order to be able to compute MSS and MGS also in case of $\overline{\Sigma}_{C,\mathtt{MSS}} \cup \overline{\Sigma}_{C,\mathtt{MGS}} \neq \emptyset$, we extend $\mathtt{MGS}_{\mathtt{cond}}(A, B)$ and $\mathtt{MSS}_{\mathtt{cond}}(A, B)$ by introducing a new condition for concepts $A \in \overline{\Sigma}_{C,\mathtt{MSS}} \cup \overline{\Sigma}_{C,\mathtt{MGS}}$. Here, we require the second parameter B to determine when the quantification of the fixpoint expressions is necessary. If $\mathtt{MGS}_{\mathtt{cond}}$ or $\mathtt{MSS}_{\mathtt{cond}}$ is called from outside of the corresponding cycles ($\overline{\Sigma}_{C,\mathtt{MGS}}$ for $\mathtt{MGS}_{\mathtt{cond}}$ and $\overline{\Sigma}_{C,\mathtt{MSS}}$ for $\mathtt{MSS}_{\mathtt{cond}}$), we return the complete fixpoint expression in its quantified form. Otherwise, we prefer to return the simplest possible value necessary, which can then be used as a part of a more complex, quantified concept expression. This second parameter is used by the caller – $\mathtt{MGS}_{\mathtt{core}}$ or $\mathtt{MSS}_{\mathtt{core}}$ – to pass its own parameter to the called $\mathtt{MGS}_{\mathtt{cond}}$ or $\mathtt{MSS}_{\mathtt{cond}}$ and let it then decide, whether to return a quantified fixpoint expression or a non-quantified one.

Definition 6. *Let n, m be the cardinality of $\overline{\Sigma}_{C,\mathtt{MSS}}$ and $\overline{\Sigma}_{C,\mathtt{MGS}}$, respectively. Further, let $A_i \in \overline{\Sigma}_{C,\mathtt{MSS}}$ with $0 \leq i \leq n$ and $A_j \in \overline{\Sigma}_{C,\mathtt{MGS}}$ with $0 \leq j \leq m$. Let $\{X(A_i) | A_i \in \overline{\Sigma}_{C,\mathtt{MSS}}\}$ and $\{Y(A_j) | A_j \in \overline{\Sigma}_{C,\mathtt{MGS}}\}$ be two disjoint sets of concept variables. Then, we define for each $A_i \in \overline{\Sigma}_{C,\mathtt{MSS}}$ and each $A_j \in \overline{\Sigma}_{C,\mathtt{MGS}}$:*

$$N(A_i) = \nu_i X(A_1), ..., X(A_n). \sqcap_{C \in \mathtt{MSS}_{\mathtt{core}}(A_1)} C, ..., \sqcap_{C \in \mathtt{MSS}_{\mathtt{core}}(A_n)} C$$

$$M(A_j) = \mu_j Y(A_1), ..., Y(A_m). \sqcup_{C \in \mathtt{MGS}_{\mathtt{core}}(A_1)} C, ..., \sqcup_{C \in \mathtt{MGS}_{\mathtt{core}}(A_m)} C.$$

$\mathtt{MSS}_{\mathtt{cond}}(A, B)$ *and* $\mathtt{MGS}_{\mathtt{cond}}(A, B)$ *for any $A, B \in N_C$ is then given by:*

$$\mathtt{MSS}_{\mathsf{cond}}(A,B) = \begin{cases} \{A\} & \textit{if } A \in \Sigma \\ \{X(A)\} & \textit{if } A \in \overline{\Sigma}_{C,\mathtt{MSS}}, \\ & B \in \overline{\Sigma}_{C,\mathtt{MSS}} \\ \{N(A)\} & \textit{if } A \in \overline{\Sigma}_{C,\mathtt{MSS}}, \\ & B \notin \overline{\Sigma}_{C,\mathtt{MSS}} \\ \mathtt{MSS}_{\mathsf{core}}(A) & \textit{otherwise} \end{cases} \qquad \mathtt{MGS}_{\mathsf{cond}}(A,B) = \begin{cases} \{A\} & \textit{if } A \in \Sigma \\ \{Y(A)\} & \textit{if } A \in \overline{\Sigma}_{C,\mathtt{MGS}}, \\ & B \in \overline{\Sigma}_{C,\mathtt{MGS}} \\ \{M(A)\} & \textit{if } A \in \overline{\Sigma}_{C,\mathtt{MGS}}, \\ & B \notin \overline{\Sigma}_{C,\mathtt{MGS}} \\ \mathtt{MGS}_{\mathsf{core}}(A) & \textit{otherwise} \end{cases},$$

and $\mathtt{MGS}(A) = \mathtt{MGS}_{\mathsf{cond}}(A, \top)$ *and* $\mathtt{MSS}(A) = \mathtt{MSS}_{\mathsf{cond}}(A, \top)$.

Note that, in case of an acyclic TBox, $\mathtt{MGS}(A)$ coincides with $\mathtt{MGS}_{\mathrm{acyc}}(A)$ described in Section 5, and the same holds for $\mathtt{MSS}(A)$.

Theorem 1 (Termination). *Let $A \in N_C$. The computation of* $\mathtt{MSS}(A)$ *and* $\mathtt{MGS}(A)$ *always terminates in at most exponential time.*

Proof Sketch. We first show by induction in case $\overline{\Sigma}_{C,\mathtt{MSS}} = \emptyset$ that the computation of $\mathtt{MSS}(A)$ for each $A \in N_C$ terminates. Then, we consider the case $\overline{\Sigma}_{C,\mathtt{MSS}} \neq \emptyset$. $\mathtt{MSS}_{\mathsf{cond}}$ encapsulates all concepts in $\overline{\Sigma}_{C,\mathtt{MSS}}$ into a single computational unit with direct or indirect incoming dependencies from concepts referencing concepts in $\overline{\Sigma}_{C,\mathtt{MSS}}$ and direct or indirect outgoing dependencies to concepts referenced from any concept in $\overline{\Sigma}_{C,\mathtt{MSS}}$. These two sets of referencing and referenced concepts are disjoint by definition of cyclicity. In the computation of $N(A)$, either concept variables or results of acyclic computations of $\mathtt{MSS}(B)$ for B not referencing $\overline{\Sigma}_{C,\mathtt{MSS}}$ are used, therefore each computation terminates. Once $N(A)$ is computed, references to $A \in \overline{\Sigma}_{C,\mathtt{MSS}}$ do not require further computation and the remaining computation terminates as shown in case $\overline{\Sigma}_{C,\mathtt{MSS}} \neq \emptyset$. Since the structure of $\mathtt{MGS}_{\mathsf{cond}}$ and $\mathtt{MSS}_{\mathsf{cond}}$ is analogous and $\mathtt{MGS}_{\mathsf{core}}$ also only iterates through the finite input directly, the argumentation for the termination of $\mathtt{MGS}$ is identical. The complexity is due to the conjunction constructs introduced in line 3 of Algorithm 1. □

Theorem 2 (Correctness $\mathtt{MSS}$ and $\mathtt{MGS}$). *Let $A \in N_C$. The computed* $\mathtt{MSS}(A)$ *and* $\mathtt{MGS}(A)$ *satisfy the conditions stated in Definition 2.*

The proof of Theorem 2 is based on a Gentzen-style proof system for normalized TBoxes.

7 Computing Uniform Interpolants

Given $\mathtt{MGS}(A)$ and $\mathtt{MSS}(A)$ for each $A \in N_C$, we can compute the UI in $\mathcal{EL}_{\mu(\sqcup),\nu}$ as follows:

Definition 7. $\mathtt{UI}(\mathcal{T},\Sigma) = \mathtt{UI}_{\Sigma,\mathtt{MSS}}(\mathcal{T},\Sigma) \cup \mathtt{UI}_{\Sigma,\mathtt{MGS}}(\mathcal{T},\Sigma) \cup \mathtt{UI}_{\overline{\Sigma}}(\mathcal{T},\Sigma)$ *with*

- $\mathtt{UI}_{\Sigma,\mathtt{MSS}}(\mathcal{T},\Sigma) = \{A \sqsubseteq D | A \in N_C \cap \Sigma, D \in \mathtt{MSS}(A)\}$,
- $\mathtt{UI}_{\Sigma,\mathtt{MGS}}(\mathcal{T},\Sigma) = \{C \sqsubseteq A | A \in N_C \cap \Sigma, C \in \mathtt{MGS}(A)\}$,
- $\mathtt{UI}_{\overline{\Sigma}}(\mathcal{T},\Sigma) = \{C \sqsubseteq D|$ *there is* $A \in N_C \cap \overline{\Sigma}$*, such that* $C \in \mathtt{MGS}(A)$ *and* $D \in \mathtt{MSS}(A)\}$.

Now, we have to prove that $\mathtt{UI}(\mathcal{T},\Sigma) \equiv^c_\Sigma \mathcal{T}$, i.e., the TBox $\mathtt{UI}(\mathcal{T},\Sigma)$ is in fact a uniform interpolant of $\mathcal{T}$ w.r.t. Σ.

Theorem 3 (UI). $\mathtt{UI}(\mathcal{T},\Sigma)$ *always exists and it holds that* $\mathtt{UI}(\mathcal{T},\Sigma) \equiv^c_\Sigma \mathcal{T}$.

The proof of Theorem 3 is also based on a Gentzen-style proof system for normalized TBoxes.

Deciding the Existence of UI in $\mathcal{EL}$

Clearly, if $\mathcal{T}$ does not contain pure $\overline{\Sigma}$ cycles, the $\mathtt{UI}(\mathcal{T},\Sigma)$ only contains $\mathcal{EL}$ constructs and, therefore, a $\mathtt{UI}$ in $\mathcal{EL}$ exists. This would be a sufficient, but not necessary criterion for the existence of a UI. From Definition 7, we can deduce a very general form of criterion requiring the deductive closure of any UI[2] to contain an (arbitrary) finite $\mathcal{EL}$ justification for the set of all non-$\mathcal{EL}$ axioms in the $\mathtt{UI}(\mathcal{T},\Sigma)$. Interestingly, given the $\mathcal{EL}$ TBox $\mathtt{UI}^{\mathcal{EL}}(\mathcal{T},\Sigma)$ obtained by extracting the $\mathcal{EL}$ part of each fixpoint concept within the non-mutual representation of $\mathtt{UI}(\mathcal{T},\Sigma)$, this criterion can be easily checked, since it is equivalent to a very simple criterion, which is an immediate consequence of the following theorem:

Theorem 4 (Existence). *Let* $\mathtt{UI}^{\mathcal{EL}}(\mathcal{T},\Sigma)$ *be the* $\mathcal{EL}$ *TBox obtained by extracting the* $\mathcal{EL}$ *part of each fixpoint concept within the non-mutual representation of* $\mathtt{UI}(\mathcal{T},\Sigma)$ *and let* $\mathcal{T}'$ *be an* $\mathcal{EL}$ *TBox with* $sig(\mathcal{T}') \subseteq \Sigma$ *such that* $\mathcal{T}' \equiv^c_\Sigma \mathcal{T}$*. Then* $\mathtt{UI}^{\mathcal{EL}}(\mathcal{T},\Sigma) \equiv \mathcal{T}'$.

The theorem claims that, if a finite $\mathcal{EL}$ justification for the set of all non-$\mathcal{EL}$ axioms in $\mathtt{UI}(\mathcal{T},\Sigma)$ exists, it is already a contained in the non-mutual representation of $\mathtt{UI}(\mathcal{T},\Sigma)$. Subsequently, a UI of $\mathcal{T}$ w.r.t. Σ in $\mathcal{EL}$ exists, iff $\mathtt{UI}^{\mathcal{EL}}(\mathcal{T},\Sigma) \models \mathtt{UI}(\mathcal{T},\Sigma)$. The proof of this theorem is based on the ideas stated in Lemmas 2 and 3, which show that there is a close relation between the existence of a UI in $\mathcal{EL}$ and redundancy in $\mathtt{UI}(\mathcal{T},\Sigma)$.

Lemma 2. *Let* $\mathcal{T}'$ *be an* $\mathcal{EL}$ *TBox with* $sig(\mathcal{T}') \subseteq \Sigma$ *such that* $\mathcal{T}' \equiv^c_\Sigma \mathcal{T}$*. Further, let* $A \in \overline{\Sigma}_{C,\mathtt{MGS}}$ *with* $C_1 \in \mathtt{MGS}(A)$ *and* $C_2 \in \mathtt{MSS}(A)$*. Then there is an* $\mathcal{EL}$ *concept* C' *such that*

- $\mathcal{T} \not\models C' \equiv C_1$ *and* $\mathcal{T} \not\models C' \equiv C_2$
- $\{\} \models C_1 \sqsubseteq C'$
- $\mathtt{UI}(\mathcal{T},\Sigma) \models C' \sqsubseteq C_2$.

Let $A \in \overline{\Sigma}_{C,\mathtt{MSS}}$ *with* $C_1 \in \mathtt{MGS}(A)$ *and* $C_2 \in \mathtt{MSS}(A)$*. Then there is an* $\mathcal{EL}$ *concept* C' *such that*

- $\mathcal{T} \not\models C' \equiv C_1$ *and* $\mathcal{T} \not\models C' \equiv C_2$
- $\{\} \models C' \sqsubseteq C_2$
- $\mathtt{UI}(\mathcal{T},\Sigma) \models C_1 \sqsubseteq C'$.

[2] The deductive closure is the same for any $\mathtt{UI}$ by definition.

Proof Sketch. Consider $C_1 = \mu X.(A \sqcup \exists r.X)$, which is the simplest possible non-$\mathcal{EL}$ concept in $\texttt{MGS}$. C_1 is semantically equivalent to an infinite disjunction of more and more specific $\mathcal{EL}$ concepts. The language constructs of $\mathcal{EL}$ do not allow us to specify a concept, which captures exactly the subset of the interpretation domain $C_1^{\mathcal{I}}$ in all models. Let C_2 be an arbitrary concept with $\texttt{UI}(\mathcal{T}, \Sigma) \models C_1 \sqsubseteq C_2$. If $C_1 \sqsubseteq C_2$ is a consequence of $\mathcal{T}'$, then there must be an $\mathcal{EL}$ concept C_1', which subsumes $C_1^{\mathcal{I}}$ in all models. Since $\mathcal{T}'$ is a finite $\mathcal{EL}$ TBox, it must hold that $\mathcal{T}' \models C_1 \sqsubseteq C_1'$, i.e., the latter inclusion axiom must be derived from the finite $\mathcal{EL}$ TBox itself (e.g., $C_1' = B$ with $\{\exists r.B \sqsubseteq B, A \sqsubseteq B\} \in \mathcal{T}'$). Moreover $C_1' \sqsubseteq C_2$ must have a justification in $\mathcal{T}'$ consisting of finitely many $\mathcal{EL}$ axioms. The same argumentation applies to C_2 as a concept with greatest fixpoint constructs. □

The above proof is the first step towards a connection between the redundancy in $\texttt{UI}(\mathcal{T}, \Sigma)$ and the existence of a UI in $\mathcal{EL}$. Since $\{C_1 \sqsubseteq C', C' \sqsubseteq C_2\} \models C_1 \sqsubseteq C_2$ and any minimal justification of $\{C_1 \sqsubseteq C', C' \sqsubseteq C_2\}$ in any $\mathcal{T}'$ does not contain $C_1 \sqsubseteq C_2$, it also holds that $\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma) \setminus \{C_1 \sqsubseteq C_2\} \models C_1 \sqsubseteq C_2$. Therefore, if $\mathcal{T}'$ exists, each non-$\mathcal{EL}$ axiom is redundant, i.e., it could be removed from $\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)$ without losing any consequences. To avoid confusion, we denote the non-mutual representation of $\texttt{MSS}(A)$ and $\texttt{MGS}(A)$ with the corresponding $\mathcal{EL}$ parts explicitly appearing outside of all fixpoint quantifiers by $\texttt{MSS}(A) \cup \texttt{EL}(\texttt{MSS}(A))$ and $\texttt{MGS}(A) \cup \texttt{EL}(\texttt{MGS}(A))$. The functions $\texttt{REDUCE}_{\texttt{MSS}}$ and $\texttt{REDUCE}_{\texttt{MGS}}$ have to be applied also when computing $\texttt{MSS}(A) \cup \texttt{EL}(\texttt{MSS}(A))$ and $\texttt{MGS}(A) \cup \texttt{EL}(\texttt{MGS}(A))$. Therefore, the redundancy can only appear during the construction of $\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)$. From the definition of $\texttt{MGS}$ and $\texttt{MSS}$ follows that the sets $\texttt{UI}_{\Sigma,\texttt{MSS}}(\mathcal{T}, \Sigma)$ and $\texttt{UI}_{\Sigma,\texttt{MGS}}(\mathcal{T}, \Sigma)$ in Definition 7 cannot be redundant if the sets $\texttt{MSS}(A) \cup \texttt{EL}(\texttt{MSS}(A))$ and $\texttt{MGS}(A) \cup \texttt{EL}(\texttt{MGS}(A))$ contain only incomparable elements. Therefore, it remains to consider the redundancy introduced during the construction of $\texttt{UI}_{\overline{\Sigma}}(\mathcal{T}, \Sigma)$. We denote by $P_{\overline{\Sigma}} = \{(C_1, C_2) |$ there is $A \in \overline{\Sigma}$ s.t. $C_1 \in \texttt{MGS}(A) \cup \texttt{EL}(\texttt{MGS}(A)), C_2 \in \texttt{MSS}(A) \cup \texttt{EL}(\texttt{MSS}(A))\}$ the set of all concept pairs relevant for the construction of $\texttt{UI}_{\overline{\Sigma}}(\mathcal{T}, \Sigma)$ and the subset of $P_{\overline{\Sigma}}$ containing the "redundant" concept pairs by $\mathcal{R} = \{(C_1, C_2) \in P_{\overline{\Sigma}} | (\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)) \setminus \{C_1 \sqsubseteq C_2\} \models C_1 \sqsubseteq C_2\}$. I.e., $\mathcal{R}$ is the set of concept pairs that are potentially nonessential for the construction of a $\texttt{UI}$ due to entailment of the corresponding inclusion axiom by the remainder of a $\texttt{UI}$ if the axiom itself is omitted. Due to possible dependencies between the elements of $\mathcal{R}$, there may be several different maximal subsets M of $\mathcal{R}$ such that $(\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)) \setminus \{C_1 \sqsubseteq C_2 | (C_1, C_2) \in M\} \models \texttt{UI}(\mathcal{T}, \Sigma)$. We denote the set of all such maximal subsets of $\mathcal{R}$ as $\mathcal{R}_{\texttt{MAX}} = \{M | M \subseteq \mathcal{R}, (\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)) \setminus \{C_1 \sqsubseteq C_2 | (C_1, C_2) \in M\} \models \texttt{UI}(\mathcal{T}, \Sigma)$, for all $(C_1', C_2') \in P_{\overline{\Sigma}} \setminus M$ holds $(\texttt{UI}(\mathcal{T}, \Sigma) \cup \texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma)) \setminus (\{C_1' \sqsubseteq C_2'\} \cup \{C_1 \sqsubseteq C_2 | (C_1, C_2) \in M\}) \not\models \texttt{UI}(\mathcal{T}, \Sigma)\}$. The next lemma states that if a concept pair with at least one non-$\mathcal{EL}$ concept is contained in one set $M \in \mathcal{R}_{\texttt{MAX}}$, it is contained in all $M \in \mathcal{R}_{\texttt{MAX}}$.

Lemma 3. *Let $\mathcal{T}'$ be an $\mathcal{EL}$ TBox with $sig(\mathcal{T}') \subseteq \Sigma$ such that $\mathcal{T}' \equiv_{\Sigma}^{c} \mathcal{T}$. Further, let $A \in \overline{\Sigma}_{C,\texttt{MSS}} \cup \overline{\Sigma}_{C,\texttt{MGS}}$ with $C_1 \in \texttt{MGS}(A)$ and $C_2 \in \texttt{MSS}(A)$. Let let $M' \in \mathcal{R}_{\texttt{MAX}}$ such that $(C_1, C_2) \in M'$. Then for each $M \in \mathcal{R}_{\texttt{MAX}}$ holds $(C_1, C_2) \in M$.*

Note that all concept pairs with at least one non-$\mathcal{EL}$ concept are contained in the intersection of $\mathcal{R}_{\texttt{MAX}}$, iff $\texttt{UI}^{\mathcal{EL}}(\mathcal{T}, \Sigma) \equiv \mathcal{T}'$. As a consequence of the above two lemmas and the fact that for any $(C_1, C_2) \in \mathcal{R}$ there exists at least one $M \in \mathcal{R}_{\mathtt{MAX}}$, it is sufficient to check whether all concept pairs with at least one non-$\mathcal{EL}$ concept are contained in $\mathcal{R}$ to determine whether the $\mathcal{T}'$ in Theorem 4 exists.

8 Summary

In this paper, we provide an *ExpTime* algorithm for computing a uniform interpolant of general $\mathcal{EL}$ terminologies preserving all $\mathcal{EL}$ concept inclusions for a particular signature based on the notion of *most general subconcepts* and *most specific superconcepts*. The result of the computation is expressed in logic $\mathcal{EL}_{\mu(\sqcup),\nu}$—an extension of $\mathcal{EL}$ with least fixpoint and greatest fixpoint constructors μ,ν as well as the disjunction used only on the left-hand side of concept inclusions. We also state the exact existence criteria for an $\mathcal{EL}$ interpolant and show how it can be obtained from the corresponding interpolant expressed in $\mathcal{EL}_{\mu(\sqcup),\nu}$.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05. Morgan-Kaufmann Publishers, Edinburgh, UK (2005)
2. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03). pp. 319–324 (2003)
3. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. J. Applied Logic 5(3), 392–420 (2007)
4. Calvanese, D., De Giacomo, G., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99). pp. 84–89 (1999)
5. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: Proc. of the 19th European Conf. on Artificial Intelligence (ECAI'10). pp. 479–484 (2010)
6. Kazakov, Y.: Consequence-driven reasoning for $\mathcal{H}$orn $\mathcal{SHIQ}$ ontologies. In: IJCAI. pp. 2040–2045 (July 11-17 2009)
7. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. of the 21st Int.Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 830–835 (2009)
8. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to dl-lite. Artif. Intell. 174, 1093–1141 (October 2010)
9. Lutz, C., Piro, R., Wolter, F.: Enriching $\mathcal{EL}$-concepts with greatest fixpoints. In: Proc. of the 19th European Conf. on Artificial Intelligence (ECAI'10). pp. 41–46 (2010)
10. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11) (2011)
11. Nikitina, N.: Uniform interpolation in general $\mathcal{EL}$-terminologies. Techreport, Institut AIFB, KIT, Karlsruhe (Mai 2011)
12. Schild, K.: Terminological cycles and the propositional μ-calculus. In: Proc. of the KR'94. pp. 509–520 (1994)

Reasoning-Supported Interactive Revision of Knowledge Bases

Nadeschda Nikitina[1], Sebastian Rudolph[1], and Birte Glimm[2]

[1] Karlsruhe Institute of Technology, Germany,
[1] {nikitina,rudolph}@kit.edu,
[2] The University of Oxford, Department of Computer Science, UK
[2] birte.glimm@cs.ox.ac.uk

Abstract. We propose a method for controlling the quality of (semi-)automatically acquired axioms. We combine the manual inspection of axioms with automatic evaluation decisions and propose *decision spaces* as a means to efficiently compute which decisions can be automatized and which axiom evaluation order is beneficial.

1 Introduction

Manual knowledge formalization for real-world knowledge-intensive applications is highly time-consuming. An application of (semi-)automatic knowledge acquisition methods such as ontology learning or matching is, therefore, often considered a reasonable way to reduce the cost of ontology development. Automatically acquired knowledge usually has to be manually inspected; either partially, to estimate the overall quality, or even fully, to maintain high quality standards.

So far, the knowledge representation community has been focusing on restoring the consistency of ontologies enriched with new axioms as done in various belief revision and repair approaches. Such approaches, however, are not directly suited in case a more restrictive quality control is required. We support an exhaustive manual inspection of newly acquired axioms before adding the selected ones into the ontology and call this process *interactive ontology revision*. Once a decision (add or not, i.e., accept or decline) has been made, we determine which other axioms can be evaluated automatically by exploiting logical dependencies between axioms.

We illustrate the main challenges with an example in which we have already confirmed that the axioms

$$\text{Metal} \sqsubseteq \text{Chemical\_Element} \tag{1}$$

$$\text{Chemical\_Element} \sqsubseteq \text{Material} \tag{2}$$

belong to the desired consequences, while the following axioms are still to be evaluated:

$$\text{Copper} \sqsubseteq \text{Material} \tag{3}$$

$$\text{Copper} \sqsubseteq \text{Chemical\_Element} \tag{4}$$

$$\text{Copper} \sqsubseteq \text{Metal} \tag{5}$$

If Axiom (3) is declined, we can immediately also decline Axioms (4) and (5) since accepting the axioms would implicitly lead to the undesired consequence (3). Similarly, if Axiom (5) is approved, Axioms (3) and (4) are implicit consequences, which can be approved automatically. If we start, however, with declining Axiom (5), no automatic evaluation can be performed. It can be observed that

– a high grade of automation requires a good evaluation order, and
– approval and decline decisions have a different impact.

Which axioms have the highest impact on decline or approval and which axioms can be automatically evaluated once a decision has been made can be determined with the help of algorithms for automated reasoning. Even for not very expressive knowledge representation formalisms, reasoning is an expensive task and in an interactive setting it is crucial to minimize the number of reasoning tasks while maximizing the number of automated decisions. We reduce the number of reasoning tasks by transferring ideas for ontology classification [8] to our problem. For this, we introduce the notion of *decision spaces*, which exploit the characteristics of the logical entailment relation between axioms to maximize the amount of information gained by reasoning. From the evaluation of our prototypical system, it can be observed that a considerable proportion of axioms can be evaluated automatically. Furthermore, decision spaces significantly reduce the number of required reasoning operations, resulting in a considerable performance gain.

In the next sction we formalize the basic notions and ideas; in Section 3, we define decision spaces, how they can be updated, and how they help to determine a beneficial axiom order. Our evaluation is presented in Section 4. Finally, we discuss related approaches in Section 5 before we conclude in Section 6. Further details and proofs can be found in a technical report [4]. The paper is an adaptation of our IJCAI'2011 paper [5].

2 Revision of Knowledge Bases

The approach proposed here is not only applicable to Description Logics, but to any logic where taking all consequences is a closure operation, i.e., extensive ($\{\varphi\} \models \varphi$), monotone ($\Phi \models \varphi$ implies $\Phi \cup \Psi \models \varphi$), and idempotent ($\Phi \models \varphi$ and $\Phi \cup \{\varphi\} \models \psi$ imply $\Phi \models \psi$). Moreover, we presume the existence of a decision procedure for logical entailment.

The revision of a knowledge base $\mathcal{K}$ aims at a separation of its axioms (i.e., logical statements) into two disjoint sets: the set of wanted consequences $\mathcal{K}^{\models}$ and the set of unwanted consequences $\mathcal{K}^{\not\models}$. This motivates the following definitions.

Definition 1 (Revision State). *A* revision state *is defined as a tuple* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *of knowledge bases with* $\mathcal{K}^{\models} \subseteq \mathcal{K}, \emptyset \neq \mathcal{K}^{\not\models} \subseteq \mathcal{K}$*, and* $\mathcal{K}^{\models} \cap \mathcal{K}^{\not\models} = \emptyset$*. Given two revision states* $(\mathcal{K}, \mathcal{K}_1^{\models}, \mathcal{K}_1^{\not\models})$ *and* $(\mathcal{K}, \mathcal{K}_2^{\models}, \mathcal{K}_2^{\not\models})$*, we call* $(\mathcal{K}, \mathcal{K}_2^{\models}, \mathcal{K}_2^{\not\models})$ *a refinement of* $(\mathcal{K}, \mathcal{K}_1^{\models}, \mathcal{K}_1^{\not\models})$*, if* $\mathcal{K}_1^{\models} \subseteq \mathcal{K}_2^{\models}$ *and* $\mathcal{K}_1^{\not\models} \subseteq \mathcal{K}_2^{\not\models}$*. A revision state is* complete*, if* $\mathcal{K} = \mathcal{K}^{\models} \cup \mathcal{K}^{\not\models}$*, and* incomplete *otherwise. An incomplete revision state* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *can be refined by evaluating a further axiom* $\alpha \in \mathcal{K} \setminus (\mathcal{K}^{\models} \cup \mathcal{K}^{\not\models})$*, obtaining* $(\mathcal{K}, \mathcal{K}^{\models} \cup \{\alpha\}, \mathcal{K}^{\not\models})$ *or* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models} \cup \{\alpha\})$*. We call the resulting revision state an* elementary refinement *of* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$*.*

Since we expect that the deductive closure of the wanted consequences in $\mathcal{K}^{\models}$ must not contain unwanted consequences, we introduce the notion of *consistency* for revision

Algorithm 1 Interactive Knowledge Base Revision

```
Input:  $(\mathcal{K}, \mathcal{K}_0^{\models}, \mathcal{K}_0^{\not\models})$ a consistent revision state
Output: $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ a complete and consistent revision state
1: $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) \leftarrow \mathrm{clos}(\mathcal{K}, \mathcal{K}_0^{\models}, \mathcal{K}_0^{\not\models})$
2: while $\mathcal{K}^{\models} \cup \mathcal{K}^{\not\models} \neq \mathcal{K}$ do
3:     choose $\alpha \in \mathcal{K} \setminus (\mathcal{K}^{\models} \cup \mathcal{K}^{\not\models})$
4:     if expert confirms $\alpha$ then
5:         $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) \leftarrow \mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models} \cup \{\alpha\}, \mathcal{K}^{\not\models})$
6:     else
7:         $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) \leftarrow \mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models} \cup \{\alpha\})$
8:     end if
9: end while
```

states. If we want to maintain consistency, a single evaluation decision can predetermine the decision for several yet unevaluated axioms. These implicit consequences of a refinement are captured in the *revision closure*.

Definition 2 (Revision State Consistency and Closure). *A (complete or incomplete) revision state* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *is* consistent *if there is no* $\alpha \in \mathcal{K}^{\not\models}$ *such that* $\mathcal{K}^{\models} \models \alpha$. *The* revision closure $\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *of* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *is* $(\mathcal{K}, \mathcal{K}_c^{\models}, \mathcal{K}_c^{\not\models})$ *with* $\mathcal{K}_c^{\models} := \{\alpha \in \mathcal{K} \mid \mathcal{K}^{\models} \models \alpha\}$ *and* $\mathcal{K}_c^{\not\models} := \{\alpha \in \mathcal{K} \mid \mathcal{K}^{\models} \cup \{\alpha\} \models \beta$ *for some* $\beta \in \mathcal{K}^{\not\models}\}$.

We can show the following useful properties of the closure of consistent revision states:

Lemma 1. *For* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *a consistent revision state,*

1. $\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *is consistent,*
2. *every elementary refinement of* $\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *is consistent,*
3. *every consistent complete refinement of* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *is a refinement of* $\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$.

Algorithm 1 employs the above properties to implement a general methodology for interactive knowledge base revision.

Instead of starting with empty sets for $\mathcal{K}_0^{\models}$ and $\mathcal{K}_0^{\not\models}$, we can initialize the latter sets with approved and declined axioms from a previous revision or add axioms of the knowledge base that is being developed to $\mathcal{K}_0^{\models}$. We can further initialize $\mathcal{K}_0^{\not\models}$ with axioms that express inconsistency and unsatisfiability of predicates (i.e. of classes or relations) in $\mathcal{K}$, which we assume to be unwanted consequences.

In line 3, an axiom is chosen that is evaluated next. As motivated in the introduction, a random decision can have a detrimental effect on the amount of manual decisions. Ideally, we want to rank the axioms and choose one that allows for a high number of consequential automatic decisions. For this purpose, we introduce the following notion of *axiom impact*.

Definition 3 (Impact). *Let* $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ *be a consistent revision state with* $\alpha \in \mathcal{K}$ *and let* $?(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) := |\mathcal{K} \setminus (\mathcal{K}^{\models} \cup \mathcal{K}^{\not\models})|$. *For an axiom* α,

- *the* approval impact *is:* $\mathit{impact}^{+}(\alpha) = ?(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) - ?(\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models} \cup \{\alpha\}, \mathcal{K}^{\not\models}))$,
- *the* decline impact *is:* $\mathit{impact}^{-}(\alpha) = ?(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models}) - ?(\mathrm{clos}(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models} \cup \{\alpha\}))$,
- *the* guaranteed impact *is:* $\mathit{guaranteed}(\alpha) = \min(\mathit{impact}^{+}(\alpha), \mathit{impact}^{-}(\alpha))$.

The approval (decline) impact of an axiom α is determined by the number of automatically evaluated axioms in case α is approved (declined), while the guaranteed impact is the minimum of the two impact functions.

In the example from Section 1, Axioms (3), (4) and (5) have an approval impact of 0, 1, and 2, a decline impact of 2, 1, and 0, and a guaranteed impact of 0, 1, and 0, respectively. We show in the evaluation that the ratio of accepted axioms to all axioms that are to be evaluated can be used to determine which impact function is best.

Since computing such an impact as well as computing the closure after each evaluation (lines 1, 5, and 7) can be considered very expensive, we next introduce *decision spaces*, auxiliary data structures which significantly reduce the cost of computing the closure upon elementary revisions and provide an elegant way of determining high impact axioms.

3 Decision Spaces

Intuitively, the purpose of decision spaces is to keep track of the dependencies between the axioms in such a way, that we can read-off the consequences of revision state refinements upon an approval or a decline of an axiom, thereby reducing the required reasoning operations. Furthermore, we will show how we can update these structures after a refinement step avoiding many costly recomputations.

Definition 4 (Decision Space). *Given a revision state $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ with $\mathcal{K}^{\not\models} \neq \emptyset$, the according* decision space $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})} = (\mathcal{K}^{?}, E, C)$ *contains the set*

$$\mathcal{K}^{?} := \mathcal{K} \setminus (\{\alpha \mid \mathcal{K}^{\models} \models \alpha\} \cup \{\alpha \mid \mathcal{K}^{\models} \cup \{\alpha\} \models \beta \text{ for some } \beta \in \mathcal{K}^{\not\models}\})$$

of unevaluated axioms together with two binary relations, E (entails) and C (conflicts) on $\mathcal{K}^{?}$:

$$\alpha E \beta \text{ iff } \mathcal{K}^{\models} \cup \{\alpha\} \models \beta \qquad\qquad \alpha C \beta \text{ iff } \mathcal{K}^{\models} \cup \{\alpha, \beta\} \models \gamma \text{ for some } \gamma \in \mathcal{K}^{\not\models}$$

The requirement that $\mathcal{K}^{\not\models} \neq \emptyset$ is without loss of generality since we can always add an axiom that expresses a contradiction (an inconsistency), which is clearly undesired. As a direct consequence of this definition, we have $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})} = \mathbb{D}_{\mathrm{clos}(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})}$. Also the following properties are immediate from the above definition:

Lemma 2. *Given $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})} = (\mathcal{K}^{?}, E, C)$ for a revision state $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$, $\mathcal{K}^{\not\models} \neq \emptyset$,*

P1 $(\mathcal{K}^{?}, E)$ is a quasi-order (i.e., reflexive and transitive),
P2 C is symmetric,
P3 $\alpha E \beta$ and $\beta C \gamma$ imply $\alpha C \gamma$ for all $\alpha, \beta, \gamma \in \mathcal{K}^{?}$, and
P4 if $\alpha E \beta$ then $\alpha C \beta$ does not hold.

On the other hand, the properties established in the above lemma are characteristic:[1]

Lemma 3. *Let V be finite set and let $\underline{E}, \underline{C} \subseteq V \times V$ be relations for which $(V, \underline{E})$ is a quasi-order, $\underline{C} = \underline{C}^{-}$, $\underline{E} \circ \underline{C} \subseteq \underline{C}$ and $\underline{E} \cap \underline{C} = \emptyset$. Then there is a decision space $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})}$ isomorphic to $(V, \underline{E}, \underline{C})$.*

[1] As usual, we let $R^{-} = \{(y, x) \mid (x, y) \in R\}$ as well as $R \circ S = \{(x, z) \mid (x, y) \in R, (y, z) \in S \text{ for some } y\}$.

The following lemma shows how decision spaces can be used for calculating closures of updated revision states and impacts of axioms. As usual for (quasi)orders, we define $\uparrow\alpha = \{\beta \mid \alpha E\beta\}$ and $\downarrow\alpha = \{\beta \mid \beta E\alpha\}$. Moreover, we let $\wr\alpha = \{\beta \mid \alpha C\beta\}$.

Lemma 4. *Given* $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})} = (\mathcal{K}^{?}, E, C)$ *for a revision state* $(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})$ *such that* $(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models}) = \mathrm{clos}(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})$ *with* $\mathcal{K}^{\not\models} \neq \emptyset$ *and* $\alpha \in \mathcal{K}^{?}$*, then*

1. $\mathrm{clos}(\mathcal{K},\mathcal{K}^{\models} \cup \{\alpha\},\mathcal{K}^{\not\models}) = (\mathcal{K},\mathcal{K}^{\models} \cup \uparrow\alpha,\mathcal{K}^{\not\models} \cup \wr\alpha)$ *and*
2. $\mathrm{clos}(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models} \cup \{\alpha\}) = (\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models} \cup \downarrow\alpha)$.
3. $\mathit{impact}^{+}(\alpha) = |\uparrow\alpha| + |\wr\alpha|$
4. $\mathit{impact}^{-}(\alpha) = |\downarrow\alpha|$

Hence, the computation of the revision closure (lines 5 and 7) and axiom impacts does not require any entailment checks if the according decision space is available. For the computation of decision spaces, we exploit the structural properties established in Lemmas 2 and 3 in order to reduce the number of required entailment checks in cases where the relations E and C are partially known. For this purpose, we define the rules R0 to R9, which describe the connections between the relations E and C and their complements $\overline{E}$ and $\overline{C}$. The rules can serve as production rules to derive new instances of these relations thereby minimizing calls to costly reasoning procedures.

R0	$\rightarrow E(x,x)$	reflexivity of E
R1	$E(x,y) \wedge E(y,z) \rightarrow E(x,z)$	transitivity of E
R2	$E(x,y) \wedge C(y,z) \rightarrow C(x,z)$	(P3)
R3	$C(x,y) \rightarrow C(y,x)$	symmetry of C
R4	$E(x,y) \rightarrow \overline{C}(x,y)$	disjointness of E and C
R5	$\overline{C}(x,y) \rightarrow \overline{C}(y,x)$	symmetry of C
R6	$E(x,y) \wedge \overline{C}(x,z) \rightarrow \overline{C}(y,z)$	(P3)
R7	$C(x,y) \rightarrow \overline{E}(x,y)$	disjointness of E and C
R8	$\overline{C}(x,y) \wedge C(y,z) \rightarrow \overline{E}(x,z)$	(P3)
R9	$E(x,y) \wedge \overline{E}(x,z) \rightarrow \overline{E}(y,z)$	transitivity of E

An analysis of the dependencies between the rules R0 to R9 reveals an acyclic structure (indicated by the order of the rules). Therefore E,C,$\overline{C}$, and $\overline{E}$ can be saturated one after another. Moreover, the exhaustive application of the rules R0 to R9 can be condensed into the following operations:

$$\begin{aligned}
&E \leftarrow E^{*}\\
&C \leftarrow E \circ (C \cup C^{-}) \circ E^{-}\\
&\overline{C} \leftarrow E^{-} \circ (\overline{C} \cup Id \cup \overline{C}^{-}) \circ E\\
&\overline{E} \leftarrow E^{-} \circ (\overline{C} \circ C \cup \overline{E}) \circ E^{-}
\end{aligned}$$

The correctness of the first operation (where $(\cdot)^{*}$ denotes the reflexive and transitive closure) is a direct consequence of R0 and R1. For the second operation, we exploit the relationships

$$E{\circ}C{\circ}E^{-} \overset{\text{R2}}{\subseteq} C{\circ}E^{-} \overset{\text{R3}}{\subseteq} C^{-}{\circ}E^{-} \overset{\text{R2}}{\subseteq} C^{-} \overset{\text{R3}}{\subseteq} C$$

$$E{\circ}C^{-}{\circ}E^{-} \overset{\text{R2}}{\subseteq} E{\circ}C^{-} \overset{\text{R3}}{\subseteq} E{\circ}C \overset{\text{R2}}{\subseteq} C$$

that can be further composed into

$$E{\circ}C{\circ}E^{-} \cup E{\circ}C^{-}{\circ}E^{-} = E \circ (C \cup C^{-}) \circ E^{-} \subseteq C$$

Conversely, iterated backward chaining for C w.r.t. R2 and R3 yields $E \circ (C \cup C^{-}) \circ E^{-}$ as a fixpoint, under the assumption $E = E^{*}$. The correctness of the last two operations can be shown accordingly.

Algorithm 2 Decision Space Completion

```
Input: $(\mathcal{K}, \mathcal{K}^{\models}, \mathcal{K}^{\not\models})$ a consistent revision state; $E, \overline{E}, C, \overline{C}$ subsets of the entailment and conflict relations and their complements
Output: $(\mathcal{K}^{?}, E, C)$ the corresponding decision space
 1: $E \leftarrow E^{*}$
 2: $C \leftarrow E \circ C \circ E^{-}$
 3: $C \leftarrow C \cup C^{-}$
 4: $\overline{C} \leftarrow E^{-} \circ \overline{C} \cup Id_{\mathcal{K}^{?}} \circ E$
 5: $\overline{C} \leftarrow \overline{C} \cup \overline{C}^{-}$
 6: $\overline{E} \leftarrow (\overline{C} \circ C) \cup \overline{E}$
 7: $\overline{E} \leftarrow E^{-} \circ \overline{E} \circ E^{-}$
 8: while $E \cup \overline{E} \neq \mathcal{K}^{?} \times \mathcal{K}^{?}$ do
 9:     pick one $(\alpha,\beta) \in \mathcal{K}^{?} \times \mathcal{K}^{?} \setminus (E \cup \overline{E})$
10:     if $\mathcal{K}^{\models} \cup \{\alpha\} \models \beta$ then
11:         $E' \leftarrow \texttt{transupdatediff}(E, (\alpha,\beta))$
12:         $E \leftarrow E \cup E'$
13:         $C' \leftarrow (E' \circ C) \setminus C$
14:         $C' \leftarrow C' \cup (C' \circ E'^{-}) \setminus C$
15:         $C \leftarrow C \cup C'$
16:         $\overline{C}' \leftarrow (E'^{-} \circ \overline{C}) \setminus \overline{C}$
17:         $\overline{C}' \leftarrow \overline{C}' \cup (\overline{C}' \circ E') \setminus \overline{C}$
18:         $\overline{C} \leftarrow \overline{C} \cup \overline{C}'$
19:         $\overline{E}' \leftarrow ((\overline{C}' \circ C) \cup (\overline{C} \circ C')) \setminus \overline{E}$
20:         $\overline{E} \leftarrow \overline{E} \cup \overline{E}'$
21:         $\overline{E}' \leftarrow ((E'^{-} \circ \overline{E}) \cup (E^{-} \circ \overline{E}')) \setminus \overline{E}$
22:         $\overline{E} \leftarrow \overline{E} \cup \overline{E}' \cup (\overline{E}' \circ E^{-}) \cup (\overline{E} \circ E'^{-})$
23:     else
24:         $\overline{E} \leftarrow \overline{E} \cup (E^{-} \circ \{(\alpha,\beta)\} \circ E^{-})$
25:     end if
26: end while
27: while $C \cup \overline{C} \neq \mathcal{K}^{?} \times \mathcal{K}^{?}$ do
28:     pick one $(\alpha,\beta) \in \mathcal{K}^{?} \times \mathcal{K}^{?} \setminus (C \cup \overline{C})$
29:     if $\mathcal{K}^{\models} \cup \{\alpha,\beta\} \models \gamma$ for some $\gamma \in \mathcal{K}^{\not\models}$ then
30:         $C' \leftarrow E \circ \{(\alpha,\beta), (\beta,\alpha)\} \circ E^{-}$
31:         $C \leftarrow C \cup C'$
32:         $\overline{E} \leftarrow \overline{E} \cup (E^{-} \circ \overline{C} \circ C' \circ E^{-})$
33:     else
34:         $\overline{C}' \leftarrow (E^{-} \circ \{(\alpha,\beta), (\beta,\alpha)\} \circ E) \setminus \overline{C}$
35:         $\overline{C} \leftarrow \overline{C} \cup \overline{C}'$
36:         $\overline{E} \leftarrow \overline{E} \cup (E^{-} \circ \overline{C}' \circ C \circ E^{-})$
37:     end if
38: end while
```

Algorithm 3 Decision Space Update on Declining α

```
Input: $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})}$ a decision space, $\alpha \in \mathcal{K}^{?}$ an axiom
Output: $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models} \cup \{\alpha\})}$ the updated decision space
 1: $\mathcal{K}^{?} \leftarrow \mathcal{K}^{?} \setminus \downarrow\alpha$,
 2: $E \leftarrow E \cap (\mathcal{K}^{?} \times \mathcal{K}^{?})$
 3: $\overline{E} \leftarrow \overline{E} \cap (\mathcal{K}^{?} \times \mathcal{K}^{?})$
 4: $C \leftarrow C \cap (\mathcal{K}^{?} \times \mathcal{K}^{?})$
 5: $\overline{C} \leftarrow E^{-} \circ E$
 6: while $C \cup \overline{C} \neq \mathcal{K}^{?} \times \mathcal{K}^{?}$ do
 7:     pick one $(\beta,\gamma) \in \mathcal{K}^{?} \times \mathcal{K}^{?} \setminus (C \cup \overline{C})$
 8:     if $\mathcal{K}^{\models} \cup \{\beta,\gamma\} \models \alpha$ then
 9:         $C \leftarrow C \cup (E \circ \{(\beta,\gamma), (\gamma,\beta)\} \circ E^{-})$
10:     else
11:         $\overline{C} \leftarrow \overline{C} \cup (E^{-} \circ \{(\beta,\gamma), (\gamma,\beta)\} \circ E)$
12:     end if
13: end while
```

Algorithm 4 Decision Space Update on Approving α

```
Input: $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models},\mathcal{K}^{\not\models})}$ a decision space, $\alpha \in \mathcal{K}^{?}$ an axiom
Output: $\mathbb{D}_{(\mathcal{K},\mathcal{K}^{\models} \cup \{\alpha\},\mathcal{K}^{\not\models})}$ the updated decision space
 1: $\mathcal{K}^{?} \leftarrow \mathcal{K}^{?} \setminus (\uparrow\alpha \cup \wr\alpha)$
 2: $E \leftarrow E \cap (\mathcal{K}^{?} \times \mathcal{K}^{?})$
 3: $C \leftarrow C \cap (\mathcal{K}^{?} \times \mathcal{K}^{?})$
 4: $\overline{C} \leftarrow E^{-} \circ E$
 5: $\overline{E} \leftarrow E^{-} \circ \overline{C} \circ C \circ E^{-}$
 6: execute lines 8–38 from Alg. 2
```

Algorithm 2 realizes the cost-saving identification of the complete entailment and conflict relations of a decision space. Maintaining sets of known entailments (E), non-entailments ($\overline{E}$), conflicts (C) and non-conflicts ($\overline{C}$), the algorithm always closes these sets under the above operations before it cautiously executes expensive deduction checks to clarify missing cases. First, the initially known (non-)entailments and (non-)conflicts are closed in the aforementioned way (lines 1–7). There and in the subsequent lines, we split computations into several ones where appropriate in order to minimize the size of sets subject to the join operation ($\circ$). Lines 8–26 describe the successive clarification of the entailment relation (for cases where neither entailment nor non-entailment is known yet) via deduction checks. After each such clarification step, the sets $E, \overline{E}, C$, and $\overline{C}$ are closed. Thereby, we exploit known properties of intermediate results such as already being transitive or symmetric to avoid redoing the according closure operations unnecessarily (`transupdatediff` computes, for a relation R and a pair of elements (α,β), the difference between the reflexive transitive closure of R extended with (α,β) and R^*, i.e., $(R \cup \{(\alpha,\beta)\})^* \setminus R^*$)). Likewise, we also avoid redundant computations and reduce the size of the input sets for the join operations by explicitly bookkeeping sets E',C',$\overline{C}'$, and $\overline{E}'$ containing only the instances newly added in the current step. Lines 27–38 proceed in the analog way for stepwise clarification of the conflicts relation.

3.1 Updating Decision Spaces

We proceed by formally describing the change of the decision space as a consequence of approving or declining one axiom with the objective of again minimizing the required number of entailment checks. We first consider the case that an expert approves an axiom $\alpha \in \mathcal{K}^?$, and hence α is added to the set $\mathcal{K}^\models$ of wanted consequences.

Lemma 5. *Let* $\mathbb{D}_{(\mathcal{K},\mathcal{K}^\models,\mathcal{K}^\not\models)} = (\mathcal{K}^?, E, C)$, $\alpha \in \mathcal{K}^?$, $\mathbb{D}_{(\mathcal{K},\mathcal{K}^\models \cup \{\alpha\},\mathcal{K}^\not\models)} = (\mathcal{K}^?_{\text{new}}, E', C')$. *Then*
- $\mathcal{K}^?_{\text{new}} = \mathcal{K}^? \setminus (\uparrow\alpha \cup \wr\alpha)$,
- $\beta E \gamma$ *implies* $\beta E' \gamma$ *for* $\beta,\gamma \in \mathcal{K}^?_{\text{new}}$, *and*
- $\beta C \gamma$ *implies* $\beta C' \gamma$ *for* $\beta,\gamma \in \mathcal{K}^?_{\text{new}}$.

Essentially, the lemma states that all axioms entailed by α (as witnessed by E) as well as all axioms conflicting with α (indicated by C) will be removed from the decision space if α is approved. Moreover due to monotonicity, all positive information about entailments and conflicts remains valid. Algorithm 4 takes advantage of these correspondences when fully determining the updated decision space.

The next lemma considers changes to be made to the decision space on the denial of an axiom α by characterizing it as unwanted consequence.

Lemma 6. *Let* $\mathbb{D}_{(\mathcal{K},\mathcal{K}^\models,\mathcal{K}^\not\models)} = (\mathcal{K}^?, E, C)$, $\alpha \in \mathcal{K}^?$, $\mathbb{D}_{(\mathcal{K},\mathcal{K}^\models,\mathcal{K}^\not\models \cup \{\alpha\})} = (\mathcal{K}^?_{\text{new}}, E', C')$. *Then*
- $\mathcal{K}^?_{\text{new}} = \mathcal{K}^? \setminus \downarrow\alpha$,
- $\beta E \gamma$ *exactly if* $\beta E' \gamma$ *for* $\beta,\gamma \in \mathcal{K}^?_{\text{new}}$, *and*
- $\beta C \gamma$ *implies* $\beta C' \gamma$ *for* $\beta,\gamma \in \mathcal{K}^?_{\text{new}}$.

The lemma shows that the updated decision space can be obtained by removing all axioms that entail α. Furthermore entailments between remaining axioms remain unaltered whereas the set of conflicts may increase. Algorithm 3 implements the respective

Table 1. Characteristics of the evaluated datasets

dataset	size	validity ratio	dataset	size	validity ratio
S_1	54	94%	S_4	35	48%
S_2	60	100%	S_5	26	26%
S_3	40	45%	S_6	72	12%

decision space update, additionally exploiting that new conflicts can only arise from derivability of the newly declined axiom α.

Algorithms 4 and 3 have to be called in Alg. 1 after the accept (line 5) or decline revision step (line 7), respectively.

For n the number of involved axioms, Algorithms 2, 4, and 3 run in time bounded by $O(n^5)$ and space bounded by $O(n^2)$ if we treat entailment checking as a constant time operation. Without the latter assumption, the complexity of reasoning usually dominates. For example, if the axioms use all features of OWL 2 DL, entailment checking is $N2ExpTime$-complete, which then also applies to our algorithm.

4 Evaluation

For a first evaluation of the developed methodology, we choose a scenario motivated by ontology-supported literature search. The hand-crafted *NanOn* ontology models the scientific domain of nano technology, including substances, structures, procedures used in that domain. The ontology, denoted here with $\mathcal{O}$, is specified in the Web Ontology Language OWL DL [6] and comprises 2,289 logical axioms. The project associated to NanOn aims at developing techniques to automatically analyze scientific documents for the occurrence of NanOn concepts. When such concepts are found, the document is automatically annotated with NanOn concepts to facilitate topic-specific information retrieval on a fine-grained level. Since total accuracy of the automatically added annotations (which can be seen as logical axioms expressing factual knowledge) cannot be guaranteed, they need to be inspected by human experts, which provides a natural application scenario for our approach.

For our evaluation, we employed tools for automated textual analysis to produce a set of document annotations, the validity of which was then manually evaluated. This provided us with sets of valid and invalid annotation facts (denoted by $\mathcal{A}^+$ and $\mathcal{A}^-$, respectively). To investigate how the a priori quality of each axiom set influences the results, we created six distinct annotation sets S_1 to S_6 using different annotation methods. The different methods result in different *validity ratios* $|\mathcal{A}^+|/(|\mathcal{A}^+| + |\mathcal{A}^-|)$ of the datasets, where $|S|$ denotes the cardinality of a set S. The size of each set followed by the corresponding validity ratio in percent are shown in Table 1.

We then applied our methodology starting from the revision state $(\mathcal{O} \cup \mathcal{O}^- \cup \mathcal{A}^+ \cup \mathcal{A}^-, \mathcal{O}, \mathcal{O}^-)$ with $\mathcal{O}$ containing the axioms of the NanOn ontology and with $\mathcal{O}^-$ containing axioms expressing inconsistency and concept unsatisfiability. We obtained a complete revision state $(\mathcal{O} \cup \mathcal{O}^- \cup \mathcal{A}^+ \cup \mathcal{A}^-, \mathcal{O} \cup \mathcal{A}^+, \mathcal{O}^- \cup \mathcal{A}^-)$ where on-the-fly expert

Table 2. Revision results for different axiom choosing strategies

	impact$^+$			*guaranteed*			*impact*$^-$			upper bound			rand
S_1	69%	4,677	36,773	48%	11,860	51,677	9%	17,828	46,461	74%	4,110	11,399	45%
S_2	83%	2,584	18,702	65%	8,190	55,273	12%	20,739	67,625	83%	2,645	27,850	60%
S_3	20%	3,137	26,759	43%	3,914	27,629	28%	9,947	46,461	48%	3,509	13,202	31%
S_4	29%	2,198	15,601	43%	3,137	18,367	31%	7,309	10,217	51%	2,177	7,002	31%
S_5	8%	1,778	11,443	39%	1,290	6,647	54%	954	1,438	54%	801	1,989	41%
S_6	13%	9,352	212,041	54%	8,166	99,586	76%	6,797	16,922	76%	5,219	19,861	57%

decisions about approval or decline were simulated according to the membership in $\mathcal{A}^+$ or $\mathcal{A}^-$. For computing the entailments, we used the OWL reasoner HermiT.[2]

For each set, Table 2 shows the effects of the different choice functions *impact*$^+$, *guaranteed* and *impact*$^-$ by measuring the reduction of expert decisions compared to evaluating the whole set manually (1st column for each axiom set and choice function), followed by the number of necessary reasoner calls when using decision spaces (2nd column for each axiom set and choice function) and the corresponding number of reasoner calls without the use of decision spaces (3rd column for each axiom set and choice function). As a baseline, we also include the reduction of expert decisions when choosing axioms randomly (last column). The upper bound for the manual effort reduction was obtained by applying the "impact oracle" function:

$$\text{KnownImpact}(\alpha) = \begin{cases} \text{impact}^+(\alpha) & \text{if } \alpha \in \mathcal{A}^+, \\ \text{impact}^-(\alpha) & \text{if } \alpha \in \mathcal{A}^-. \end{cases}$$

The results of the evaluation show that:

- Decision spaces save on average 75% of reasoner calls, which leads to a considerable overall performance gain given that, on average, 88% of computation time in our experiments is spent within the methods of the reasoner according to our profiling measurements. The experiments with the same datasets took on average 8 times longer without the application of decision spaces.
- Compared to an all manual revision, a significant effort reduction of on average 44% is already achieved when axioms are chosen randomly for each expert decision by automatically approving and declining axioms based on the computed revision closure. However, there is still some space for improvement, since the "impact oracle" manages to reduce the manual effort of revision on average by 64%.
- If the ratio of approved axioms is rather high or rather low, *impact*$^+$ or *impact*$^-$, respectively, perform best.
- If the ratios of approved and declined axioms are more or less equal, the guaranteed impact is the best choice.

From these observations we can conclude that the appropriate axiom choosing strategy has to be selected based on the expected ratio of valid axioms. We see that an application of the most suitable axiom choosing strategy for each validity ratio, listed in grey rows, yields on average an effort reduction of 61%, which is 15% higher than the performance of *random* and only 3% less than the effort reduction achieved by the "impact oracle".

[2] http://www.hermit-reasoner.com

5 Related Work

In our previous work [3], we proposed an approach for determining a beneficial order of axiom evaluation under the assumption of a high validity ratio within the axiom set under investigation. The latter approach aims at reducing the manual effort of revision by eliminating the redundancy within the corresponding axiom set, which is the major factor leading to automatic axiom evaluation under the assumption of a high validity ratio. Prior to the interactive revision, a minimal non-redundant subset of axioms under investigation is identified and then reviewed by the expert thereby not requiring the expensive computation of axiom impacts after each expert decision.

In addition to our own work, we are aware of two approaches for supporting the revision of ontological data based on logical appropriateness: Meilicke et al. [2] and Jiménez-Ruiz et al. [1] propose two approaches, both of which are applied in the context of mapping revision. In these approaches, dependencies between evaluation decisions are determined based on a set of logical criteria, each of which is a subset of the criteria that can be derived from the notion of revision state consistency introduced in Definition 1. Similarly to our approach, Meilicke et al. aim at reducing the manual effort of mapping revision by relying on a heuristic notion of impact. The approach is, however, difficult to generalize to the revision of ontologies since the notion of impact is based on the hypothetically possible number of mapping axioms for two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ and further relies on the assumption that the set of possible mapping axioms is mostly disjoint from the axioms in $\mathcal{O}_1 \cup \mathcal{O}_2$. This assumption is justified in case of mapping revision, since axioms in $\mathcal{O}_1$ ($\mathcal{O}_2$) usually refer only to entities from $\mathcal{O}_1$ ($\mathcal{O}_2$), while mapping axioms link entities from $\mathcal{O}_1$ and $\mathcal{O}_2$. For interactive ontology revision in general, however, the axioms that are to be revised are typically not disjoint from the already evaluated axioms.

The focus of ContentMap [1] lies within the visualization of consequences and user guidance in case of difficult evaluation decisions, while the minimization of the manual and computational effort required for the revision is out of scope. ContentMap selectively materializes and visualizes the logical consequences caused by the axioms under investigation and supports the revision of those consequences. ContentMap requires an exponential number of reasoning operations in the size of the ontology under revision since dependencies between the consequences are determined by comparing their *justifications* (sets of axioms causing the entailment aka minAs). Our approach, however, requires at most a polynomial number of entailment checks.

Another strand of work starting from [7] is related to the overall motivation of enriching knowledge bases with additional expert-curated knowledge in a way that minimizes the workload of the human expert: based on the *attribute exploration* algorithm from formal concept analysis (FCA), several works have proposed structured interactive enumeration strategies of inclusion dependencies or axioms of certain fragments of description logics which then are to be evaluated by the expert. While similar in terms of the workflow, the major difference of these approaches to ours is that the axioms are not pre-specified but created on the fly and therefore, the exploration may require (in the worst case exponentially) many human decisions.

6 Conclusions and Future Work

In this paper, we proposed a methodology for supporting interactive ontology revision based on logical criteria. We stated consistency criteria for revision states and introduced the notion of revision closure, based on which the revision of ontologies can be partially automatized. Even though a significant effort reduction can be achieved when axioms are chosen randomly for each expert decision, choosing an appropriate order usually yields a higher effort reduction. We introduced the notion of axiom impact which is used to determine a beneficial order of evaluation. Depending on the expected ratio of approved axioms, impact^{+}, impact^{-} or the guaranteed impact can be employed in order to achieve a higher effort reduction. In fact, in three out of six cases during the evaluation, the maximum possible effort reduction was achieved when employing the best suitable axiom choosing strategy. Moreover, we provided an efficient and elegant way of determining the revision closure and axiom impact by computing and updating structures called *decision spaces* which saved 75% of reasoner calls during our evaluation.

In our future work, we will investigate how the axiom choosing strategy can be adjusted according to the current ratio of approved axioms. Another open question is how the axioms under investigation can be efficiently partitioned into sets that can be reviewed independently.

Acknowledgments This work is supported by the German Federal Ministry of Education and Research (BMBF) under the SAW-project NanOn, by the German Research Foundation (DFG) under the project ExpresST, and by the EPSRC project HermiT: Reasoning with Large Ontologies.

References

1. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Llavori, R.B.: Ontology integration using mappings: Towards getting the right logical consequences. In: Proc. ESWC 2009. pp. 173–187 (2009)
2. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Supporting manual mapping revision using logical reasoning. In: Proc. AAAI 2008. pp. 1213–1218 (2008)
3. Nikitina, N.: Semi-automatic revision of formalized knowledge. In: Proc. ECAI 2010. pp. 1097–1098 (2010)
4. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. Technical Report 3013, Institut AIFB, KIT, Karlsruhe (April 2011)
5. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proc. IJCAI 2011 (2011), to Appear
6. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at `http://www.w3.org/TR/owl2-overview/`
7. Rudolph, S.: Exploring relational structures via FLE. In: Proc. ICCS 2004. pp. 196–212. Springer (2004)
8. Shearer, R., Horrocks, I.: Exploiting partial information in taxonomy construction. In: Proc. ISWC 2009. pp. 569–584 (2009)

Dependencies to Optimize Ontology Based Data Access[⋆]

Mariano Rodríguez-Muro and Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
{rodriguez,calvanese}@inf.unibz.it

Abstract. Query answering in Ontology Based Data Access (OBDA) exploits the knowledge of an ontology's TBox to deal with incompleteness of the ABox (or data source). Current query-answering techniques with *DL-Lite* require exponential size query reformulations, or expensive data pre-processing. Also, these techniques present severe redundancy issues when dealing with ABoxes that are already (partially) complete. It has been shown that addressing redundancy is not only required for tractable implementations of decision procedures, but may also allow for sizable improvements in execution times. Considering the previous observations, in this paper we extend the results aiming at improving query answering performance in OBDA systems that were developed in [9] for *DL-Lite*$_\mathcal{F}$, to the case where also role inclusions are present in the TBox. Specifically, we first show that we can characterize completeness of an ABox by means of dependencies, and that we can use these to optimize *DL-Lite*$_\mathcal{A}$ TBoxes. Second, we show that in OBDA systems we can create ABox repositories that appear to be complete w.r.t. a significant portion of any *DL-Lite*$_\mathcal{A}$ TBox. The combination of these results allows us to design OBDA systems based on *DL-Lite*$_\mathcal{A}$ in which redundancy is minimal, the exponential aspect of query answering is notably reduced and that can be implemented efficiently using existing RDBMSs.

1 Introduction

The current approaches to Ontology Based Data Access (OBDA) with lightweight Description Logics (DLs) of the *DL-Lite* family [2] rely on query reformulation. These techniques are based on the idea of using the ontology to rewrite a given query into a new query that, when evaluated over the data sources, returns the certain answers to the original query. Experiments with unions of conjunctive queries (UCQs) have shown that reformulations may be very large, and that the execution of these reformulations suffers from poor performance. This triggered the development of alternative reformulation techniques [6,10], in which the focus has been on the reduction of the number of generated queries/rules. These techniques have shown some success, however query reformulation in all of them is still worst-case exponential in the size of the original query. Alternative approaches [5] use the expansion of the extensional layer of the ontology (i.e., the ABox) w.r.t. the intensional knowledge (i.e., the TBox) to avoid query reformulation almost entirely. However, the cost of data expansion imposes severe limitations on the

⋆ This work has been supported by the EU FP7-ICT Project ACSI (257593).

system. We believe that approaching the problem of the high cost of query answering in OBDA systems requires a change of focus: namely, from the ’number of queries’ perspective, to the perspective that takes into account the ’duplication in the answers’ appearing in query results under SQL multiset semantics. Duplication in results is a sign of *redundancy* in the reasoning process; it not only generated not only by the reformulation procedures as traditionally thought, since also techniques based on ABox expansion show this problem. Instead, redundancy is the consequence of ignoring the semantics of the data sources. In particular, when the data in a source (used to populate the ABox of the ontology) already satisfies an inclusion assertion of the TBox (i.e., is *complete* w.r.t. such an inclusion assertion), then using that inclusion assertion during query answering might generate redundant answers [8]. As noted in [4], the runtime of decision procedures might change from exponential to polynomial if redundancy is addressed, and this is also the case in OBDA query answering. In [9], we addressed both problems, redundancy and the exponential blow-up of query reformulations, for the DL *DL-Lite*$_{\mathcal{F}}$. We followed two complementary directions and in this paper we extend both to deal also with the case where role inclusions are present in the TBox.

Specifically, in [9], we first presented an approach to take into account completeness of the data with respect to *DL-Lite*$_{\mathcal{F}}$ TBoxes. We characterized completeness using *ABox dependencies* and showed that it is possible to use dependencies to optimize the TBox in order to avoid redundant computations *independently of the reasoning technique*. Second, we focused on how we can optimally complete ABoxes in OBDA systems by relying on the fact that in OBDA systems it is possible to manipulate not only the data, but also the mappings and database schema. This allows us to conceive procedures to store an ABox in a source in such a way that it *appears* to be complete with respect to a significant portion of the TBox, but without actually *expanding* the data. We presented two such procedures, one for general and one for ’virtual’ OBDA systems, both designed to take advantage of the features of modern RDBMSs effectively. These results allow for the design of systems that can delegate reasoning tasks (e.g., dealing with hierarchies, existentially quantified individuals, etc.) to stages of the reasoning process where these tasks can be handled most effectively. The result is a (sometimes dramatic) reduction of the exponential runtime and an increase in the quality of the answers due to the reduction of duplication. Here, we extend the TBox optimization procedure and one of the ABox completion mechanisms to *DL-Lite*$_{\mathcal{A}}$ ontologies, in which role inclusions are allowed.

The rest of the paper is organized as follows: Section 2 gives technical preliminaries. Section 3 presents our extension to *DL-Lite*$_{\mathcal{A}}$ of the general technique for optimizing TBoxes w.r.t. dependencies. Section 4 introduces data dependencies in OBDA systems, describing why it is natural to expect completeness of ABoxes. Section 5 presents our extension of one of the techniques for completing ABoxes in OBDA systems to allow for *DL-Lite*$_{\mathcal{A}}$ ontologies. Section 6 concludes the paper.

2 Preliminaries

In the rest of the paper, we assume a fixed vocabulary V of *atomic concepts*, denoted A (possibly with subscripts), and *atomic roles*, denoted P, representing unary and binary relations, respectively, and an alphabet Γ of *(object) constants*.

Databases. In the following, we regard a *database (DB)* as a pair $\mathbf{D} = \langle \mathbf{R}, \mathbf{I} \rangle$, where $\mathbf{R}$ is a relational schema and $\mathbf{I}$ is an instance of $\mathbf{R}$. The active domain $\Gamma_{\mathbf{D}}$ of $\mathbf{D}$ is the set of constants appearing in $\mathbf{I}$, which we call *value constants*. An SQL query φ over a DB schema $\mathbf{R}$ is a mapping from a DB instance $\mathbf{I}$ of $\mathbf{R}$ to a set of tuples.

***DL-Lite* ontologies**. We introduce the DL *DL-Lite*$_{\mathcal{A}}$, on which we base our results. In *DL-Lite*$_{\mathcal{A}}$, a *basic role*, denoted R, is an expression of the form P or P^-, and a *basic concept*, denoted B, is an expression of the form A or $\exists R$. An *ontology* is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox. A TBox is a finite set of *(positive) inclusions* $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$, *disjointness assertions* $B_1 \sqsubseteq \neg B_2$, and *functionality assertions* $(\text{funct } R)$. An ABox is a finite set of *membership assertions* $A(c)$ or $P(c, c')$, where $c, c' \in \Gamma$. Moreover, *DL-Lite*$_{\mathcal{A}}$ imposes the syntactic restriction that a role P declared functional, via $(\text{funct } P)$, or inverse functional, via $(\text{funct } P^-)$, cannot be specialized, i.e., cannot appear in the right-hand side of a role inclusion assertion $R \sqsubseteq P$ or $R \sqsubseteq P^-$.

Queries over ontologies. An *atom* is an expression of the form $A(t)$ or $P(t, t')$, where t and t' are *atom terms*, i.e., variables or constants in Γ. An atom is *ground* if it contains no variables. A *conjunctive query* (CQ) q over an ontology $\mathcal{O}$ is an expression of the form $q(\boldsymbol{x}) \leftarrow \beta(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is a tuple of distinct variables, called *distinguished*, $\boldsymbol{y}$ is a tuple of distinct variables not occurring in $\boldsymbol{x}$, called *non-distinguished*, and $\beta(\boldsymbol{x}, \boldsymbol{y})$ is a *conjunction* of atoms with variables in $\boldsymbol{x}$ and $\boldsymbol{y}$, whose predicates are atomic concepts and roles of $\mathcal{O}$. We call $q(\boldsymbol{x})$ the *head* of the query and $\beta(\boldsymbol{x}, \boldsymbol{y})$ its *body*. A *union of CQs* (UCQ) is a set of CQs (called disjuncts) with the same head. Given a CQ Q with body $\beta(\boldsymbol{z})$ and a tuple $\boldsymbol{v}$ of constants of the same arity as $\boldsymbol{z}$, we call a *ground instance* of Q the set $\beta[\boldsymbol{z}/\boldsymbol{v}]$ of ground atoms obtained by replacing in $\beta(\boldsymbol{z})$ each variable with the corresponding constant from $\boldsymbol{v}$.

Semantics. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each constant c an element $c^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, to each atomic concept A a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each atomic role P a binary relation over $\Delta^{\mathcal{I}}$. Moreover, basic roles and basic concepts are interpreted as follows: $(P^-)^{\mathcal{I}} = \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\}$ and $(\exists R)^{\mathcal{I}} = \{o \mid \exists o'.\ (o, o') \in R^{\mathcal{I}}\}$. An interpretation $\mathcal{I}$ is a *model* of $B_1 \sqsubseteq B_2$ if $B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}$, of $R_1 \sqsubseteq R_2$ if $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$, of $B_1 \sqsubseteq \neg B_2$ if $B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}} = \emptyset$, and of $(\text{funct } R)$ if for each $o, o_1, o_2 \in \Delta^{\mathcal{I}}$ we have that $(o, o_1) \in R^{\mathcal{I}}$ and $(o, o_2) \in R^{\mathcal{I}}$ implies $o_1 = o_2$. Also, $\mathcal{I}$ is a model of $A(c)$ if $c^{\mathcal{I}} \in A^{\mathcal{I}}$, and of $P(c, c')$ if $(c^{\mathcal{I}}, c'^{\mathcal{I}}) \in P^{\mathcal{I}}$. In *DL-Lite*$_{\mathcal{A}}$, we adopt the Unique Name Assumption (UNA), which enforces that for each pair of constants o_1, o_2, if $o_1 \neq o_2$, then $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$. For a *DL-Lite*$_{\mathcal{A}}$ assertion α (resp., a set Θ of *DL-Lite*$_{\mathcal{A}}$ assertions), $\mathcal{I} \models \alpha$ (resp., $\mathcal{I} \models \Theta$) denotes that $\mathcal{I}$ is a model of α (resp., Θ). A *model of an ontology* $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$. An ontology is *satisfiable* if it admits a model. An ontology $\mathcal{O}$ *entails* an assertion α, denoted $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ is also a model of α. Similarly, for a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ instead of $\mathcal{O}$. The *saturation* of a TBox $\mathcal{T}$, denoted $sat(\mathcal{T})$, is the set of *DL-Lite*$_{\mathcal{A}}$ assertions α s.t. $\mathcal{T} \models \alpha$. Notice that $sat(\mathcal{T})$ is finite, hence a TBox.

Let $\Gamma_{\mathcal{A}}$ denote the set of constants appearing in an ABox $\mathcal{A}$. The *answer* to a CQ $Q = q(\boldsymbol{x}) \leftarrow \beta(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ in an interpretation $\mathcal{I}$, denoted $ans(Q, \mathcal{O}, \mathcal{I})$, is the set of tuples $\boldsymbol{c} \in \Gamma_{\mathcal{A}} \times \cdots \times \Gamma_{\mathcal{A}}$ such that there exists a tuple $\boldsymbol{c}' \in \Gamma_{\mathcal{A}} \times \cdots \times \Gamma_{\mathcal{A}}$ such that the ground atoms in $\beta[(\boldsymbol{x}, \boldsymbol{y})/(\boldsymbol{c}, \boldsymbol{c}')]$ are true in $\mathcal{I}$. The answer to an UCQ

Q in $\mathcal{I}$ is the union of the answers to each CQ in Q. The *certain answers* to Q in $\mathcal{O}$, denoted $cert(Q, \mathcal{O})$, is the intersection of every $ans(Q, \mathcal{O}, \mathcal{I})$ for all models $\mathcal{I}$ for $\mathcal{O}$. The *answer to Q over an ABox $\mathcal{A}$*, denoted $eval(Q, \mathcal{A})$, is the answers to Q over $\mathcal{A}$ viewed as a DB instance. A *perfect reformulation* of Q w.r.t. a TBox $\mathcal{T}$ is a query Q' such that for every ABox $\mathcal{A}$ such that $\langle\mathcal{T}, \mathcal{A}\rangle$ is satisfiable, $cert(Q, \langle\mathcal{T}, \mathcal{A}\rangle) = eval(Q', \mathcal{A})$.

Mappings. We adopt the definitions for ontologies with mappings from [7]. First we extend interpretations to be able to create object constants from the value constants in a DB $\mathbf{D}$. Given an alphabet Λ of *function symbols* we define the set $\tau(\Lambda, \Gamma_{\mathbf{D}})$ of *object terms* as the set of all terms of the form $\mathbf{f}(d_1, \ldots, d_n)$, where $\mathbf{f} \in \Lambda$, the arity of $\mathbf{f}$ is n, and $d_1, \ldots, d_n \in \Gamma_{\mathbf{D}}$. We set $\Gamma = \Gamma_{\mathbf{D}} \cup \tau(\Lambda, \Gamma_{\mathbf{D}})$, and we extend the interpretation function so that for each $c \in \tau(\Lambda, \Gamma_{\mathbf{D}})$ we have that $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. We extend queries by allowing the use of predicate arguments that are *variable terms*, i.e., expressions of the form $\mathbf{f}(\boldsymbol{t})$, where $\mathbf{f} \in \Lambda$ with arity n and $\boldsymbol{t}$ is an n-tuple of variables or value constants. Given a TBox $\mathcal{T}$ and a DB $\mathbf{D}$, a *mapping (assertion)* m for $\mathcal{T}$ is an expression of the form $\varphi(\boldsymbol{x}) \leadsto \psi(\boldsymbol{t})$ where $\varphi(\boldsymbol{x})$ is an SQL query over $\mathbf{D}$ with answer variables $\boldsymbol{x}$, and $\psi(\boldsymbol{t})$ is a CQ over $\mathcal{T}$ without non-distinguished variables using variable terms over variables in $\boldsymbol{x}$. We call the mapping *simple* if the body of $\psi(\boldsymbol{t})$ consists of a single atom, and *complex* otherwise. A simple mapping *is for* an atomic concept A (resp., atomic role P) if the atom in the body of $\psi(\boldsymbol{t})$ has A (resp., P) as predicate symbol. In the following, we might abbreviate the query ψ in a mapping by showing only its body. A *virtual ABox* $\mathcal{V}$ is a tuple $\langle\mathbf{D}, \mathcal{M}\rangle$, where $\mathbf{D}$ is a DB and $\mathcal{M}$ a set of mappings, and an *ontology with mappings* is a tuple $\mathcal{OM} = \langle\mathcal{T}, \mathcal{V}\rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{V} = \langle\mathbf{D}, \mathcal{M}\rangle$ is a virtual ABox in which $\mathcal{M}$ is a set of mappings for $\mathcal{T}$.

An interpretation $\mathcal{I}$ *satisfies* a mapping assertion $\varphi(\boldsymbol{x}) \leadsto \psi(\boldsymbol{t})$ w.r.t. a DB $\mathbf{D} = \langle\mathbf{R}, \mathbf{I}\rangle$ if for every tuple $\boldsymbol{v} \in \varphi(\mathbf{I})$ and for every ground atom X in $\psi[\boldsymbol{x}/\boldsymbol{v}]$ we have that: if X has the form $A(\mathbf{f}(\boldsymbol{c}))$, then $(\mathbf{f}(\boldsymbol{c}))^{\mathcal{I}} \in A^{\mathcal{I}}$, and if X has the form $P(\mathbf{f_1}(\boldsymbol{c}_1), \mathbf{f_2}(\boldsymbol{c}_2))$, then $((\mathbf{f_1}(\boldsymbol{c}_1))^{\mathcal{I}}, \mathbf{f_2}(\boldsymbol{c}_2)^{\mathcal{I}}) \in P^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{V} = \langle\mathbf{D}, \mathcal{M}\rangle$, denoted $\mathcal{I} \models \mathcal{V}$, if it satisfies every mapping in $\mathcal{M}$ w.r.t. $\mathbf{D}$. A virtual ABox $\mathcal{V}$ *entails* an ABox assertion α, denoted $\mathcal{V} \models \alpha$, if every model of $\mathcal{V}$ is a model of α. $\mathcal{I}$ is a model of $\mathcal{OM} = \langle\mathcal{T}, \mathcal{V}\rangle$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{V}$. As usual, $\mathcal{OM}$ is *satisfiable* if it admits a model. We note that, in an ontology with mappings $\mathcal{OM} = \langle\mathcal{T}, \langle\mathbf{D}, \mathcal{V}\rangle\rangle$, we can always replace $\mathcal{M}$ by a set of *simple* mappings, while preserving the semantics of $\mathcal{OM}$. It suffices to *split* each complex mapping $\varphi \leadsto \psi$ into a set of simple mappings that share the same SQL query φ (see [7]). In the following, we assume to deal only with simple mappings.

Dependencies. ABox dependencies are assertions that restrict the *syntactic* form of allowed ABoxes. In this paper, we focus on unary and binary inclusion dependencies only. A *unary (resp., binary) inclusion dependency* is an assertion of the form $B_1 \sqsubseteq_A B_2$, where B_1 and B_2 are basic concepts (resp., $R_1 \sqsubseteq_A R_2$, where R_1 and R_2 are basic roles). In the following, for a basic role R and constants c, c', $R(c, c')$ stands for $P(c, c')$ if $R = P$ and for $P(c', c)$ if $R = P^-$. An ABox $\mathcal{A}$ *satisfies* an inclusion dependency σ, denoted $\mathcal{A} \models \sigma$, if the following holds: *(i)* if σ is $A_1 \sqsubseteq_A A_2$, then for all $A_1(c) \in \mathcal{A}$ we have $A_2(c) \in \mathcal{A}$; *(ii)* if σ is $\exists R \sqsubseteq_A A$, then for all $R(c, c') \in \mathcal{A}$ we have $A(c) \in \mathcal{A}$; *(iii)* if σ is $A \sqsubseteq_A \exists R$, then for all $A(c) \in \mathcal{A}$ there exists c' such that $R(c, c') \in \mathcal{A}$; *(iv)* if σ is $\exists R_1 \sqsubseteq_A \exists R_2$, then for all $R_1(c, c') \in \mathcal{A}$ there exists c'' such that $R_2(c, c'') \in \mathcal{A}$; *(v)* if σ is $R_1 \sqsubseteq_A R_2$, then for all $R_1(c, c') \in \mathcal{A}$ we have $R_2(c, c') \in \mathcal{A}$. An ABox $\mathcal{A}$

satisfies a set of dependencies Σ, denoted $\mathcal{A} \models \Sigma$, if $\mathcal{A} \models \sigma$ for each $\sigma \in \Sigma$. A set of dependencies Σ *entails* a dependency σ, denoted $\Sigma \models \sigma$, if for every ABox $\mathcal{A}$ s.t. $\mathcal{A} \models \Sigma$ we also have that $\mathcal{A} \models \sigma$. The *saturation* of a set Σ of dependencies, denoted $sat(\Sigma)$, is the set of dependencies σ s.t. $\Sigma \models \sigma$.Given two queries Q_1, Q_2, we say that Q_1 *is contained in* Q_2 *relative to* Σ if $eval(Q_1, \mathcal{A}) \subseteq eval(Q_2, \mathcal{A})$ for each ABox $\mathcal{A}$ s.t. $\mathcal{A} \models \Sigma$.

3 Optimizing TBoxes w.r.t. Dependencies

In a *DL-Lite*$_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, the ABox $\mathcal{A}$ may be *incomplete* w.r.t. the TBox $\mathcal{T}$, i.e., there may be assertions $B_1 \sqsubseteq B_2$ in $\mathcal{T}$ s.t. $\mathcal{A} \not\models B_1 \sqsubseteq_A B_2$. When computing the certain answers to queries over $\mathcal{O}$, the TBox $\mathcal{T}$ is used to overcome such incompleteness. However, an ABox may already be (partially) complete w.r.t. $\mathcal{T}$, e.g., an ABox $\mathcal{A}$ satisfying $A_1 \sqsubseteq_A A_2$ is complete w.r.t. $A_1 \sqsubseteq A_2$. While ignoring completeness of an ABox is 'harmless' in the theoretical analysis of reasoning over *DL-Lite*$_{\mathcal{A}}$ ontologies, in practice, it introduces *redundancy*, which manifests itself as *containment w.r.t. dependencies* among the disjuncts (CQs) of the perfect reformulation, making the contained disjuncts *redundant*. For example, let $\mathcal{T}$ and $\mathcal{A}$ be as before, and let Q be $q(x) \leftarrow A_2(x)$, then any perfect reformulation of Q must include $q_1 = q(x) \leftarrow A_1(x)$ and $q_2 = q(x) \leftarrow A_2(x)$ as disjuncts. However, since q_1 is contained in q_2 relative to $A_1 \sqsubseteq_A A_2$, we have that q_1 will not contribute new tuples w.r.t. those contributed by q_2.

It is possible to use information about completeness of an ABox, expressed as a set of dependencies, to avoid redundancy in the reasoning process. One place to do this is during query reformulation, using techniques based on conjunctive query containment (CQC) with respect to dependencies to avoid the generation of redundant queries. However, this approach is expensive, since CQC is an NP-complete problem (even ignoring dependencies), and such optimizations would need to be performed every time a query is reformulated. We show now how we can improve efficiency by pre-processing the TBox before performing reformulation. In particular, given a TBox $\mathcal{T}$ and a set Σ of dependencies, we show how to compute a TBox $\mathcal{T}'$ that is smaller than $\mathcal{T}$ and such that for every query Q the certain answers are preserved if Q is executed over an ABox that satisfies Σ. Specifically, our objective is to determine when an inclusion assertion of $\mathcal{T}$ is *redundant w.r.t.* Σ, and to do so we use the following auxiliary notions.

Definition 1. *Let $\mathcal{T}$ be a TBox, B, C basic concepts, R, S basic roles, and Σ a set of dependencies over $\mathcal{T}$. A $\mathcal{T}$*-chain from B to C in $\mathcal{T}$ *(resp., a Σ*-chain from B to C in Σ*) is a sequence of inclusion assertions $(B_i \sqsubseteq B'_i)_{i=0}^n$ in $\mathcal{T}$ (resp., a sequence of inclusion dependencies $(B_i \sqsubseteq_A B'_i)_{i=0}^n$ in Σ), for some $n \geq 0$, such that: $B_0 = B$, $B'_n = C$, and for $1 \leq i \leq n$, we have that B'_{i-1} and B_i are basic concepts s.t., either (i) $B'_{i-1} = B_i$, or (ii) $B'_{i-1} = \exists R'$ and $B_i = \exists R'^-$, for some basic role R'. A $\mathcal{T}$-chain from R to S in $\mathcal{T}$ (resp., a Σ-chain from R to S in Σ) is a sequence of inclusion assertions $(R_i \sqsubseteq R'_i)_{i=0}^n$ in $\mathcal{T}$ (resp., a sequence of inclusion dependencies $(R_i \sqsubseteq_A R'_i)_{i=0}^n$ in Σ), for some $n \geq 0$, such that: $R_0 = R$, $R'_n = S$ and for $1 \leq i \leq n$, we have that $R'_{i-1} = R_i$.*

Intuitively, when there is a $\mathcal{T}$-chain from B to C, the existence of an instance of B in a model implies the existence of an instance of C. For a Σ-chain, this holds for ABox assertions. We use $\mathcal{T}$-chains and Σ-chains to characterize redundancy as follows.

Definition 2. *Let $\mathcal{T}$ be a TBox, B, C basic concepts, R, S basic roles, and Σ a set of dependencies. The inclusion assertion $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is* directly redundant in $\mathcal{T}$ w.r.t. Σ *if (i) $\Sigma \models B \sqsubseteq_A C$ (resp., $\Sigma \models R \sqsubseteq_A S$) and (ii) for every $\mathcal{T}$-chain $(B_i \sqsubseteq B'_i)_{i=0}^n$ with $B'_n = B$ in $\mathcal{T}$ (resp., for every $\mathcal{T}$-chain $(B_i \sqsubseteq B'_i)_{i=0}^n$ with $B'_n = \exists R$ and for every $\mathcal{T}$-chain $(R_i \sqsubseteq R'_i)_{i=0}^m$ with $R'_m = R$), there is a Σ-chain $(B_i \sqsubseteq_A B'_i)_{i=0}^n$ (resp., a Σ-chain $(B_i \sqsubseteq_A B'_i)_{i=0}^n$ and a Σ-chain $(R_i \sqsubseteq_A R'_i)_{i=0}^m$). Then, $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is* redundant in $\mathcal{T}$ w.r.t. Σ *if (a) it is directly redundant, or (b) there exists $B' \neq B$ (resp., $R' \neq R$) s.t. (i) $\mathcal{T} \models B' \sqsubseteq C$ (resp., $\mathcal{T} \models R' \sqsubseteq S$), (ii) $B' \sqsubseteq C$ (resp., $R' \sqsubseteq S$) is not directly redundant in $\mathcal{T}$ w.r.t. Σ, and (iii) $B \sqsubseteq B'$ (resp., $R \sqsubseteq R'$) is directly redundant. in $\mathcal{T}$ w.r.t. Σ.*

Given a TBox $\mathcal{T}$ and a set of dependencies Σ, we apply our notion of redundancy w.r.t. Σ to the assertions in the saturation of $\mathcal{T}$ to obtain a TBox $\mathcal{T}'$ that is equivalent to $\mathcal{T}$ for certain answer computation.

Definition 3. *Given a TBox $\mathcal{T}$ and a set of dependencies Σ over $\mathcal{T}$, the* optimized version of $\mathcal{T}$ w.r.t. Σ, *denoted $optim(\mathcal{T}, \Sigma)$, is the set of inclusion assertions $\{\alpha \in sat(\mathcal{T}) \mid \alpha$ is not redundant in $sat(\mathcal{T})$ w.r.t. $sat(\Sigma)\}$.*

Correctness of using $\mathcal{T}' = optim(\mathcal{T}, \Sigma)$ instead of $\mathcal{T}$ when computing the certain answers to a query follows from the following theorem.

Theorem 1. *Let $\mathcal{T}$ be a TBox and Σ a set of dependencies over $\mathcal{T}$. Then for every ABox $\mathcal{A}$ such that $\mathcal{A} \models \Sigma$ and every UCQ Q over $\mathcal{T}$, we have that $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(Q, \langle optim(\mathcal{T}, \Sigma), \mathcal{A} \rangle)$.*

Proof. First we note that during query answering, only the positive inclusions are relevant, hence we ignore disjointness and functionality assertions. Since $sat(\mathcal{T})$ adds to $\mathcal{T}$ only entailed assertions, $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(Q, \langle sat(\mathcal{T}), \mathcal{A} \rangle)$, for every Q and $\mathcal{A}$, and we can assume w.l.o.g. that $\mathcal{T} = sat(\mathcal{T})$. Moreover, $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle)$ is equal to the evaluation of Q over $chase(\mathcal{T}, \mathcal{A})$. (We refer to [2] for the definition of chase for a *DL-Lite*$_{\mathcal{F}}$ ontology.) Hence it suffices to show that for every $B \sqsubseteq C$ (resp., $R \sqsubseteq R$) that is redundant with respect to Σ, $chase(\mathcal{T}, \mathcal{A}) = chase(\mathcal{T} \setminus \{B \sqsubseteq C\}, \mathcal{A})$. We show this by proving that if $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is redundant (hence, removed by $optim(\mathcal{T}, \Sigma)$), then there is always a $chase(\mathcal{T}, \mathcal{A})$ in which $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is never applicable. Assume by contradiction that $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is applicable to some assertion $B(c)$ (resp., $R(c, c')$) during some step in $chase(\mathcal{T}, \mathcal{A})$. We distinguish two cases that correspond to the cases of Definition 2.

(a) Case where $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is directly redundant, and hence $\Sigma \models B \sqsubseteq_A C$ (resp., $\Sigma \models R \sqsubseteq_A S$). We distinguish two subcases: *(i)* $B(c) \in \mathcal{A}$ (resp., $R(c, c') \in \mathcal{A}$). Since $\mathcal{A} \models B \sqsubseteq_A C$ (resp., $\mathcal{A} \models R \sqsubseteq_A S$), we have $C(c) \in \mathcal{A}$ (resp., $S(c, c') \in \mathcal{A}$), and hence $B \sqsubseteq C$ is not applicable to $B(c)$ (resp., $R \sqsubseteq S$ is not applicable to $R(c, c')$). Contradiction. *(ii)* $B(c) \notin \mathcal{A}$ (resp., $R(c, c') \notin \mathcal{A}$). Then there is a sequence of chase steps starting from some ABox assertion $B'(c')$ (resp., $R(a, a')$ or $B(a)$) that generates $B(c)$ (resp., $R(c, c')$). Such a sequence requires a $\mathcal{T}$-chain $(B_i \sqsubseteq B'_i)_{i=0}^n$ with $B_0 = B'$ and $B'_n = B$ (resp., a $\mathcal{T}$-chain $(R_i \sqsubseteq R'_i)_{i=0}^n$ with $R_0 = R'$ and $R'_n = R$, or a $\mathcal{T}$-chain $(B_i \sqsubseteq B'_i)_{i=0}^n$ with $B_0 = B$ and $B'_n = \exists R$), such that each $B_i \sqsubseteq B'_i$

(resp., each $R_i \sqsubseteq R'_i$ or each $B_i \sqsubseteq B'_i$) is applicable in $chase(\mathcal{T}, \mathcal{A})$. Then, by the second condition of direct redundancy, there is a Σ-chain $(B_i \sqsubseteq_A B'_i)_{i=0}^n$ (resp., a Σ-chain $(R_i \sqsubseteq_A R'_i)_{i=0}^n$ or a Σ-chain $(B_i \sqsubseteq_A B'_i)_{i=0}^n$). Since $\mathcal{A} \models B_0 \sqsubseteq_A B'_0$ (resp., $\mathcal{A} \models R_0 \sqsubseteq R'_0$ or $\mathcal{A} \models B_0 \sqsubseteq_A B'_0$) we have that $B'_0(c') \in \mathcal{A}$ (resp., $R'_0(a, a') \in \mathcal{A}$ or $B'_0(a) \in \mathcal{A}$) and hence $B_0 \sqsubseteq B'_0$ is not applicable to $B'(c')$ (resp., $R_0 \sqsubseteq R'_0$ is not applicable to $R'(a, a')$, or $B_0 \sqsubseteq B'_0$ is not applicable to $B'(a)$). Contradiction.

(b) Case where $B \sqsubseteq C$ has been removed by Definition 2(b), and hence there exists $B' \neq B$ such that $\mathcal{T} \models B \sqsubseteq B'$ (resp., $R' \neq R$ s.t. $\mathcal{T} \models R \sqsubseteq R'$). First we note that any two oblivious chase sequences for $\mathcal{T}$ and $\mathcal{A}$ produce results that are equivalent w.r.t. query answering. Then it is enough to show that there exists some $chase(\mathcal{T}, \mathcal{A})$ in which $B' \sqsubseteq C$ (resp., $R' \sqsubseteq S$) is always applied before $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) and in which $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is never applicable. Again, we distinguish two subcases: *(i)* $B(c) \in \mathcal{A}$ (resp., $R(c, c') \in \mathcal{A}$). Then, since $B \sqsubseteq B'$ is directly redundant, we have that $\Sigma \models B \sqsubseteq_A B'$. Since $\mathcal{A} \models \Sigma$, we have that $B'(c) \in \mathcal{A}$ (resp., $R'(c, c') \in \mathcal{A}$), and given that $B' \sqsubseteq C$ (resp., $R' \sqsubseteq S$) is always applied before $B \sqsubseteq C$ (resp., $R \sqsubseteq S$), $C(c)$ (resp., $S(c, c')$) is added to $chase(\mathcal{T}, \mathcal{A})$ before the application of $B \sqsubseteq C$ (resp., $R \sqsubseteq S$), hence $B \sqsubseteq C$ (resp., $R \sqsubseteq S$) is in fact not applicable. Contradiction. *(ii)* $B(c) \notin \mathcal{A}$ (resp., $R(c, c') \notin \mathcal{A}$). Then, arguing as in Case *(a).(ii)*, using $B \sqsubseteq B'$ instead of $B \sqsubseteq C$ (resp., $R \sqsubseteq R'$ instead of $R \sqsubseteq S$), we can derive a contradiction. □

Complexity and implementation. Due to space limitations, we cannot provide a full description of how to compute $optim(\mathcal{T}, \Sigma)$. We just note that the checks that are required by $optim(\mathcal{T}, \Sigma)$ can be reduced to computing reachability between two nodes in a DAG that represents the reachability relation of the chains in $\mathcal{T}$ and Σ. This operation can be done in linear time.

Consistency checking. Consistency checking may also suffer from redundancy when the ABox is already (partially) complete w.r.t. $\mathcal{T}$. In this case, we need to consider, in addition to inclusion dependencies, also *functional* and *disjointness* dependencies. Due to spaces limitations we cannot provide more details, and just note that using these dependencies it is possible to extend the definitions to generate TBoxes that avoid redundant consistency checking operations.

4 Dependencies in OBDA Systems

The purpose of the current section is to complement our argument w.r.t. completeness of ABoxes by discussing when and why we can expect completeness in OBDA systems. We start by observing that in OBDA systems, ABoxes are constructed, in general, from existing data that resides in some form of data repository. In order to create an ABox, the system requires some form of mappings from the source to the ontology. These may be explicit logical assertions as the ones used in this paper, or they may be implicitly defined through application code. Therefore, the source queries used in these mappings become crucial in determining the structure of the ABox. In particular, any dependencies that hold over the results of these queries will be reflected in the OBDA system as ABox dependencies.

Example 1. Let $\mathbf{R}$ be a DB schema with the relation schema *employee* with attributes *id*, *dept*, and *salary*, that stores information about employees, their salaries, and the department they work for. Let $\mathcal{M}$ be the following mappings:

$$
\begin{array}{ll}
\texttt{SELECT id,dept FROM employee} \rightsquigarrow & \mathit{Employee}(\mathbf{emp}(\texttt{id})) \land \\
 & \mathit{WORKS\text{-}FOR}(\mathbf{emp}(\texttt{id}), \mathbf{dept}(\texttt{dept})) \\
\texttt{SELECT id,dept FROM employee} \rightsquigarrow & \mathit{Manager}(\mathbf{emp}(\texttt{id}))\land \\
\texttt{WHERE salary > 1000} & \mathit{MANAGES}(\mathbf{emp}(\texttt{id}), \mathbf{dept}(\texttt{dept}))
\end{array}
$$

where *Employee* and *Manager* are atomic concepts and *WORKS-FOR* and *MANAGES* are atomic roles. Then for *every* instance $\mathbf{I}$ of $\mathbf{R}$, the virtual ABox $\mathcal{V} = \langle\langle \mathbf{R}, \mathbf{I}\rangle, \mathcal{M}\rangle$ satisfies the following dependencies:

$$
\begin{array}{lll}
\mathit{Manager} \sqsubseteq_A \mathit{Employee} & \mathit{Manager} \sqsubseteq_A \exists \mathit{MANAGES} & \exists \mathit{WORKS\text{-}FOR} \sqsubseteq_A \mathit{Employee} \\
 & \exists \mathit{MANAGES} \sqsubseteq_A \mathit{Manager} & \mathit{Employee} \sqsubseteq_A \exists \mathit{WORKS\text{-}FOR}
\end{array}
$$

In particular, the dependency in Column 1 follows from the containment relation between the two SQL queries used in the mappings, and the remaining dependencies follow from the fact that we populate *WORKS-FOR* (resp., *MANAGES*) using the same SQL query used to populate *Employee* (resp., *Manager*). ■

Turning our attention to the semantics of the data sources, we note that any given DB is based on some conceptual model. At the same time, if we associate the data of any given DB to the concepts and roles of a TBox $\mathcal{T}$, it follows that this data is *semantically related* to these concepts and roles, and that the conceptual model of the DB has some common aspects with the semantics of $\mathcal{T}$. It is precisely these common aspects that get manifested as dependencies between queries in the mappings and that give rise to completeness in ABoxes. Therefore, the degree of completeness of an ABox in an OBDA system is in direct relation with the closeness of the semantics of the conceptual model of the DB and the semantics of the TBox, and with the degree in which the DB itself complies to the conceptual model that was used to design it.

Example 2. To illustrate the previous observations we extend Example 1. First we note that the intended meaning of the data stored in $\mathbf{R}$ is as follows: *(i)* employees with a salary higher than 1,000 are managers, *(ii)* managers *manage* the department in which they are employed, and *(iii)* every employee works for a department. Then, any TBox that shares some of this semantics will present redundancy. For example, if $\mathcal{T}$ is

$$
\begin{array}{lll}
\mathit{Manager} \sqsubseteq \mathit{Employee} & \mathit{Manager} \sqsubseteq \exists \mathit{MANAGES} & \mathit{Employee} \sqsubseteq \exists \mathit{WORKS\text{-}FOR} \\
 & \exists \mathit{MANAGES}^- \sqsubseteq \mathit{Department} & \exists \mathit{WORKS\text{-}FOR}^- \sqsubseteq \mathit{Department}
\end{array}
$$

then the first row of assertions is redundant w.r.t. Σ. Instead, the semantics of the assertions of the second row is not captured by the mappings. In an OBDA system with such components, we should reason only w.r.t. *Department*. This can be accomplished by optimizing $\mathcal{T}$ w.r.t. Σ using the technique presented in Section 3. ■

5 Dependency Induction

We focus now on procedures to complete ABoxes with respect to TBoxes. The final objective is to simplify reasoning by diverting certain aspects of the process (e.g., dealing with concept/role hierarchies and domain and range assertions) from the query reformulation stage to other stages of the query answering process where they can be handled more efficiently. We call these procedures *dependency induction procedures* since their result can be characterized by a set of dependencies that hold in the ABox(es) of the system. Formally, given an OBDA system $\mathcal{O} = \langle \mathcal{T}, \mathcal{V} \rangle$, where $\mathcal{V} = \langle \langle \mathbf{R}, \mathbf{I} \rangle, \mathcal{M} \rangle$, we call a *dependency induction procedure* a procedure that uses $\mathcal{O}$ to compute a virtual ABox $\mathcal{V}'$ such that the number of assertions in $\mathcal{T}$ for which $\mathcal{V}'$ is complete is higher than those for $\mathcal{V}$. An example of a dependency induction procedure is *ABox expansion*, a procedure in which the data in $\mathbf{I}$ is *chased* w.r.t. $\mathcal{T}$. The critical point in dependency induction procedures is the trade-off between the degree of completeness induced, the system's performance, and the cost of the procedure. In [9] we presented two dependency induction mechanism that provide good trade-offs. Both of them are designed for the case in which the data sources are RDBMSs. In the current paper we extend one of these procedures, the *semantic index* technique, to the *DL-Lite*$_{\mathcal{A}}$ setting. In particular, given a *DL-Lite*$_{\mathcal{A}}$ TBox $\mathcal{T}$ and a virtual ABox $\mathcal{V}$, the extended semantic index is able to generate a virtual ABox $\mathcal{V}'$ where, if $\mathcal{T} \models B \sqsubseteq A$ (resp., $\mathcal{T} \models R_1 \sqsubseteq R_2$), then $\mathcal{V}' \models B \sqsubseteq_A A$ (resp., $\Sigma \models R_1 \sqsubseteq_A R_2$). Hence, $\mathcal{V}'$ is complete for all *DL-Lite*$_{\mathcal{A}}$ inferences except those involving mandatory participation assertions, e.g., $B \sqsubseteq \exists R$.

Semantic Index. This technique applies in the context of general OBDA systems in which we are free to manipulate *any* aspect of the system to improve query answering. The basic idea is to encode the implied *is-a* relationships of $\mathcal{T}$ in the values of *numeric indexes* that we assign to concept and role names. ABox membership assertions are then inserted in the DB using these numeric values s.t. one can retrieve most of the implied instances of any concept or role by posing simple range queries to the DB (which are very efficient in modern RDBMSs). Our proposal is related to techniques for managing large transitive relations in knowledge bases (e.g., the is-a hierarchy) [1], however, our interest is not in managing hierarchies but in querying the associated instance data. Our proposal is also related to a technique for XPath query evaluation known as *Dynamic Intervals* [3], however, while the latter deals with XML trees, we have to deal with hierarchies that are DAGs. Formally, a semantic index is defined as follows.

Definition 4. *Given a DL-Lite*$_{\mathcal{A}}$ *TBox* $\mathcal{T}$ *and its vocabulary* V*,* a semantic index for $\mathcal{T}$ *is a pair of mappings* $\langle idx, range \rangle$ *with* $idx : V \to \mathbb{N}$ *and* $range : V \to 2^{\mathbb{N} \times \mathbb{N}}$*, such that, for each pair* E_1*,* E_2 *of atomic concepts or atomic roles in* V*, we have that* $\mathcal{T} \models E_1 \sqsubseteq E_2$ *iff there is a pair* $\langle \ell, h \rangle \in range(E_2)$ *such that* $\ell \leq idx(E_1) \leq h$*.*

Using a semantic index $\langle idx, range \rangle$ for a TBox $\mathcal{T}$, we construct $\mathcal{V} = \langle \mathbf{R}, \mathbf{I} \rangle$ with the completeness properties described above by proceeding as follows. We define a DB schema $\mathbf{R}$ with a universal-like relation $T_C[\mathtt{c1}, \mathtt{idx}]$ for storing ABox concept assertions, and a relation $T_R[\mathtt{c1}, \mathtt{c2}, \mathtt{idx}]$ for storing ABox role assertions, s.t. `c1` and `c2` have type *constant* and `idx` has type *numeric*. Given an ABox $\mathcal{A}$, we construct $\mathbf{I}$ such that for each $A(c) \in \mathcal{A}$ we have $\langle c, idx(A) \rangle \in T_C$ and for each $P(c, c') \in \mathcal{A}$ we have $\langle c, c', idx(P) \rangle \in T_R$. The schema and the index allow us to define, for each atomic concept A and each atomic role P, a set of range queries over $\mathbf{D}$ that retrieves most constants c, c' such that $\mathcal{O} \models A(c)$ or $\mathcal{O} \models P(c, c')$. E.g., if $range(A) = \{\langle 2, 35 \rangle\}$, we define `'SELECT c1 FROM` T_C `WHERE idx >= 2 AND idx <= 35'`. We use these queries to define the mappings of the system as follows[1]: *(i)* for each atomic concept A and each $\langle \ell, h\rangle \in \mathit{range}(A)$, we add the mapping $\sigma_{\ell \leq idx \leq h}(T_C) \leadsto A(c_1)$; *(ii)* for each atomic role P and each $\langle \ell, h\rangle \in \mathit{range}(P)$, we add the mapping $\sigma_{\ell \leq idx \leq h}(T_R) \leadsto P(c_1, c_2)$; *(iii)* for each pair of atomic roles P, P' such that $\mathcal{T} \models P'^{-} \sqsubseteq P$ and each $\langle \ell, h\rangle \in \mathit{range}(P')$ we add the mapping $\sigma_{\ell \leq idx \leq h}(T_R) \leadsto P(c_2, c_1)$; *(iv)* for each atomic concept A, each atomic role P s.t. $\mathcal{T} \models \exists P \sqsubseteq A$ (resp., $\exists P^- \sqsubseteq A$) and each $\langle \ell, h\rangle \in \mathit{range}(P)$, we add the mapping $\sigma_{\ell \leq idx \leq h}(T_R) \leadsto A(c_1)$ (resp., $\sigma_{\ell \leq idx \leq h}(T_R) \leadsto A(c_2)$); *(v)* last, we replace any pair of mappings $\sigma_{\ell \leq idx \leq h}(T_C) \leadsto A(c_1)$ and $\sigma_{\ell' \leq idx \leq h'}(T_C) \leadsto A(c_1)$ such that $\ell' \leq h$ and $\ell \leq h'$ by the mapping $\sigma_{\min(\ell,\ell') \leq idx \leq \max(h,h')}(T_C) \leadsto A(c_1)$ (similarly for role mappings).

A semantic index can be trivially constructed by assigning to each concept and role a unique (arbitrary) value and a set of ranges that covers all the values of their subsumees. However, this is not effective for optimizing query answering since the size of $\mathcal{M}$ determines exponentially the size of the final SQL query. To avoid an exponential blow-up, we create $\langle \mathit{idx}, \mathit{range}\rangle$ using the implied concept and role hierarchy as follows.

Let $\mathcal{T}$ be a TBox, and D_C the minimal DAG that represents the *implied is-a* relation between all atomic concepts of $\mathcal{T}$ (i.e., the *transitive reducts* of the concept hierarchy)[2]. Then we can construct idx by initializing a counter $i = 0$, and visiting the nodes in D_C in a depth-first fashion starting from the root nodes. At each step and given the node N visited at that step, if $\mathit{idx}(N)$ is undefined, set $\mathit{idx}(N) = i$ and $i = i+1$, else if $\mathit{idx}(N)$ is defined, backtrack until the next node for which idx is undefined. Now, to generate range we visit the nodes in D_C starting from the leafs and going up. For each node N in the visit, if N is a leaf in D_C, then we set $\mathit{range}(N) = \{\langle \mathit{idx}(N), \mathit{idx}(N)\rangle\}$, and if N is not a leaf, then we set $\mathit{range}(N) = \mathit{merge}(\{\langle \mathit{idx}(N), \mathit{idx}(N)\rangle\} \cup \bigcup_{N_i \mid N_i \rightarrow N \in D_C} \mathit{range}(N_i))$, where merge is a function that, given a set r of ranges, returns the minimal set r' of ranges that has equal coverage as r, e.g., $\mathit{merge}(\{\langle 5,7\rangle, \langle 3,5\rangle, \langle 9,10\rangle\}) = \{\langle 3,7\rangle, \langle 9,10\rangle\}$. We proceed exactly in the same way with the DAG D_R representing the role hierarchy.

Example 3. Let A, B, C, D be atomic concepts, let R, S, M be atomic roles, and consider the TBox $\mathcal{T} = \{B \sqsubseteq A, C \sqsubseteq A, C \sqsubseteq D, D \sqsubseteq \exists R, \exists R \sqsubseteq D, S \sqsubseteq R, M \sqsubseteq R\}$. Let the DAGs D_C and D_R for $\mathcal{T}$ be the ones depicted in Fig. 1. The technique generates idx and range as indicated in Fig. 1, generates the mappings in Fig. 3, and for any ABox, the technique generates a virtual ABox $\mathcal{V}$ that satisfies all the dependencies Σ in Fig. 2. Then, we will have that in query answering, rewriting is only necessary w.r.t. $D \sqsubseteq \exists R$, which would be the output of $\mathit{optim}(\mathcal{T}, \Sigma)$. ■

Our evaluation of the semantic index technique, described in [9], shows its effectiveness in improving the cost and efficiency of query answering.

6 Conclusions and Future Work

In this paper we focused on issues of redundancy and performance in OBDA systems. Several directions can be taken starting from the ideas presented here. First, although

[1] Here we use relational algebra expressions instead of SQL to simplify the exposition.

[2] We assume w.l.o.g. that $\mathcal{T}$ does not contain a cyclic chain of basic concept or role inclusions. Under such an assumption D_C is unique.

$$\begin{tikzcd}
& 1, \{(1,3)\} & & 4, \{(3,4)\} & & 5, \{(5,7)\} & \\
& A & & D & & R & \\
B \arrow[ur] & & C \arrow[ul] \arrow[ur] & & S \arrow[ur] & & M \arrow[ul] \\
2, \{(2,2)\} & & 3, \{(3,3)\} & & 6, \{(6,6)\} & & 7, \{(7,7)\}
\end{tikzcd}$$

Fig. 1. D_C, D_R and the values for idx and $range$.

$$\begin{array}{ll}
B \sqsubseteq_A A & C \sqsubseteq_A A \\
C \sqsubseteq_A D & \exists R \sqsubseteq_A D \\
S \sqsubseteq_A R & M \sqsubseteq_A R
\end{array}$$

Fig. 2. The induced dependencies.

$$\begin{array}{ll}
\sigma_{1 \leq idx \leq 3}(T_C) \rightsquigarrow A(c_1) & \sigma_{5 \leq idx \leq 7}(T_R) \rightsquigarrow R(c_1, c_2) \\
\sigma_{2 \leq idx \leq 2}(T_C) \rightsquigarrow B(c_1) & \sigma_{6 \leq idx \leq 6}(T_R) \rightsquigarrow S(c_1, c_2) \\
\sigma_{3 \leq idx \leq 3}(T_C) \rightsquigarrow C(c_1) & \sigma_{7 \leq idx \leq 7}(T_R) \rightsquigarrow M(c_1, c_2) \\
\sigma_{3 \leq idx \leq 4}(T_C) \rightsquigarrow D(c_1) & \sigma_{5 \leq idx \leq 7}(T_R) \rightsquigarrow D(c_1)
\end{array}$$

Fig. 3. The mappings created by the technique.

TBox pre-processing is the best place to first address redundancy, it is also necessary to apply redundancy elimination during reasoning, e.g., during query rewriting. Second, redundancy may also appear during consistency checking, i.e., when an ABox is sound w.r.t. to the TBox; this can also be characterized with dependencies. With respect to evaluation, in [9] we presented a preliminary experiments that show that the semantic index technique can provide excellent performance with a fraction of the cost of ABox expansion, however, further experimentation is still required. In particular, it is necessary to provide a comprehensive benchmarks of the techniques discussed in this paper in comparison to other proposals. We are also exploring the context of SPARQL queries over RDFS ontologies; here we believe that our techniques can be used to provide high-performance SPARQL end points with sound and complete RDFS entailment regime support, without relying on inference materialization, as is usually done.

References

1. R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proc. of ACM SIGMOD*, pages 253–262, 1989.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
3. D. DeHaan, D. Toman, M. P. Consens, and M. T. Özsu. A comprehensive XQuery to SQL translation using dynamic interval encoding. In *Proc. of ACM SIGMOD*, 2003.
4. G. Gottlob and C. G. Fermüller. Removing redundancy from a clause. *Artif. Intell.*, 61, 1993.
5. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of KR 2010*, 2010.
6. H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. of Applied Logic*, 8(2):186–209, 2010.
7. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
8. M. Rodríguez-Muro. *Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics*. PhD thesis, KRDB Research Centre, Free Univ. of Bozen-Bolzano, 2010.
9. M. Rodriguez-Muro and D. Calvanese. Dependencies: Making ontology based data access work in practice. In *Proc. of AMW 2011*, 2011.
10. R. Rosati and A. Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of KR 2010*, 2010.

Contextual Representation and Reasoning with Description Logics

Luciano Serafini[1] and Martin Homola[1,2]

[1] Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
[2] Comenius University, Faculty of Mathematics, Physics and Informatics, Mlynská dolina, 84248 Bratislava, Slovakia
serafini@fbk.eu, homola@fmph.uniba.sk

1 Introduction

Despite most of the information available in the Semantic Web (SW) is context dependent, there is a lack of mechanism to qualify knowledge with the context in which it is supposed to hold. In the current practice, contextual information is often crafted in the ontology identifier or in the annotations, non of which affects reasoning. Extensions of the SW languages with specific mechanisms that allow to qualify knowledge, e.g., w.r.t. its provenance [6] or w.r.t. time and events [17], were proposed. Among other works that offer possible solutions [8, 22, 13], the most interesting are $\mathcal{ALC}_{\mathcal{ALC}}$ [14] and Metaview [24], however, a widely accepted approach has not yet been reached.

Instead of extending the current SW languages, we propose a shift of approach: to adapt the theories of context proposed by McCarthy [18] and well studied in AI [7, 15, 3]. We adopt the context-as-a-box metaphor [3] to represent context, in which a context is seen as a "box" containing knowledge in form of logical statements, whose boundaries are determined with contextual attributes (called *dimensions*) qualifying the knowledge inside the box. An example context representing knowledge about football in Italy in year 2010 is depicted in Fig. 1. We will most often rely on three dimensions: time, location and topic; but others were considered as well [15].

time = 2010, location = Italy, topic = football

$$\boxed{\begin{array}{l} \mathsf{Team} \sqsubseteq {=}22\mathsf{has\_player}.\mathsf{Player} \\ \mathsf{Player} \sqsubseteq {\leqslant}1\mathsf{plays\_for}.\mathsf{Team} \\ \mathsf{Team}(\mathsf{Milan}) \\ \mathsf{plays\_for}(\mathsf{Cassano}, \mathsf{Milan}) \\ \cdots \end{array}}$$

Fig. 1. Italian national football league under the context as a box metaphor

To clarify the requirements for contextual representation in the SW, consider a scenario from the domain of football. Knowledge will be qualified with time, location, and the following topics: football (FB), FIFA world cups (FWC), national football leagues (NFL), world news (WN), and national news (NN). Suppose that all information about FWC and NFL should be included in FB, and for each nation all facts about its NFL should be included in its NN. Also all information about FWC should be included in WN. On the other hand, only a part of information about NFL should be included in WN (only that of worldwide interest). A well designed contextual representation formalism should support the following requirements:

knowledge about context: knowledge about contexts such as contextual dimensions and relations between contexts as for instance that one context is more specific than some other, should be explicitly represented and reasoned about. For example, we should be able to assert that the context of FWC in 2010 is more specific than the context of FB and WN in the same year;

contextually bounded facts: in each context we should be able to state facts with local effect that do not necessarily propagate everywhere, e.g., an axiom like "a player is a member of only one team" should be true in some contexts (e.g., FWC, NFL, for each year) but not in more general contexts like FB;

reuse/lifting of facts: be able to include "automatically" all the information contained in more specific contexts. For example, facts in FWC should be lifted up into the WN, and FB. This lifting should be done without spoiling locality of knowledge;

overlapping and varying domains: objects can be present in multiple contexts, but not necessarily in all contexts, e.g., a player can exist in both the FWC context and in the NFL contexts, but many players present in NFL will not be present in FWC;

inconsistency tolerance: two contexts may possibly contain contradicting facts. For instance NN of Italy could assert that "Cassano is the best player of the world", while at the same time the world news report that "Rooney is the best player of the world", without making the whole system inconsistent;

complexity invariance: the qualification of knowledge by context should not increase the complexity.

Based on these requirements, we propose a framework called *Contextualized Knowledge Repository* (CKR), build on top of the expressive description logic $\mathcal{SROIQ}$[3] [10] that is behind OWL 2 [26]. A CKR knowledge base is composed of DL knowledge bases, called *contexts*, each qualified by a set of contextual attributes that specify the boundaries within which the knowledge base is assumed to be true. Contexts are organized by a hierarchical *coverage* relation that regulates the propagation of knowledge between them. The paper defines the syntax and semantics of CKR; shows that concept satisfiability and subsumption are decidable with the complexity upper bound of 2NEXPTIME (i.e., same as for $\mathcal{SROIQ}$); and finally it provides a sound and complete Natural Deduction calculus that characterizes the propagation of knowledge between contexts. Proofs of our statements are available in [21].

2 Contextualized Knowledge Repository

Logical representation of contextual knowledge is based on two classes of formulae: one class to specify knowledge *within* contexts, and another to predicate *about* contexts. McCarthy [18] proposed to use a unique language for both types of knowledge, namely quantified modal logic. While this is optimal from the representational perspective, it easily leads to undecidability. At the opposite extreme there are approaches such

[3] Although we are able to represent any $\mathcal{SROIQ}$ axioms in CKR, to maintain decidability the framework currently excludes reflexivity and role disjointness axioms. See [21] for discussion.

as multi-context systems [7], distributed [4] or package-based description logics [2], where context structure is fixed and it is not possible to specify knowledge about contexts, which limits their practical applicability. We therefore propose an intermediate approach, by allowing to specify the context structure and properties in a (simple) logical *meta-language*, but avoiding to mix it with the *object-language* used within each context in order to maintain good computational properties.

The meta-language is used to specify context structure. It uses a *meta-vocabulary* Γ, a standard DL vocabulary that contains: (a) a set of individuals called *context identifiers*; (b) a finite set of roles $\mathbf{A} = \{A_1, \ldots, A_n\}$ called *dimensions*; (c) for each dimension $A \in \mathbf{A}$ a set of individuals D_A called *dimensional values* and a role $\prec_A$ called *coverage relation*. The number of dimensions $n = |\mathbf{A}|$ is assumed to be a fixed constant. This will be important in order not to introduce additional complexity blow up. Also, relevant research on contextual dimensions suggests that their number is usually very limited [16]. The meta-assertions of the form $A(\mathcal{C}, d)$ for a context identifier $\mathcal{C}$ and some $d \in D_A$ (e.g., $\mathsf{time}(\mathsf{c0}, 2010)$), state that the value of the dimension A of the context $\mathcal{C}$ is d. The meta-assertions of the form $d \prec_A e$ (e.g., $\mathsf{Italy} \prec_{\mathsf{space}} \mathsf{Europe}$) state that the value d of the dimensions A is covered by the value e. Depending on the dimension, the coverage relation has different intuitive meanings, e.g., if A is space then the coverage relation is topological containment, if A is topic then it is topic specificity.

A (full) *dimensional vector* $\mathbf{d}$ is a set of assignments $\{A_1{:=}d_{A_1}, \ldots, A_n{:=}d_{A_n}\}$, with $d_{A_i} \in D_{A_i}$ for each $1 \leq i \leq n$. Note that d_{A_i} ($e_{A_i}, \ldots$) denotes the actual value that $\mathbf{d}$ ($\mathbf{e}, \ldots$) assigns to the dimension A_i. $\mathfrak{D}_\Gamma$ is the set of all dimensional vectors of Γ. For any $\mathbf{B} \subseteq \mathbf{A}$, $\mathbf{d_B} = \{B{:=}d_B \mid B \in \mathbf{B}\}$ and if $\mathbf{B} \subset \mathbf{A}$, then $\mathbf{d_B}$ is called *partial dimensional vector*. Note that $\mathbf{d_A} = \mathbf{d}$. Given two (partial) dimensional vectors $\mathbf{d_B}$ and $\mathbf{e_C}$, the completion of $\mathbf{d_B}$ w.r.t. $\mathbf{e_C}$ is $\mathbf{d_B}{+}\mathbf{e_C} = \mathbf{d_B} \cup \{(A{:=}e_A) \in \mathbf{e_C} \mid A \notin \mathbf{B}\}$.

The object-language is used to specify knowledge inside the contexts. It uses an *object-vocabulary*, obtained from any standard DL vocabulary Σ (containing individuals, concepts, and roles) by closing it w.r.t. what we call *concept/role qualification*. That is, for every concept/role symbol X of Σ and every (partial) dimensional vector $\mathbf{d_B}$, a new concept/role symbol $X_{\mathbf{d_B}}$, called the *qualification of X w.r.t. $\mathbf{d_B}$*, is added to Σ. Qualified symbols are necessary for cross context semantic reference, e.g., the concept of "Italian professor" in the context of France will be formalized by $\mathsf{Professor}_{\mathsf{location:=Italy}}$. If not ambiguous we will omit the attribute name, using e.g. $\mathsf{Professor}_{\mathsf{Italy}}$ instead of $\mathsf{Professor}_{\mathsf{location:=Italy}}$.

Definition 1 (Context). *A* context $\mathcal{C}$ *on the meta/object-vocabulary pair* $\langle\Gamma,\Sigma\rangle$ *is a triple* $\langle \mathrm{id}(\mathcal{C}), \dim(\mathcal{C}), \mathrm{K}(\mathcal{C})\rangle$ *where:*

1. $\mathrm{id}(\mathcal{C})$ *is a* context identifier *of* Γ*;*
2. $\dim(\mathcal{C})$ *is a full dimensional vector of* $\mathfrak{D}_\Gamma$*;*
3. $\mathrm{K}(\mathcal{C})$ *is a DL knowledge base over* Σ*.*

Note that while symbols appearing inside contexts can possibly be qualified with partial dimensional vectors, $\dim(\mathcal{C})$, the dimensional vector of the context $\mathcal{C}$, is never partial. We use the notation $\mathcal{C}_\mathbf{d}$ to denote a context with $\dim(\mathcal{C}) = \mathbf{d}$.

Definition 2 (Contextualized Knowledge Repository). *A* contextualized knowledge repository (CKR) *on a meta/object-vocabulary pair* $\langle\Gamma,\Sigma\rangle$ *is a pair* $\mathfrak{K} = \langle \mathfrak{M}, \mathfrak{C}\rangle$ *where:*

1. *$\mathfrak{C}$ is a set of contexts on $\langle \Gamma, \Sigma \rangle$, one for each context identifier of Γ;*
2. *$\mathfrak{M}$, called meta-knowledge, is a DL knowledge base on Γ where*
   (a) *every $A \in \mathbf{A}$ is a functional role;*
   (b) *for every $\mathcal{C}_{\mathbf{d}} \in \mathfrak{C}$, and every $A \in \mathbf{A}$, $\mathfrak{M} \models A(\mathrm{id}(\mathcal{C}_{\mathbf{d}}), d_A)$;*
   (c) *for every $A \in \mathbf{A}$, the relation $\{d \prec_A d' \mid \mathfrak{M} \models \prec_A(d, d')\}$ is a strict partial order on D_A.*

For a CKR $\mathfrak{K}$, $\mathbf{B} \subseteq \mathbf{A}$, dimensional vectors $\mathbf{d}$, $\mathbf{e}$, and contexts $\mathcal{C}$, $\mathcal{C}'$ we say: (a) $\mathbf{e}$ *covers* $\mathbf{d}$ w.r.t. $\mathbf{B}$ (denoted $\mathbf{d} \prec_{\mathbf{B}} \mathbf{e}$) if $\mathfrak{M} \models \prec_B(d_B, e_B)$ for every $B \in \mathbf{B}$; (b) $\mathbf{e}$ *covers* $\mathbf{d}$ (denoted $\mathbf{d} \prec \mathbf{e}$) if $\mathbf{d} \prec_{\mathbf{A}} \mathbf{e}$; (c) $\mathcal{C}'$ covers $\mathcal{C}$ (denoted $\mathcal{C} \prec \mathcal{C}'$) if $\dim(\mathcal{C}) \prec \dim(\mathcal{C}')$.

If one context covers another it means that its perspective is broader. We will see that this is reflected in the semantics and the domain of the broader context always contains the domain of the narrower context. The coverage induces a hierarchical organization of contexts in each CKR. For instance Fig. 2 depicts the context coverage induced from the following coverage relations between dimensional values:

$$\begin{array}{lll} \mathsf{FWC} \prec_{\mathsf{topic}} \mathsf{WN} & \mathsf{NFL} \prec_{\mathsf{topic}} \mathsf{FB} & \mathsf{africa} \prec_{\mathsf{space}} \mathsf{world} \\ \mathsf{FWC} \prec_{\mathsf{topic}} \mathsf{FB} & \mathsf{NFL} \prec_{\mathsf{topic}} \mathsf{NN} & \mathsf{italy} \prec_{\mathsf{space}} \mathsf{world} \end{array}$$

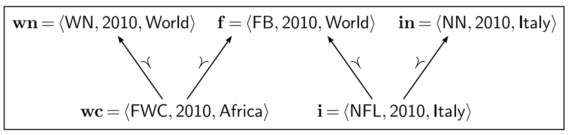


Fig. 2. Coverage relation between contexts

Besides for the coverage relation, which is explicitly expressed in CKR, there are other relations between contexts [3]. We chose to represent the coverage relation because many other relations between contexts can be axiomatized on top of it. For instance the temporal relation between contexts can be axiomatized via GCI axioms in a broader context, e.g., to assert that everyone who is a professor in 2011 typically is a professor also in 2012 (i.e., none of the years covers the other, but instead they are consecutive), we can add the axiom $\mathsf{Professor}_{2011} \sqsubseteq \mathsf{Professor}_{2012}$ into some context that covers both 2011 and 2012 (e.g., one associated with the decade 2011–2020).

A model of a CKR is composed of local models for each context that must satisfy some additional restrictions. Given a CKR $\mathfrak{K}$, a model for a context $\mathcal{C}_{\mathbf{d}}$ is a pair $\mathcal{I}_{\mathbf{d}} = \left\langle \Delta_{\mathbf{d}}, \cdot^{\mathcal{I}_{\mathbf{d}}} \right\rangle$ such that $\mathcal{I}_{\mathbf{d}} \models \mathrm{K}(\mathcal{C}_{\mathbf{d}})$ in the usual DL sense [10] with two exceptions: (a) $\Delta_{\mathbf{d}}$ may also be empty; (b) $\cdot^{\mathcal{I}_{\mathbf{d}}}$ is not required to interpret individuals of Σ that do not occur in $\mathrm{K}(\mathcal{C})$. In the rest of the paper, whenever we write $\phi^{\mathcal{I}_{\mathbf{d}}}$ for any expression ϕ, we will also mean that $\mathcal{I}_{\mathbf{d}}$ is defined on all constants occurring in ϕ.

Definition 3 (CKR Model). *A model of a CKR $\mathfrak{K}$ is a family $\mathfrak{I} = \{\mathcal{I}_{\mathbf{d}}\}_{\mathbf{d} \in \mathfrak{D}_\Gamma}$ of local models such that for all $\mathbf{d}$, $\mathbf{e}$, and $\mathbf{f}$, for every atomic concept A, atomic role R, atomic concept/role X and individual a:*

1. $(\top_{\mathbf{d}})^{\mathcal{I}_{\mathbf{f}}} \subseteq (\top_{\mathbf{e}})^{\mathcal{I}_{\mathbf{f}}}$ *if* $\mathbf{d} \prec \mathbf{e}$
2. $(A_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \subseteq (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}}$
3. $(R_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \subseteq (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}} \times (\top_{\mathbf{f}})^{\mathcal{I}_{\mathbf{d}}}$

4. $a^{\mathcal{I}_\mathbf{d}} = a^{\mathcal{I}_\mathbf{e}}$, *given* $\mathbf{d} \prec \mathbf{e}$, *either if* $a^{\mathcal{I}_\mathbf{d}}$ *is defined,*
   or if $a^{\mathcal{I}_\mathbf{e}}$ *is defined and* $a^{\mathcal{I}_\mathbf{e}} \in \Delta_\mathbf{d}$
5. $(X_{\mathbf{d_B}})^{\mathcal{I}_\mathbf{e}} = (X_{\mathbf{d_B}+\mathbf{e}})^{\mathcal{I}_\mathbf{e}}$
6. $(X_\mathbf{d})^{\mathcal{I}_\mathbf{e}} = (X_\mathbf{d})^{\mathcal{I}_\mathbf{d}}$ *if* $\mathbf{d} \prec \mathbf{e}$
7. $(A_\mathbf{f})^{\mathcal{I}_\mathbf{d}} = (A_\mathbf{f})^{\mathcal{I}_\mathbf{e}} \cap \Delta_\mathbf{d}$ *if* $\mathbf{d} \prec \mathbf{e}$
8. $(R_\mathbf{f})^{\mathcal{I}_\mathbf{d}} = (R_\mathbf{f})^{\mathcal{I}_\mathbf{e}} \cap (\Delta_\mathbf{d} \times \Delta_\mathbf{d})$ *if* $\mathbf{d} \prec \mathbf{e}$
9. $\mathcal{I}_\mathbf{d} \models \mathrm{K}(\mathcal{C}_\mathbf{d})$

The semantics takes care that local domains respect the coverage hierarchy (condition 1). Given contexts $\mathcal{C}_\mathbf{d} \prec \mathcal{C}_\mathbf{e}$, if an individual a occurs in the narrower context then it must be defined also in the broader context with the same meaning; if a only occurs in the broader context however, it does not have to be defined in the narrower one (condition 4). The interpretation of any concept or role qualified with some $\mathbf{f} \in \mathfrak{D}_\Gamma$ is always roofed under $(\top_\mathbf{f})^{\mathcal{I}_\mathbf{d}}$ in any context $\mathcal{C}_\mathbf{d}$, i.e., in a sense $\top_\mathbf{f}$ represents the $\top$ of $\mathcal{C}_\mathbf{f}$ inside $\mathcal{C}_\mathbf{d}$ (conditions 2 and 3). This is always true regardless of the relation between $\mathcal{C}_\mathbf{f}$ and $\mathcal{C}_\mathbf{d}$. If $\mathcal{C}_\mathbf{d} \prec \mathcal{C}_\mathbf{e}$ then the interpretation of any concept and role $X_\mathbf{f}$ in these contexts must be equal modulo the domain of the narrower context (conditions 7 and 8). Treatment of partially qualified symbols is done in condition 5: missing values are always taken from the current context in which the symbol appears. Therefore in the end all symbols (even those with empty qualifying vector) are treated as fully qualified by the semantics. Finally, for each CKR model we require that each local interpretation $\mathcal{I}_\mathbf{d}$ is a model of $\mathcal{C}_\mathbf{d}$ in the usual sense for DL (condition 9). Finally notice that $\bot_\mathbf{d}$ is always interpreted in the empty set. Therefore we can simplify the notation by using just $\bot$.

3 Reasoning in CKR

In the following we devise a proof theoretical characterization of CKR entailment in the natural deduction (ND) style [20], with special focus on the rules for transferring knowledge across contexts. We decide to characterize CKR entailment with ND, since ND provides a clear intuition on how knowledge propagates across contexts. This allows to show interesting properties of CKR reasoning like the fact that (a) in consistent CKR, unconnected contexts do not interact (b) propagation always follows the coverage relation, and other similar properties. ND formalisms also provide a first base for the development of a forward reasoning algorithm, which constitutes a natural extension of the forward local reasoning supported by OWLIM, the platform on top of which a first version of CKR with limited expressive capacity of RDFS has been implemented [9].

We now briefly introduce ND, for more details see [20]. A ND calculus is a set of inference rules of the form:

$$\frac{\alpha_1 \cdots \alpha_n \quad \begin{matrix}[B_{n+1}] \\ \alpha_{n+1}\end{matrix} \quad \cdots \quad \begin{matrix}[B_{n+m}] \\ \alpha_{n+m}\end{matrix}}{\alpha}\rho \tag{1}$$

with $n, m \geq 0$, where α_i and α are formulae and B_i are sets of formulae. The α_i's are the *premises* of ρ, α is the conclusion and the B_i's are the *assumptions discharged* by ρ. A *deduction* of α depending on a set of formulae Φ is a tree rooted in α inductively constructed starting from a set of assumptions in Φ by applying the inference rules. More formally: a formula α is a deduction of α depending on $\{\alpha\}$; if for each $1 \leq i \leq n+m$, Π_i is a deduction of α_i depending on Φ_i and the calculus contains a rule of the form (1), then $\frac{\Pi_1 \cdots \Pi_{n+m}}{\alpha}$ is a deduction of α depending on $(\bigcup_{i=1}^{n} \Phi_i) \cup \left(\bigcup_{i=n+1}^{n+m} (\Phi_i \setminus B_i)\right)$. A formula α is derivable from Φ if there is a deduction of α depending on a subset of Φ.

CKR reasoning tasks are, as in any DL, concept satisfiability and entailment; however in CKR these tasks are relativized w.r.t. a context. A CKR $\mathfrak{K}$ is $\mathbf{d}$-satisfiable (a concept C is $\mathbf{d}$-satisfiable w.r.t. $\mathfrak{K}$) if there exists a model $\mathfrak{I}$ of $\mathfrak{K}$ with $\Delta_{\mathbf{d}} \neq \emptyset$ ($C^{\mathcal{I}_{\mathbf{d}}} \neq \emptyset$). A formula ϕ is $\mathbf{d}$-entailed by $\mathfrak{K}$ (denoted $\mathfrak{K} \models \mathbf{d} : \phi$) if $\mathcal{I}_{\mathbf{d}} \models \phi$ in every model $\mathfrak{I}$ of $\mathfrak{K}$.

Reasoning rules in the ND calculus for CKR allow to deduce conclusions in one of the contexts based on evidence from other contexts, they are therefore a kind of *bridge rules* [7]. As an example consider the following simple bridge rule:

$$\frac{\mathbf{d} : A \sqsubseteq B \quad \mathbf{d} \prec \mathbf{e}}{\mathbf{e} : A_{\mathbf{d}} \sqsubseteq B_{\mathbf{d}}} \tag{2}$$

It implies that whenever $A \sqsubseteq B$ is true in a context $C_{\mathbf{d}}$ such that $\mathbf{d} \prec \mathbf{e}$, then $A_{\mathbf{d}} \sqsubseteq B_{\mathbf{d}}$ should be true in $C_{\mathbf{e}}$. The rule is indeed sound thanks to conditions 5 and 6 of Definition 3 that impose that in any CKR model $\mathfrak{I}$ the interpretation of A and B in $\mathcal{I}_{\mathbf{d}}$ coincide respectively with the interpretations of $A_{\mathbf{d}}$ and $B_{\mathbf{d}}$ in $\mathcal{I}_{\mathbf{e}}$. The rationale of rule (2) is that a statement in a narrower context, can be embedded into a larger context, by applying a transformation that preserves its semantics.

We generalize this idea by introducing the notion of embedding between DL knowledge bases. An embedding is a function that translates expressions from one vocabulary to another in a suitable manner. The input vocabulary Σ will be split into Σ_c (symbols fully specified w.r.t. the current context) and Σ_e (symbols external to the current context) and each of the sets of symbols will be translated differently. More formally: let Σ and Σ' be two DL alphabets, $\Sigma = \Sigma_c \uplus \Sigma_e$, $\top \in \Sigma_c$. A *DL embedding* is a total function $f : \Sigma \to \Sigma'$ that maps individuals, atomic concepts, and atomic roles of Σ to individuals, atomic concepts, and atomic roles of Σ' respectively. For every embedding f the extension f^* that maps complex expressions and axioms over Σ into complex expressions and axioms over Σ' is recursively defined on top of f as given in Table 1.

Two DL-interpretations $\mathcal{I}$ and $\mathcal{I}'$ of Σ and Σ' respectively are said to *comply with* the DL embedding f if: (a) $a^{\mathcal{I}} = f(a)^{\mathcal{I}'}$ for each individual a of Σ such that $a^{\mathcal{I}}$ is defined; (b) $X^{\mathcal{I}} = f(X)^{\mathcal{I}'}$ for each concept/role $X \in \Sigma_c$; (c) $A^{\mathcal{I}} = f(A)^{\mathcal{I}'} \cap f(\top)^{\mathcal{I}'}$ for each concept $A \in \Sigma_e$; (d) $R^{\mathcal{I}} = f(R)^{\mathcal{I}'} \cap f(\top)^{\mathcal{I}'} \times f(\top)^{\mathcal{I}'}$ for each role $R \in \Sigma_e$.

Lemma 1. *If $\mathcal{I}$ and $\mathcal{I}'$ comply with the DL-embedding $f : \Sigma \to \Sigma'$ then: (a) for every concept/role X, $X^{\mathcal{I}} = (f^*(X))^{\mathcal{I}'}$; (b) for every axiom ϕ, $\mathcal{I} \models \phi$ iff $\mathcal{I}' \models f^*(\phi)$.*

The specific embedding that will be instrumental in order to characterize the logical consequence in CKR is now introduced as the $@\mathbf{d}$ operator.

Definition 4 ($@\mathbf{d}$ operator). *Given a CKR $\mathfrak{K}$ over $\langle \Gamma, \Sigma \rangle$, for every $\mathbf{d} \in \mathfrak{D}_\Gamma$, the operator $(\cdot)@\mathbf{d}$ is defined as $f^*_{\mathbf{d}}(\cdot)$, using the embedding $f_{\mathbf{d}}$ of Σ into itself such that: (a) $f_{\mathbf{d}}(a) = a$ for every individual a; (b) $f_{\mathbf{d}}(X_{\mathbf{d}'_{\mathbf{B}}}) = X_{\mathbf{d}'_{\mathbf{B}}+\mathbf{d}}$ for every concept/role $X_{\mathbf{d}'_{\mathbf{B}}} \in \Sigma$; (c) $\Sigma_c = \{X_{\mathbf{d}'_{\mathbf{B}}} \in \Sigma \mid \mathbf{d}'_{\mathbf{B}} \preceq \mathbf{d}_{\mathbf{B}}\}$; (d) $\Sigma_e = \Sigma \setminus \Sigma_c$.*

$f^*(A) = \begin{cases} f(A) & \text{if } A \in \Sigma_c \\ f(\top) \sqcap f(A) & \text{if } A \in \Sigma_e \end{cases}$	$f^*(\bot) = \bot$
	$f^*(C \sqcap D) = f^*(C) \sqcap f^*(D)$
$f^*(R) = \begin{cases} f(R) & \text{if } R \in \Sigma_c \\ f(I) \circ f(R) \circ f(I) & \text{if } R \in \Sigma_e \end{cases}$	$f^*(C \sqcup D) = f^*(C) \sqcup f^*(D)$
	$f^*(\{a\}) = \{f(a)\}$
$f^*(\neg C) = f(\top) \sqcap \neg f^*(C)$	$f^*(\leqslant nR.C) = f(\top) \sqcap \leqslant nf(R).f^*(C)$
$f^*(\exists R.C) = \begin{cases} \exists f(R).f^*(C) & \text{if } R \in \Sigma_c \\ f(\top) \sqcap \exists f(R).f^*(C) & \text{if } R \in \Sigma_e \end{cases}$	$f^*(R^-) = (f(R))^-$
	$f^*(R \circ S) = f^*(R) \circ f^*(S)$
$f^*(\forall R.C) = \begin{cases} f(\top) \sqcap \forall f(R).f^*(C) & \text{if } R \in \Sigma_c \\ f(\top) \sqcap \forall f(R).(\neg f(\top) \sqcup f^*(C)) & \text{if } R \in \Sigma_e \end{cases}$	$f^*(C(a)) = f^*(C)(f(a))$
	$f^*(R(a, b)) = f(R)(f(a), f(b))$
$f^*(\geqslant nR.C) = \begin{cases} \geqslant nf(R).f^*(C) & \text{if } R \in \Sigma_c \\ f(\top) \sqcap \geqslant nf(R).f^*(C) & \text{if } R \in \Sigma_e \end{cases}$	$f^*(C \sqsubseteq D) = f^*(C) \sqsubseteq f^*(D)$
	$f^*(R \sqsubseteq S) = f^*(R) \sqsubseteq f(S)$
$f^*(\exists R.\text{Self}) = \begin{cases} \exists f(R).\text{Self} & \text{if } R \in \Sigma_c \\ f(\top) \sqcap \exists f(R).\text{Self} & \text{if } R \in \Sigma_e \end{cases}$	$f^*(a = b) = f(a) = f(b)$
	$f^*(a \neq b) = f(a) \neq f(b)$

Table 1. DL-embedding on complex expressions and axioms; note: I is the identity role, which can be easily added to $\mathcal{SROIQ}$ using the axioms $\top \sqsubseteq \exists I.\text{Self}$ and $\top \sqsubseteq \leqslant 1 I.\top$

For instance if the concept Team occurs in $\mathcal{C}_{\mathbf{d}}$ with $\mathbf{d} = \langle \mathsf{FWC}, 2010, \mathsf{Africa} \rangle$, it belongs to Σ_c as $\mathbf{d}'_{\mathbf{B}} \preceq \mathbf{d}_{\mathbf{B}}$ for $\mathbf{B} = \emptyset$. Hence $\mathsf{Team}@\mathbf{d} = \mathsf{Team}_{\mathsf{FWC},2010,\mathsf{Africa}}$. This is natural, as in a context wider than $\mathcal{C}_{\mathbf{d}}$ the concept $\mathsf{Team}_{\mathsf{FWC},2010,\mathsf{Africa}}$ is fully defined by Team in $\mathcal{C}_{\langle \mathsf{FWC},2010,\mathsf{Africa} \rangle}$. But $\mathsf{NationalTeam}_{\mathsf{FB}} \notin \Sigma_c$ as $\mathsf{FB} \npreceq \mathsf{FWC}$. Hence $\mathsf{NationalTeam}_{\mathsf{FB}}@\langle \mathsf{FWC}, 2010, \mathsf{Africa} \rangle = \mathsf{NationalTeam}_{\mathsf{FB},2010,\mathsf{Africa}} \sqcap \top_{\mathsf{FWC},2010,\mathsf{Africa}}$. Intuitively, to embed $\mathsf{NationalTeam}_{\mathsf{FB}}$ from $\mathcal{C}_{\langle \mathsf{FWC},2010,\mathsf{Africa} \rangle}$ into a broader context one must restrict it to $\top_{\mathsf{FWC},2010,\mathsf{Africa}}$ because its interpretation in the broader context may be broader.

A ND system for a CKR $\mathfrak{K} = \langle \mathfrak{C}, \mathfrak{M} \rangle$ over $\langle \Gamma, \Sigma \rangle$ is shown in Table 2. Here α_i are either object-formulae of the form $\mathbf{d} : \phi$ ($\mathbf{d} \in \mathfrak{D}_\Gamma$, ϕ is a DL formula over Σ) or meta-formulae μ over Γ, while α and β_i are always object-formulae. A formula $\mathbf{d} : \phi$ is derivable from $\mathfrak{K}$ and Φ (denoted $\mathfrak{K}, \Phi \vdash \mathbf{d} : \phi$) if it is derivable from $\Phi \cup \{\mathbf{d} : \phi \mid \phi \in \mathcal{C}_{\mathbf{d}}, \mathbf{d} \in \mathfrak{D}_\Gamma\} \cup \{\mu \mid \mathfrak{M} \models \mu\}$ using the ND rules of Table 2. A shorthand $\mathfrak{K} \vdash \mathbf{d} : \phi$ is used for $\mathfrak{K}, \emptyset \vdash \mathbf{d} : \phi$.

Theorem 1 (Soundness and Completeness). *$\mathfrak{K} \vdash \mathbf{d} : \phi$ if and only if $\mathfrak{K} \models \mathbf{d} : \phi$.*

Let us show some example deductions in the CKR $\mathfrak{K}$ with structure depicted in Fig 2. Example 1 shows how knowledge is propagated from $\mathcal{C}_{\mathbf{wc}}$ to $\mathcal{C}_{\mathbf{i}}$ via the common super-context $\mathcal{C}_{\mathbf{f}}$, and Example 2 shows how knowledge is propagated from $\mathcal{C}_{\mathbf{wn}}$ to $\mathcal{C}_{\mathbf{f}}$ via the common sub-context $\mathcal{C}_{\mathbf{wc}}$. Finally Example 3 shows how contradicting knowledge can coexist in different separated context.

Example 1. The following deduction shows how the subsumption $\mathbf{wc} : \mathsf{WChamp} \sqsubseteq \mathsf{Player}$ propagates from the FWC context $\mathcal{C}_{\mathbf{wc}}$ to the Italian NFL context $\mathcal{C}_{\mathbf{i}}$. Notice that the result of this deduction, i.e., $\mathbf{i} : \mathsf{WChamp}_{\mathbf{wc}} \sqsubseteq \mathsf{Player}_{\mathbf{wc}}$, in the context $\mathcal{C}_{\mathbf{i}}$ is weaker than the premise as it holds only on the set of players of the Italian National League. In other words, the knowledge shifting from $\mathcal{C}_{\mathbf{wc}}$ to $\mathcal{C}_{\mathbf{i}}$ is limited by the domain of interpretation of $\mathcal{C}_{\mathbf{i}}$.

(1)	$\mathbf{wc} : \mathsf{WChamp} \sqsubseteq \mathsf{Player}$	premise
(2)	$\mathbf{f} : (\mathsf{WChamp} \sqsubseteq \mathsf{Player})@\mathbf{wc}$	Pop, $\mathbf{wc} \preceq \mathbf{f}$
(3)	$\mathbf{f} : \mathsf{WChamp}_{\mathbf{wc}} \sqsubseteq \mathsf{Player}_{\mathbf{wc}}$	by @
(4)	$\mathbf{f} : \mathsf{WChamp}_{\mathbf{wc}} \sqcap \top_{\mathbf{i}} \sqsubseteq \mathsf{Player}_{\mathbf{wc}} \sqcap \top_{\mathbf{i}}$	LReas
(5)	$\mathbf{f} : (\mathsf{WChamp}_{\mathbf{wc}} \sqsubseteq \mathsf{Player}_{\mathbf{wc}})@\mathbf{i}$	by @
(6)	$\mathbf{i} : \mathsf{WChamp}_{\mathbf{wc}} \sqsubseteq \mathsf{Player}_{\mathbf{wc}}$	Push, $\mathbf{i} \preceq \mathbf{f}$

$$\frac{\mathbf{d}:\phi_1 \ \dots\ \mathbf{d}:\phi_n \quad \{\phi_1\dots\phi_n\}\models\phi}{\mathbf{d}:\phi}\,\mathsf{LReas} \qquad \frac{\mathbf{d}:\bot(a)}{\mathbf{e}:\top\sqsubseteq\bot}\,\mathsf{Bot}$$

$$\frac{\mathbf{d}\preceq\mathbf{e}}{\mathbf{f}:A_{\mathbf{d}}\sqsubseteq\top_{\mathbf{e}}} \quad \frac{-}{\mathbf{f}:\exists R_{\mathbf{d}}\top\sqsubseteq\top_{\mathbf{d}}} \quad \frac{-}{\mathbf{f}:\top\sqsubseteq\forall R_{\mathbf{d}}\top_{\mathbf{d}}}\,\mathsf{Top} \qquad \frac{\begin{array}{c}[\mathbf{d}:\top(a)]\\ \mathbf{d}:\top\sqsubseteq\bot\end{array}}{\mathbf{d}:\top\sqsubseteq\bot}\,a\mathsf{E}$$

$$\frac{\mathbf{e}:\phi@\mathbf{d} \quad \mathbf{e}:\top_{\mathbf{d}}(a_1)\ \cdots\ \mathbf{e}:\top_{\mathbf{d}}(a_n) \quad \mathbf{d}\preceq\mathbf{e}}{\mathbf{d}:\phi}\,\mathsf{Push} \qquad \frac{\mathbf{d}:\phi \quad \mathbf{d}\preceq\mathbf{e}}{\mathbf{e}:\phi@\mathbf{d}}\,\mathsf{Pop}$$

$$\frac{\mathbf{d}:A\sqcup B(x) \quad \begin{array}{c}[\mathbf{d}:A(x)]\\ \mathbf{e}:\phi\end{array} \quad \begin{array}{c}[\mathbf{d}:B(x)]\\ \mathbf{e}:\phi\end{array}}{\mathbf{e}:\phi}\,\sqcup\mathsf{E} \qquad \frac{\mathbf{d}:\exists R.A(x) \quad \begin{array}{c}[\mathbf{d}:R(x,y),\ \mathbf{d}:A(y)]\\ \mathbf{e}:\phi\end{array}}{\mathbf{e}:\phi}\,\exists\mathsf{E}$$

$$\frac{\mathbf{d}:\geqslant nR.A(x) \quad \begin{array}{c}[\mathbf{d}:y_i\neq y_j,\ \mathbf{d}:R(x,y_i),\ \mathbf{d}:A(y_i)]_{1\leq i\neq j\leq n}\\ \mathbf{e}:\phi\end{array}}{\mathbf{e}:\phi}\,(\geqslant n)\mathsf{E}$$

Restrictions: **1)** LReas can be applied if every individual occurring in ϕ occurs in a ϕ_i for some $1 \leq i \leq n$; **2)** in the Push rule $a_1, \dots, a_n$ are assumed to be all individuals occuring in ϕ; **3)** the individuals a, y, and y_i, $1 \leq i \leq n$, occuring in aE, $\exists$E, and $(\geqslant n)$E are new, not occuring elsewhere in $\mathfrak{K}$ and the proof apart from the assumptions discharded by these rules.

Table 2. CKR inference rules

Example 2. The following deduction shows how $\mathbf{wn}$: $\mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro}$ (i.e., every football player mentioned in the world news is a professional) propagates from $\mathcal{C}_{\mathbf{wn}}$ to $\mathcal{C}_{\mathbf{f}}$, trough the common sub-context $\mathcal{C}_{\mathbf{wc}}$.

(1)	$\mathbf{wn} : \mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro}$	premise
(2)	$\mathbf{wn} : (\mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro})@\mathbf{wn}$	Pop, $\mathbf{wn} \preceq \mathbf{wn}$
(3)	$\mathbf{wn} : \mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}}$	by @
(4)	$\mathbf{wn} : \mathsf{Player}_{\mathbf{f}} \sqcap \top_{\mathbf{wc}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}} \sqcap \top_{\mathbf{wc}}$	by LReas
(5)	$\mathbf{wc} : \mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}}$	Push, $\mathbf{wc} \preceq \mathbf{wn}$
(6)	$\mathbf{f} : \mathsf{Player}_{\mathbf{f}} \sqcap \top_{\mathbf{wc}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}} \sqcap \top_{\mathbf{wc}}$	Pop, $\mathbf{wc} \preceq \mathbf{f}$
(7)	$\mathbf{f} : \mathsf{Player}_{\mathbf{f}} \sqcap \top_{\mathbf{wc}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}}$	LReas

Notice that we did not infer that $\mathbf{f}$: $\mathsf{Player}_{\mathbf{f}} \sqsubseteq \mathsf{Pro}_{\mathbf{wn}}$, i.e., that every Player of football is a professional player in the world news, but the fact that this subsumption holds only on the players of the FWC domain.

Example 3. Suppose that the Italian News context $\mathcal{C}_{\mathbf{in}}$ contains the facts that Rooney does not take part to the Italian league in 2010, i.e., $\neg\top_{\mathbf{i}}(\mathsf{Rooney})$, and that he is not considered a good football player, i.e., $\neg\mathsf{GoodPlayer}_{\mathbf{f}}(\mathsf{Rooney})$. Suppose also that the world news context $\mathcal{C}_{\mathbf{wn}}$ contains the opposite evaluation, i.e. $\mathsf{GoodPlayer}_{\mathbf{f}}(\mathsf{Rooney})$. In the CKR of Fig. 2, these two contradicting statements do not necessarily lead to inconsistency. Indeed, to derive inconsistency one has to find a context where to combine the two contradicting facts. However, to transfer the facts $\mathbf{wn}$: $\mathsf{GoodPlayer}_{\mathbf{f}}(\mathsf{Rooney})$ and $\mathbf{in}$: $\neg\mathsf{GoodPlayer}_{\mathbf{f}}(\mathsf{Rooney})$ into a common context, one have to pass through $\mathcal{C}_{\mathbf{i}}$. But the fact that Rooney is not an individual of $\mathcal{C}_{\mathbf{i}}$ disables any inference about Rooney in $\mathcal{C}_{\mathbf{i}}$. Model-theoretically we admit CKR models where $\mathsf{Rooney}^{\mathcal{I}_{\mathbf{wn}}} \neq \mathsf{Rooney}^{\mathcal{I}_{\mathbf{in}}}$.

4 Decidability and Complexity

Decidability of CKR entailment is proved indirectly by embedding a CKR into a single DL knowledge base, we will again use DL-embeddings. Given a meta-vocabulary Γ and an object-vocabulary $\Sigma = N_{\mathrm{C}} \uplus N_{\mathrm{R}} \uplus N_{\mathrm{I}}$, a DL-vocabulary $\#(\Gamma,\Sigma) = \#N_{\mathrm{C}} \uplus \#N_{\mathrm{R}} \uplus \#N_{\mathrm{I}}$ is defined as follows: $\#N_{\mathrm{C}} = \{A^{\mathbf{e}}_{\mathbf{d}} \mid A \in N_{\mathrm{C}} \wedge \mathbf{d}, \mathbf{e} \in \mathfrak{D}_\Gamma\}$; $\#N_{\mathrm{R}} = \{R^{\mathbf{e}}_{\mathbf{d}} \mid R \in N_{\mathrm{R}} \wedge \mathbf{d}, \mathbf{e} \in \mathfrak{D}_\Gamma\}$; $\#N_{\mathrm{I}} = \{a^{\mathbf{e}} \mid a \in N_{\mathrm{I}} \wedge \mathbf{e} \in \mathfrak{D}_\Gamma\}$. An embedding of $\mathcal{C}_{\mathbf{d}}$ into $\#(\Gamma,\Sigma)$ is now done by the $\#\mathbf{d}$ operator:

Definition 5 **($\#\mathbf{d}$ operator).** *Given $\mathfrak{K} = \langle \mathfrak{C}, \mathfrak{M} \rangle$ over $\langle \Gamma,\Sigma \rangle$ and $\mathbf{d} \in \mathfrak{D}_\Gamma$, $(\cdot)\#\mathbf{d}$ is defined as $g^*_{\mathbf{d}}(\cdot)$, where $g_{\mathbf{d}} : \Sigma \to \#(\Gamma,\Sigma)$ is a DL-embedding defined as follows: (a) $g_{\mathbf{d}}(a) = a^{\mathbf{d}}$ for every individual a; (b) $g_{\mathbf{d}}(X_{\mathbf{d}'_{\mathbf{B}}}) = X^{\mathbf{d}}_{\mathbf{d}'_{\mathbf{B}}+\mathbf{d}}$ for every concept/role $X_{\mathbf{d}'_{\mathbf{B}}}$; (c) $\Sigma_c = \Sigma$; (d) $\Sigma_e = \emptyset$.*

Using the $\#\mathbf{d}$ operator we now transform a CKR $\mathfrak{K}$ over $\langle \Gamma,\Sigma \rangle$ into a DL theory $\#(\mathfrak{K})$ over $\#(\Gamma,\Sigma)$. For every individual a, concept C, role R, concept/role X, and for every $\mathbf{d}, \mathbf{e}, \mathbf{f} \in \mathfrak{D}_\Gamma$, $\#(\mathfrak{K})$ contains the following axioms (the gap in the numbering is to maintain the correspondence with Definition 3):

1. $\top^{\mathbf{f}}_{\mathbf{d}} \sqsubseteq \top^{\mathbf{f}}_{\mathbf{e}}$ for $\mathbf{d} \prec \mathbf{e}$;
2. $C^{\mathbf{d}}_{\mathbf{e}} \sqsubseteq \top^{\mathbf{d}}_{\mathbf{e}}$;
3. $\exists R^{\mathbf{d}}_{\mathbf{e}}.\top \sqsubseteq \top^{\mathbf{d}}_{\mathbf{e}}$ and $\top \sqsubseteq \forall R^{\mathbf{d}}_{\mathbf{e}}.\top^{\mathbf{d}}_{\mathbf{e}}$;
4. $a^{\mathbf{d}} = a^{\mathbf{e}}$, if $\mathbf{d} \prec \mathbf{e}$;
6. $X^{\mathbf{d}}_{\mathbf{d}} \equiv X^{\mathbf{e}}_{\mathbf{d}}$, if $\mathbf{d} \prec \mathbf{e}$;
7. $C^{\mathbf{d}}_{\mathbf{f}} \equiv C^{\mathbf{e}}_{\mathbf{f}} \sqcap \top^{\mathbf{d}}_{\mathbf{d}}$, if $\mathbf{d} \prec \mathbf{e}$;
8. $I^{\mathbf{d}}_{\mathbf{d}} \circ R^{\mathbf{e}}_{\mathbf{f}} \circ I^{\mathbf{d}}_{\mathbf{d}} \sqsubseteq R^{\mathbf{d}}_{\mathbf{f}}$ and $R^{\mathbf{d}}_{\mathbf{f}} \sqsubseteq R^{\mathbf{e}}_{\mathbf{f}}$, if $\mathbf{d} \prec \mathbf{e}$;
9. $\phi\#\mathbf{d}$ for all $\phi \in \mathrm{K}(\mathcal{C})$ and $\mathbf{d} = \dim(\mathcal{C})$.

Lemma 2. *Given a CKR $\mathfrak{K}$, (a) if $\mathfrak{K}$ is $\mathbf{d}$-satisfiable then $\#(\mathfrak{K})$ is satisfiable; (b) if there is a $\mathbf{d}$ such that $\#(\mathfrak{K}) \not\models \top^{\mathbf{d}}_{\mathbf{d}} \sqsubseteq \bot$, then $\mathfrak{K}$ is $\mathbf{d}$-satisfiable.*

Reasoning in CKR is now reduced into reasoning in $\mathcal{SROIQ}$. Subsumption is decidable for $\mathcal{SROIQ}$ KB that are $\precsim$-stratified [12]. Hence we can prove decidability only for CKRs that are transformed into $\precsim$-stratified KBs. We say that a CKR is $\precsim$-stratified if the set of RIA $\bigcup_{\mathbf{d}\in\mathfrak{D}_\Gamma} \{(R \sqsubseteq S)\#\mathbf{d} \mid R \sqsubseteq S \in \mathrm{K}(\mathcal{C}_{\mathbf{d}})\}$ is $\precsim$-stratified. The RIA introduced in step 8 are not $\precsim$-stratified, but it suffices to add $I^{\mathbf{d}}_{\mathbf{d}} \circ R^{\mathbf{e}}_{\mathbf{f}} \sqsubseteq S_1$, $S_1 \circ I^{\mathbf{d}}_{\mathbf{d}} \sqsubseteq R^{\mathbf{d}}_{\mathbf{f}}$, where S_1 is a new role w.r.t. each pair $R^{\mathbf{d}}_{\mathbf{f}}$ and $R^{\mathbf{e}}_{\mathbf{f}}$. Hence if a $\mathfrak{K}$ is $\precsim$-stratified, there is a $\precsim$-stratified $\mathcal{SROIQ}$ KB equivalent to $\#(\mathfrak{K})$, and hence subsumption is decidable.

Theorem 2. *If $\mathfrak{K}$ is $\precsim$-stratified, then checking if $\mathfrak{K} \models \mathbf{d} : C \sqsubseteq D$ is decidable with the complexity upper bound of* 2NEXPTIME.

The complexity upper bound is established by the fact that the number of dimensions (a fixed constant) and also the number of contexts are bounded. The number of contexts n is always smaller than the size m of the knowledge base $\mathfrak{K}$ because in order to initialize a context we must add several axioms into $\mathfrak{M}$. Consecutive analysis of the construction of $\#(\mathfrak{K})$ shows that its size is bounded by $k \times m \times n^2$ for some constant k, and therefore under $O(m^3)$. So the size of $\#(\mathfrak{K})$ and the time required to generate it is polynomial in the size of $\mathfrak{K}$.

5 Related Work

Both aRDF [25] and Context Description Framework [13] extend RDF triples by an n-tuple of qualification attributes with partially ordered domains. Apart from CKR being based on $\mathcal{SROIQ}$ it differs from these approaches by qualifying whole theories and not each formula separately. This approach is more compact as usually the context is shared by a group of formulae. An extension of RDFS to cope with context was proposed by [8] and further developed in [1]. A new predicate $\mathsf{isin}(c,\phi)$ is used to assert that the triple ϕ occurs in the context c. A set of operators to combine contexts ($c_1 \wedge c_1$, $c_1 \vee c_2$, $\neg c$) and to relate contexts ($c \Rightarrow c_2$, $c \rightarrow c_2$) is defined, making the approach particularly suited for manipulating contexts. Unfortunately, no sound and complete axiomatization or decision procedure was provided so far.

The contextual DL $\mathcal{ALC}_{\mathcal{ALC}}$ [14] is a multi-modal extension of the $\mathcal{ALC}$ DL with the contextual modal operator $[C]_r A$ representing "all objects of type A in all contexts of type C reachable from the current context via relation r." In both $\mathcal{ALC}_{\mathcal{ALC}}$ and CKR contextual structure is formalized in a meta-language separated from the domain language used to describe the domain. The main difference is that CKR is more expressive in the object-language ($\mathcal{SROIQ}$ vs. $\mathcal{ALC}$) but less expressive in the contextual assertions, allowing qualification of knowledge only w.r.t. individual contexts rather than context classes as in $\mathcal{ALC}_{\mathcal{ALC}}$.

The Metaview approach [24] enriches OWL ontologies with logically treated annotations and it can be used to model contextual metadata similarly to CKR albeit on per-axiom basis. The main difference is that in the Metaview approach the contextual level has no implications on ontology reasoning. Instead, a contextually sensitive query language MQL is provided.

CKR is also logically related to approaches such as multi-context systems [7], distributed description logics [4], and especially to package-based description logics [2] and semantic imports [19]. While similar techniques are employed in CKR in order to facilitate information reuse in between contexts, they are used to meet different goals. The amount of information that is possibly "imported" from one context to another by qualified symbols depends on the relation of these context in the CKR's coverage hierarchy, thus reflecting the underlaying ideas of the AI theories of context.

6 Conclusion

CKR is a novel framework for representing contextual knowledge in the SW. We have provided a sound and complete axiomatization and we have shown that reasoning in CKR is decidable at no additional complexity costs. After the recent introduction of a tractable version of CKR built on top of RDFS [11] we plan to investigate on other tractable local languages, e.g., OWL-Horst [23]. For the tractable version we have developed a prototype [9, 11] on top of the Sesame 2 RDF triple store, where contexts have been naturally implemented with named graphs [5]. We also want to study a distributed tableaux based reasoning technique for CKR.

Acknowledgements. The authors would like to thank to Andrei Tamilin, Mathew Joseph, and Loris Bozzato. This work was supported from the LiveMemories project (`http://www.livememories.org/`). Martin Homola also supported from the Slovak national project VEGA 1/0688/10.

References

1. Bao, J., Tao, J., McGuinness, D.: Context representation for the Semantic Web. In: WebSci10 (2010)
2. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-based description logics. In: Modular Ontologies, LNCS, vol. 5445, pp. 349–371. Springer (2009)
3. Benerecetti, M., Bouquet, P., Ghidini, C.: Contextual Reasoning Distilled. JETAI 12(3), 279–305 (2000)
4. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. JoDS 1, 153–184 (2003)
5. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW'05. ACM (2005)
6. Ding, L., Finin, T., Peng, Y., da Silva, P.P., McGuinness, D.: Tracking RDF graph provenance using RDF molecules. In: ISWC (2005)
7. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artif. Intell. 65(1), 29–70 (1994)
8. Guha, R., McCool, R., Fikes, R.: Contexts for the Semantic Web. In: ISWC (2004)
9. Homola, M., Serafini, L., Tamilin, A.: Modeling contextualized knowledge. In: CIAO (2010)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: KR (2006)
11. Joseph, M., Serafini, L.: Simple reasoning for contextualized rdf knowledge. In: WoMo (2011)
12. Kazakov, Y.: An extension of complex role inclusion axioms in the description logic $\mathcal{SROIQ}$. In: IJCAR (2010)
13. Khriyenko, O., Terziyan, V.: A framework for context sensitive metadata description. IJSMO 1(2), 154–164 (2006)
14. Klarman, S., Gutiérrez-Basulto, V.: $\mathcal{ALC}_{\mathcal{ALC}}$: a context description logic. In: JELIA (2010)
15. Lenat, D.: Cyc: A large-scale investment in knowledge infrastructure. Commun. ACM 38(11), 33–38 (1995)
16. Lenat, D.: The dimensions of context space. Tech. rep., CYCorp (1998), http://www.cyc.com/doc/context-space.pdf
17. Liao, H.C., Tu, C.C.: A RDF and OWL-based temporal context reasoning model for smart home. Inform. Tech. J. 6, 1130–1138 (2007)
18. McCarthy, J.: Notes on formalizing context. In: IJCAI (1993)
19. Pan, J.Z., Serafini, L., Zhao, Y.: Semantic import: An approach for partial ontology reuse. In: WoMo-06 (2006)
20. Prawitz, D.: Natural Deduction: A Proof-Theoretical Study. Almquist and Wiksell (1965)
21. Serafini, L., Homola, M.: Contextualized Knowledge Repositories for the Semantic Web. Tech. Rep. 31940, FBK (2011), https://dkm.fbk.eu/images/6/64/TR31940.pdf
22. Stoermer, H.: Introducing context into Semantic Web knowledge bases. In: CAiSE DC (2006)
23. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. Web Sem. 3(2–3), 79–115 (2005)
24. Tran, T., Haase, P., Motik, B., Cuenca Grau, B., Horrocks, I.: Metalevel information in ontology-based applications. In: AAAI (2008)
25. Udrea, O., Recupero, D., Subrahmanian, V.S.: Annotated RDF. ACM Trans. Comput. Log. 11(2), 1–41 (2010)
26. W3C: OWL 2 Web Ontology Language Document Overview. W3C Recommendation (2009)

In the Search of Improvements to the $\mathcal{EL}^+$ Classification Algorithm

Barış Sertkaya

sertkaya.baris@googlemail.com

Abstract. We investigate possible improvements to the existing algorithm for classifying $\mathcal{EL}^+$ TBoxes. We present a modified algorithm based on the well-known linear closure algorithm from relational databases. Despite its better worst-case complexity, surprisingly it turns out that this algorithm does not perform well in practice. We discuss optimizations to the existing algorithm and evaluate them using our prototypical reasoner CHEETAH on several large bio-medical knowledge bases.

1 Introduction

In [9, 8] Brandt has shown that the tractability result in [2] for subsumption w.r.t. cyclic $\mathcal{EL}$ TBoxes can be extended to the DL $\mathcal{ELH}$, which in addition to $\mathcal{EL}$ allows for general concept inclusion axioms and role hierarchies. Later in [3] Baader et. al. have shown that the tractability result can even be further extended to the DL $\mathcal{EL}^{++}$ which in addition to $\mathcal{ELH}$ allows for the bottom concept, nominals, role inclusion axioms, and a restricted form of concrete domains. In addition to these promising theoretical results, it turned out that despite their relatively low expressivity, these fragments are still expressive enough for the well-known bio-medical knowledge bases SNOMED [12] and (large parts of) Galen [19], and the Gene Ontology GO [11]. In [4, 6, 20] the practical usability of these fragments on large knowledge bases has been investigated. The CEL Reasoner [18] was as a result of these studies the first reasoner that could classify the mentioned knowledge bases from life sciences domain in reasonable times.

Successful applications of the $\mathcal{EL}$ family increased investment of further work in this direction. The $\mathcal{EL}$ family now provides the basis for the profile OWL2 EL[1]. Moreover, there are now a few other reasoners specifically tailored for the $\mathcal{EL}$ family, like Snorocket [16] and TrOWL [21], and CB [15], which extends the $\mathcal{EL}^{++}$ algorithm to Horn $\mathcal{SHIQ}$. A comprehensive study comparing the performace of several reasoners on large bio-medical knowledge bases has been presented in [13].

In the present work we investigate possible improvements to the existing classification algorithm for the $\mathcal{EL}$ family. We present a modified algorithm based on the well-known linear closure algorithm [7] from relational databases [17]. We evaluate both the modified algorithm and the implementation of the simple

[1] http://www.w3.org/TR/owl2-profiles/#OWL_2_EL

algorithm in our prototypical $\mathcal{EL}^+$ reasoner CHEETAH on several large knowledge bases from life sciences. Surprisingly, it turns out that despite its better worst-case complexity, the modified algorithm performs worse than the simple algorithm in practice. In Section 2 we introduce the linear closure algorithm from relational databases. In Section 3 we present our modified algorithm based on linear closure, and in Section 4 we present our experimental results.

2 Computing Closure Under Functional Dependencies

In relational databases [17], specification of constraints on data is of crucial importance for correct modelling of the world and correct design of the database schemas. One way of specifying constraints is using functional dependencies introduced in [10]. A functional dependency occurs when the values of a tuple on one set of attributes uniquely determine the values on another set of attributes. Formally, given a relation r and a set of attribute names R, a *functional dependency* (FD) is a pair of sets $X, Y \subseteq R$ written as $X \to Y$. The relation r *satisfies* the FD $X \to Y$ if the tuples with equal X-values also have equal Y-values. In this case, one says that the set of attributes X *functionally determine* the set of attributes Y.

Given a set of FDs $\mathcal{F}$ and an FD $X \to Y$, one interesting question is whether $\mathcal{F}$ *implies* $X \to Y$, i.e., whether every relation that satisfies all FDs in $\mathcal{F}$ also satisfy $X \to Y$, which we denote as $\mathcal{F} \models X \to Y$. In order to answer this, one can compute the smallest set of all FDs that $\mathcal{F}$ implies by using a set of inference axioms called *Armstrong's axioms* [1] and check whether the mentioned FD is an element of this set. However, the set of FDs that $\mathcal{F}$ implies can be considerably larger than $\mathcal{F}$ and costly to compute. Thus one is interested in answering this question without computing this set. Instead, one computes the so-called closure of X under $\mathcal{F}$ and checks whether it contains Y. The *closure* of a set of attributes $X \subseteq R$ under a set of FDs $\mathcal{F}$ is the smallest subset X^+ of R such that $X \subseteq X^+$ and for every $A \to B \in \mathcal{F}$ if $A \subseteq X^+$ holds, then $B \subseteq X^+$ holds as well. [2]

2.1 The Linear Closure Algorithm

In [7] Beeri and Bernstein have given an algorithm for efficiently computing closure under a set of FDs. Briefly, for each attribute the algorithm keeps an index pointing to the set of FDs whose left handsides contain that attribute. Additionally, for each FD it keeps a counter whose value is initally the size of the left handside of that FD. Initialization of these data structures is shown in the procedure `Initialization` in Algorithm 1.

For computing the closure of a set of attributes `x` under a set of FDs `F` it keeps a queue `update` which is initally equal to `x`. In the procedure `Closure` it repeatedly fetches and removes an attribute from `update` and decrements the

[2] Note that, from the viewpoint of logic, computing closure is computing consequences in propositional Horn logic. In fact, the notions we have defined can easily be reformulated in propositional logic when we view the attributes as propositional variables.

Algorithm 1 The Linear Closure Algorithm

```
Procedure: Initialization
 1: for all FD $W \to Z \in F$ do
 2:     $count[W \to Z] := |W|$
 3:     for all attribute $A \in W$ do
 4:         $list[A] := list[A] \cup \{W \to Z\}$
 5:     end for
 6: end for
 7: $newdep := update := x$
Procedure: Closure
 1: while $update \neq \emptyset$ do
 2:     choose an $A$ from $update$
 3:     $update := update \setminus \{A\}$
 4:     for all FD $W \to Z \in list[A]$ do
 5:         $count[W \to Z] := count[W \to Z] - 1$
 6:         if $count[W \to Z] = 0$ then
 7:             $add := Z \setminus newdep$
 8:             $newdep := newdep \cup add$
 9:             $update := update \cup add$
10:         end if
11:     end for
12: end while
    return $newdep$
```

counters of FDs that contain this attribute in the left handside. Once a counter becomes zero, it extends the queue `update` and the closure `newdep` with the new attributes on the right handside of that FD. This continues until the queue `update` becomes empty.

Note that the initialization takes at most $|F|.|W|$ time, which is linear in the size of the input. Now consider the closure computation: Each attribute can enter `update` at most once. For each attribute `A` fetched from `update` the counters of the FDs in `list[A]` are decremented, which is performed at most $\Sigma_{W \to Z \in F}|W|$ times. If the counter of any FD $W \to Z$ becomes 0, then the new attributes in `Z` are added to `update` and `newdep`. If the involved sets are represented as bit vectors, this operation takes time proportional to $\Sigma_{W \to Z \in F}|Z|$.

Since all steps of the algorithm can be performed in time linear in the sizes of FDs `F` and the set of attributes, the algorithm has complexity $O(n)$.

3 A Modified Algorithm for classifying $\mathcal{EL}^+$ TBoxes

In the present section we present an $\mathcal{EL}^+$ classification algorithm based on the linear closure algorithm introduced in the previous section. $\mathcal{EL}^+$ is the DL allowing for the top concept $\top$, conjunction $C \sqcap D$, existential restriction $\exists r.A$, general concept inclusion axioms (GCIs) $C \sqsubseteq D$ and role inclusion axioms (RIs) $r_1 \circ \cdots \circ r_n \sqsubseteq s$, where A is an atomic concept, r an atomic role, and C, D

concept descriptions. RIs are interpreted as $r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$, where $\circ$ denotes composition of binary relations.

In [9, 8] Brandt has shown that the tractability result in [2] for subsumption w.r.t. cyclic TBoxes can be extended to the DL $\mathcal{ELH}$, which in addition to $\mathcal{EL}$ allows for GCIs and simple RIs, i.e., RIs with an atomic role on the left handside. Later in [3] Baader et. al. have shown that the tractability result can even be further extended to the DL $\mathcal{EL}^{++}$ which in addition to $\mathcal{EL}^{+}$ allows for the bottom concept $\bot$, nominals $\{a\}$ and a restricted form of concrete domains.

In [4, 5] Baader et. al. have considered a restriction of the polynomial-time classification algorithm in [3] to $\mathcal{EL}^{+}$ and have given a refined version of the algorithm tailored for efficient implementations of it. This algorithm initially turns the input TBox into a normalized TBox by applying a series of normalization rules. Afterwards, it applies a set of completion rules to compute a mapping S assigning to each concept name a subset of the concept names occurring in the original TBox. The completion rules are repeatedly applied until no rule applies any more. Consequently, the mapping S maps every concept name A to its set of subsumers. That is, $B \in S(A)$ implies that the subsumption relation $A \sqsubseteq B$ holds in the original TBox. What is important here is a clever strategy for finding the next completion rule to be applied. Because if this is done by a brute-force approach, even though still polynomial, the algorithm will not perform well in practice for large real life TBoxes. In order to avoid this, the "refined" algorithm suggested in [4, 5] uses a modification of the approach used in [14] for checking satisfiability of propositional Horn formulae. As shown in [6, 20], the refined classification algorithm makes at most $O(n^4)$ additions to the mapping S and to the other data structures used.

In the following, we present an algorithm based on the linear closure algorithm introduced in the previous section. We exploit the similarity between computing closure under a set of functional dependencies and computing the set of subsumers of a concept. In its simplest form, one can view a GCI that consists of conjunctions of concept names on both sides as an FD. In this case, computing the subsumers of a concept w.r.t. a set of such GCIs trivially boils down to computing the closure of that concept under that set of GCIs, and classifying the TBox boils down to computing the *closure of every concept name* occurring in the TBox. For the general case, where TBox contains RIs, and where GCIs contain existential restrictions, the inferences due to these should of course also be taken into account.

As in the existing algorithm, we first transform the TBox into a normal form. Our normal form slightly differs from the original one introduced in [9, 3]. Instead of only binary conjunctions on the left handsides of GCIs, it allows for conjunctions of arbitary size. This kind of GCIs have already been used in the normal form in [4, 6]. There it was reported that for large knowledge bases like SNOMED [12], this minor change considerably reduces the number of newly introduced concept names, and thus reduces the size of the normalized knowledge base. Here, in addition to the left handside, we also allow conjunctions of arbitrary size on the right handside of GCIs. Of course theoretically this does not make any difference but it in the implemtation of the algorithm it allows a compact representation of the axioms.

3.1 The Normal Form

Given a TBox $\mathcal{T}$ we write $\mathrm{CN}_{\mathcal{T}}$ and $\mathrm{CN}_{\mathcal{T}}^{\top}$ to denote the sets of concept names occurring in $\mathcal{T}$ with and without the top concept, respectively. Likewise we write $\mathrm{RN}_{\mathcal{T}}$ to denote the set of role names occurring in $\mathcal{T}$. We say that $\mathcal{T}$ is in *normal form* if

1. all GCIs in $\mathcal{T}$ are of the form

$$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq D_1 \sqcap \ldots \sqcap D_m$$

   where C_i is either a concept name from $\mathrm{CN}_{\mathcal{T}}^{\top}$ or is of the form $\exists r.A$, and D_j is either a concept name from $\mathrm{CN}_{\mathcal{T}}$ or is of the form $\exists r.A$ where $A \in \mathrm{CN}_{\mathcal{T}}^{\top}$ and $r \in \mathrm{RN}_{\mathcal{T}}$.
2. all role inclusions are of the form $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$ where $r_1, r_2, s \in \mathrm{RN}_{\mathcal{T}}$.

Basically, a normalized GCI consists of conjunctions on both sides where conjuncts are either concept names or existentially quantified concept names. Role inclusion axioms are normalized exactly the same way as in [4, 6]. Note that

NF1	$r_1 \circ \ldots \circ r_k \sqsubseteq s$	$\rightsquigarrow$	$r_1 \circ \ldots \circ r_{k-1} \sqsubseteq u,\ u \circ r_k \sqsubseteq s$
NF2	$C_1 \sqcap \ldots \sqcap \exists r.\hat{C} \sqcap \ldots \sqcap C_n \sqsubseteq D$	$\rightsquigarrow$	$\hat{C} \sqsubseteq A,\ C_1 \sqcap \ldots \sqcap \exists r.A \sqcap \ldots \sqcap C_n \sqsubseteq D$
NF3	$C \sqsubseteq D_1 \sqcap \ldots \sqcap \exists r.\hat{C} \sqcap \ldots \sqcap D_m$	$\rightsquigarrow$	$C \sqsubseteq D_1 \sqcap \ldots \sqcap \exists r.A \sqcap \ldots \sqcap D_m,\ A \sqsubseteq \hat{C}$

where $\hat{C} \notin \mathrm{CN}_{\mathcal{T}}^{\top}, C, D, C_i, D_i$ are arbitrary concept descriptions, u denotes a new role name, and A denotes a new concept name.

Fig. 1. Normalization rules

our normalization rules shown in Figure 1 are a "stripped down" version of the original normalization rules. Therefore the linear upper bound on the size of the normalized TBox shown in [4, 20] also holds for our normalization rules.

3.2 The Modified Classification Algorithm

Like the linear closure algorithm, our classification algorithm maintains a set of counters in order to decide when to apply a GCI. However we do not maintain only one counter per GCI, but for every concept name we have a counter for every GCI. This is because we want to compute the subsumer list of every concept name occurring in the input TBox, and not only one concept name. The counters initally contain the size (number of conjuncts) of the left handsides of the GCIs. For every concept name we maintain a stack that keeps track of the concepts still to be processed for that concept name. Note that our algorithm differs from the original classification algorithm in [4, 6] here in the sense that instead of keeping track of axioms to be processed we keep track of concepts to be processed. Our possible stack entries are concept names $A \in \mathrm{CN}_{\mathcal{T}}^{\top}$, or existentially restricted concept names $\exists r.A$ where $A \in \mathrm{CN}_{\mathcal{T}}^{\top}$. The counters are kept in the two-dimensional array `count[`A`][`$C \sqsubseteq D$`]` and the stacks are stored as `q(`A`)` for $A \in \mathrm{CN}_{\mathcal{T}}^{\top}$ and $C \sqsubseteq D \in \mathcal{T}$.

In addition to these, for every concept name we keep an index pointing to the list of GCIs that contain this concept name on the left handside. This information is stored in `list(`A`)` for $A \in \mathrm{CN}_{\mathcal{T}}$. As in the original algorithm we keep a subsumer list $S(B)$ for each concept name B. Unlike the original algorithm in addition to concept names this list also contains concept descriptions of the form $\exists r.A$ where $A \in \mathrm{CN}_{\mathcal{T}}^{\top}$. Therefore we do not have the data structure $R(\cdot, \cdot)$ in the original algorithm for storing relations.

Having explained the data structures we are now ready to give the algorithm. The first procedure properly initializes the data structures `count[·][·]`, `list(·)`, `q(·)`, and `S(·)`.

Algorithm 2 initialize the data structures

```
Procedure: Initialization
 1: for all GCI $\alpha = \bigsqcap\{C_1, \ldots, C_n\} \sqsubseteq \bigsqcap\{D_1, \ldots, D_m\} \in \mathcal{T}$ do
 2:     for all $A \in \mathrm{CN}_{\mathcal{T}}$ do
 3:         count[A][$\alpha$] = $n$
 4:     end for
 5:     for all $C \in \{C_1, \ldots, C_n\}$ do
 6:         list(C) = list(C) $\cup\{\alpha\}$
 7:     end for
 8: end for
 9: for all $A \in \mathrm{CN}_{\mathcal{T}}$ do
10:     $q(A) = \{A, \top\}$
11:     $S(A) = \{A, \top\}$
12: end for
```

Next we describe the processing of the stacks. Upon popping an entry (a normalized concept description) C from $q(A)$ we call `process-concept-name` if C is a concept name, and `process-existential-restriction` if C is an existential restriction. Later we traverse the GCIs that have C on the left handside and decrement the counters for A. If the counter $count[A][\bigsqcap\{C_1, \ldots, C_n\} \sqsubseteq \bigsqcap\{D_1, \ldots, D_m\}]$ becomes zero, we extend $S(A)$ and $q(A)$ with the new concept descriptions in $\{D_1, \ldots, D_m\}$.

A concept name fetched from the stack of A is processed as in Algorithm 4, and an existential restriction popped from the stack of A is processed as shown in Algorithm 5. Here $\sqsubseteq_{\mathcal{T}}^{*}$ denotes the reflexive transitive closure of the role hierarchy axioms in the normalized TBox. Processing of the stacks continues until

Algorithm 3 process stack entry C popped from $q(A)$

```
Procedure: process-stack-entry(A,C)
 1: if $C \in \mathrm{CN}_\mathcal{T}$ then
 2:     process-concept-name(A,C)
 3: end if
 4: if $C = \exists r.E$ then
 5:     process-existential-restriction(A,$\exists r.E$)
 6: end if
 7: for all GCI $\alpha = \bigsqcap\{C_1, \dots, C_n\} \sqsubseteq \bigsqcap\{D_1, \dots, D_m\} \in$ list(C) do
 8:     count[A][$\alpha$] = count[A][$\alpha$] - 1
 9:     if count[A][$\alpha$] = 0 then
10:         $q(A) = q(A) \cup \{D_i \mid D_i \notin S(A)\}$
11:         $S(A) = S(A) \cup \{D_i \mid D_i \notin S(A)\}$
12:     end if
13: end for
```

Algorithm 4 process the concept name B popped from $q(A)$

```
Procedure: process-concept-name(A,B)
 1: for all $D \in \mathrm{CN}_\mathcal{T}$ s.t. $\exists r.A \in S(D)$ and $\exists r.B \notin S(D)$ do
 2:     $q(D) = q(D) \cup \{\exists r.B\}$
 3:     $S(D) = S(D) \cup \{\exists r.B\}$
 4: end for
```

all stacks `q(·)` are empty. Note that our algorithm differs from the "refined" algorithm introduced in [5, 6] only in the way how the stacks (there queues) are processed, and how axioms that apply at a particular step are detected. In principle it still performs exactly the same operations in the "abstract" algorithm introduced there. That is, it still performs the *completion rules* in [5, 6]. In fact, the lines 9-11 of Algorithm 3 implement the completion rules R1, R2 and part of R3 in [6]. Lines 2-4 of Algorithm 5 implement rest of R3, and the whole `process-existential-restricton` procedure implement rules R4 and R5. Since we do not modify the original abstract algorithm in [5, 6] we do not need to give proof of correcteness of our algorithm here.

It has been shown in [6, 20] that the refined algorithm there makes at most n^4 additions to the subsumer list $S(\cdot)$ and to the queues used, where n is the size of the normalized TBox. For every addition to $S(\cdot)$ this algorithm performs a subset check in order to decide whether the fetched axiom from the queue is applicable at that step or not. This subset check brings an overhead which in the worst-case is n, thus the overall runtime of the original algorithm is $O(n^5)$.

The counters used by our algorithm allow us to check whether an axiom applies without doing the subset check mentioned above, thus avoid the n-step overhead in the worst-case. Basically, this is how our algorithm achieves a better worst-case complexity $O(n^4)$ instead of the $O(n^5)$ worst-case complexity of the original algorithm.

Algorithm 5 process the existential restriction $\exists r.E$ fetched from $q(A)$

```
Procedure: process-existential-restriction(A,$\exists r.E$)
1: for all $s \in \mathrm{RN}_\mathcal{T}$ s.t. $r \sqsubseteq^*_\mathcal{T} s$ do
2:     for all $D \in \mathrm{CN}_\mathcal{T}$ s.t. $D \in S(E)$ and $\exists s.D \notin S(A)$ do
3:         $q(A) = q(A) \cup \{\exists s.D\}$
4:         $S(A) = S(A) \cup \{\exists s.D\}$
5:     end for
6:     for all $D \in \mathrm{CN}_\mathcal{T}$ s.t. $\exists x.A \in S(D)$ and $\exists y.E \notin S(D)$ and $x, y \in \mathrm{RN}_\mathcal{T}$ s.t.
       $x \circ s \sqsubseteq y \in \mathcal{T}$ do
7:         $q(D) = q(D) \cup \{\exists y.E\}$
8:         $S(D) = S(D) \cup \{\exists y.E\}$
9:     end for
10:    for all $D \in \mathrm{CN}_\mathcal{T}$ s.t. $\exists x.D \in S(E)$ and $\exists y.D \notin S(A)$ and $x, y \in \mathrm{RN}_\mathcal{T}$ s.t.
       $s \circ x \sqsubseteq y \in \mathcal{T}$ do
11:        $q(A) = q(A) \cup \{\exists y.D\}$
12:        $S(A) = S(A) \cup \{\exists y.D\}$
13:    end for
14: end for
```

4 Implementation and Evaluation

In order to evaluate the runtime behaviour of our modified algorithm, we have implemented it and performed a series of tests on large bio-medical knowledge bases: Foundational Model of Anatomy[3] (FMA) is a large but simple TBox that contains 75139 concept names. Similarly, the Gene Ontology [4] (GO) and National Cancer Institute Thesaurus [5] (NCI) are large knowledge bases with shallow hierarchies. The GO contains 25070 concept names, and the NCI contains 27652 concept names. We have stripped down Galen [6] by removing functionalities and inverse roles to obtain the knowledge base Galen$^-$, which contains 23136 concept names. Finally we have also used the very large knowledge base SNOMED [7], which contains 293707 concept names. We have implemented the algorithm in the C programming language due to its speed and efficient use of the memory, which is important for dealing with these large knowledge bases. Currently our implementation can only read $\mathcal{EL}^+$ knowledge bases written in OWL 2 Functional-Style Syntax. We have implemented the parser for this syntax using the tools lex and yacc, which are used for generating lexical analyzer and parser for a given grammar.

In order to evaluate its performance, we have implemented the modified algorithm presented above, in our prototypical reasoner CHEETAH[8]. We have

[3] http://sig.biostr.washington.edu/projects/fm/AboutFM.html
[4] http://www.geneontology.org/
[5] http://www.cancer.gov/cancertopics/cancerlibrary/terminologyresources
[6] http://www.co-ode.org/galen/
[7] http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html
[8] http://code.google.com/p/cheetah

	FMA	GO	NCI	SNOMED	Galen^{-}
CHEETAH	4.78	1.37	1.29	179.57	21.41
CHEETAH*	4.03	1.00	0.88	92.23	12.32
CB	6.59	3.17	2.13	46.88	3.20

Table 1. Comparison of runtimes in seconds

compared its performance with the performance of the simple algorithm that performs subset checks instead of maintaining counters in order to decide when to apply an axiom. The simple algorithm still uses the normalization rules introduced in Section 3.1, and stacks for keeping track of concepts still to be processed, but does not use counters. Instead, for a concept name B popped from `q(A)`, it compares the left handsides of axioms containing B with the current subsumers of A, and applies an axiom if the former is a subset of the later. In our comparison we have also involved the CB Reasoner[9] that has been introduced in [15]. The underlying algorithm of the CB Reasoner extends the $\mathcal{EL}^{++}$ classification algorithm in [4] to the much more expressive fragment Horn $\mathcal{SHIQ}$, for which reasoning is not tractable any more. CB is able to classify the medical knowledge base Galen [19] that uses this expressivity, and as reported in [13], it outperforms all other reasoners for several other large bio-medical knowledge bases as well.

The results of our experiments were surprising. Despite its better worst-case complexity, in practice our modified algorithm performed worse than our implementation of the simple algorithm. The results of our experiments are shown in Table 1, where CHEETAH represents the implementation of our modified algorithm, and CHEETAH* the implementation of the simple algorithm. The experiments are run on a computer with an Intel Core i3 processor running at 2.1 GHz, 8 GB of main memory and on the Linux operating system with 2.6.38 kernel. As seen in Table 1, the modified algorithm performs always worse than the simple algorithm. A closer look into the input knowledge bases reveals that the worst-case, i.e., axioms with very long conjunctions on the left handsides do not occur in practice. In fact, in SNOMED the longest conjunction on the left handside of a GCI is 12, for the Galen version we used it is 5, for FMA, GO and NCI it is just 1. When it comes to average size of conjunctions on the left hand side, for SNOMED it is 1.30 and for Galen^{-} it is 1.29, which are very small compared to the number of concept names occurring in these knowledge bases. This explains the poor performance of the modified algorithm. The worst-case, i.e., large conjunctions on the left handsides, does not occur in any of the knowledge bases we have used in our experiments. In practice the conjunctions on the left handsides are so small that even plain subset check is fast enough compared to the overhead of maintaining the counters.

According to the table the CB Reasoner performs in general better than both CHEETAH and CHEETAH*. This is due to our unoptimized implementation of the processing of stacks. CB spends a big portion of the runtime for loading

[9] http://code.google.com/p/cb-reasoner

and normalizing the knowledge base however it is very efficient in computing the subsumer lists. For instance for SNOMED in our experiments it took CB 13.99 seconds to load and normalize the knowledge base, and only 21.47 seconds to compute the subsumer lists. On the other hand for CHEETAH* loading and normalizing the knowledge base took 3.9 seconds, and computing the subsumer lists took 84.13 seconds.

5 Concluding Remarks and Future Work

We have investigated a modification to the $\mathcal{EL}^+$ classification algorithm for improving its worst-case complexity. It turned out the modified algorithm performs worse in practice. However, there is some room for further improvement of both the modified and the simple algorithm. During the execution of both algorithms, some axioms are applied several times, which in principle could be avoided. For instance if we are processing the stack of concept name A and find out that A is subsumed by B, we can skip the axioms already applied while computing the subsumers of B and just extend the subsumer list of A with that of B. This would bring the overhead of keeping track of which axioms have already been applied for which concept name, but save the effort of applying those axioms again. One other possible improvement is to make use of the axioms that have only one concept name or existential restriction, i.e., no conjuction on the left handside. One can apply such axioms immediately before the execution of the algorithm, thus pre-filling the stacks and subsumer lists with told-subsumers appropriately.

As future work we are going to implement and test these further improvements. In addition to this, we are going to extend the CHEETAH reasoner to support the OWL2 EL Profile and improve its usability by providing a platform independent Java interface and a Protege plugin for it.

Acknowledgements: We would like to thank Yevgeny Kazakov for giving his remarks and providing information about the CB Reasoner.

References

1. W. W. Armstrong. Dependency structures of data base relationships. *Proceedings of the Information Processing Congress 74, (IFIP 74)*, pages 580–583. North-Holland, 1974.
2. F. Baader. Terminological cycles in a description logic with existential restrictions. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 325–330. Morgan Kaufmann, 2003.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, (IJCAI 05)*, pages 364–369. Professional Book Center, 2005.
4. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the Methods for Modalities Workshop (M4M-05)*, 2005.

5. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.
6. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Journal of Logic, Language and Information, Special Issue on Method for Modality (M4M)*, 2007. To appear.
7. C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems*, 4(1):30–59, 1979.
8. S. Brandt. On subsumption and instance problem in $\mathcal{ELH}$ w.r.t. general tboxes. *Proceedings of the 2004 International Workshop on Description Logics, (DL2004)*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
9. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, gci axioms, and - what else? *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI 2004)*, pages 298–302. IOS Press, 2004.
10. E. F. Codd. A relational model of data for large shared data banks. *Communications of ACM*, 13(6):377–387, 1970.
11. T. G. O. Consortium. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
12. R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, International, Northfield, IL: College of American Pathologists, 1993.
13. K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web Journal*, 2011. To appear.
14. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.
15. Y. Kazakov. Consequence-driven reasoning for horn shiq ontologies. *Proceedings of the 21st International Joint Conference on Artificial Intelligence, (IJCAI 2009)*, pages 2040–2045, 2009.
16. M. Lawley and C. Bousque. Fast classification in protege: Snorocket as an owl2 el reasoner. In *Proceedings of Australasian Ontology Workshop*, 2010.
17. D. Maier. *The Theory of Relational Databases.* Computer Science Press, Maryland, 1983.
18. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
19. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, 1997.
20. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.* Ph.D. dissertation, Institute for Theoretical Computer Science, TU Dresden, Germany, 2009.
21. E. Thomas, J. Z. Pan, and Y. Ren. Trowl: Tractable owl 2 reasoning infrastructure. *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, (ESWC 2010)*, volume 6089 of *Lecture Notes in Computer Science*, pages 431–435. Springer-Verlag, 2010.

Fixed Parameter Tractable Reasoning in DLs via Decomposition

František Simančík, Boris Motik, and Markus Krötzsch

Department of Computer Science, University of Oxford, UK

1 Introduction

DL reasoning is of high computational complexity even for basic DLs such as $\mathcal{ALCI}$ [3, Chapter 3]. Intuitively, due to disjunctions (*or-branching*) and/or existential quantifiers (*and-branching*), a DL reasoner may need to investigate (at least) exponentially many combinations of concepts. A range of highly-tuned optimizations, such as absorption, dependency-directed backtracking, blocking, and caching [3, Chapter 9], can be used to tame these sources of complexity. None of these techniques, however, provide formal tractability guarantees. Such guarantees can be obtained by restricting the language expressivity, as done in the $\mathcal{EL}$ [2], DL-Lite [4,1], and DLP [8] families of DLs. Tractable DLs typically do not support disjunctions, which eliminates or-branching, and they either significantly restrict universal quantification (as in $\mathcal{EL}$ and DL-Lite) or disallow existential quantification (as in DLP), which eliminates or reduces and-branching.

Obtaining tractability guarantees for hard computational problems has been extensively studied in *parameterized complexity* [5]. The general idea is to measure the "hardness" of a problem instance of size n using a nonnegative integer *parameter* k, and the goal is to solve the problem in time that becomes polynomial in n whenever k is fixed. A particular goal is to identify *fixed parameter tractable* (FPT) problems, which can be solved in time $f(k) \cdot n^c$, where c is a constant and f is an arbitrary computable function that depends *only* on k. Note that not every problem that becomes tractable if k is fixed is in FPT. For example, checking whether a graph of size n contains a clique of size k can clearly be performed in time $O(n^k)$, which is polynomial if k is a constant; however, since k is in the exponent of n, this does not prove membership in FPT.

Note that every problem is FPT if the parameter is the problem's size, so a useful parameterization should allow increasing the size arbitrarily while keeping the parameter bounded. Various problems in AI were successfully parameterized by exploiting the graph-theoretic notions of *tree decompositions* and *treewidth* [6,7,10], which we recapitulate next. A *hypergraph* is a pair $G = \langle V, H\rangle$ where V is a set of vertices and $H \subseteq 2^V$ is a set of *hyperedges*. A *tree decomposition* of G is a pair $\langle T, L\rangle$ where T is an undirected tree whose sets of vertices (also called *bags*) and edges are denoted with $\mathsf{B}(T)$ and $\mathsf{E}(T)$, and $L : \mathsf{B}(T) \to 2^V$ is a labeling of $\mathsf{B}(T)$ by subsets of V such that

(T1) for each $v \in V$, the set $\{b \in \mathsf{B}(T) \mid v \in L(b)\}$ induces a connected subtree of T, and
(T2) for each $e \in H$, there exists a bag $b \in \mathsf{B}(T)$ such that $e \subseteq L(b)$.

The *width* of $\langle T, L\rangle$ is defined as $\max_{b\in\mathsf{B}(T)} L(b) - 1$. Finally, the *treewidth* of G is the minimum width among all possible tree decompositions of G. Consider now an instance N of the SAT problem, where N is a finite set of clauses (i.e., disjunctions of possibly negated propositional variables). The notions of tree decompositions and treewidth of N are defined w.r.t. the hypergraph $G_N = \langle V_N, H_N \rangle$ where V_N is the set of propositional variables occurring in N, and H_N contains the hyperedge $\{p_1, \ldots, p_k\}$ for each clause $(\neg)p_1 \vee \ldots \vee (\neg)p_k \in N$. When parameterized by treewidth, SAT is FPT [10]. Intuitively, the treewidth of N shows how many propositional variables must be considered simultaneously in order to check the satisfiability of N; thus, bounding the treewidth has the effect of bounding or-branching.

Inspired by these results, we present a novel DL reasoning algorithm that ensures fixed parameter tractability. To this end, in Section 3 we introduce a notion of a *decomposition* $\mathcal{D}$ of a signature Σ. Intuitively, $\mathcal{D}$ is a graph that restricts the propagation information between the atomic concepts in Σ. A decomposition of Σ can be seen as one or more tree decompositions, each reflecting the propagation of information due to or-branching, interconnected to reflect the propagation of information due to and-branching. We identify a parameter of $\mathcal{D}$ called *width*; intuitively, this parameter determines an upper bound on the number of concepts that must be considered simultaneously to solve a reasoning problem. Let $\mathcal{O}$ be an $\mathcal{ALCI}$ ontology *normalized* to contain only axioms of the form $\bigsqcap_i A_i \sqsubseteq \bigsqcup_j B_j$, $A \sqsubseteq \exists R.B$, and $A \sqsubseteq \forall R.B$, where $A_{(i)}$ and $B_{(j)}$ are atomic concepts, and R is a (possibly inverse) role. We present a resolution-based reasoning calculus that runs in time $O(f(d) \cdot |\mathcal{D}| \cdot |\mathcal{O}|)$, where d is the width of $\mathcal{D}$, $|\mathcal{D}|$ is the size of $\mathcal{D}$, and $|\mathcal{O}|$ is the number of axioms in $\mathcal{O}$. Our calculus is not complete for all $\mathcal{D}$: it is not guaranteed to derive all consequences that might be of interest. To remedy that, we introduce a notion of $\mathcal{D}$ being *admissible* for $\mathcal{O}$ and the relevant consequences, and we show that admissibility guarantees completeness.

Ideally, given $\mathcal{O}$ and the relevant consequences, one would identify an admissible decomposition $\mathcal{D}$ of smallest width and then run our calculus in order to obtain an FPT algorithm. In Section 4, however, we show that, for certain $\mathcal{O}$, all admissible decompositions of smallest width have exponentially many vertices. This is in contrast to tree decompositions (e.g., for each instance of SAT, a tree decomposition of minimal width exists in which the number of vertices is linear in the size of the instance) and is due to the fact that, in addition to or-branching, our decompositions analyze information flow due to and-branching as well. We therefore further restrict the notion of admissible decompositions in several ways. For each of the resulting notions, one can compute a decomposition of width at most d (if one exists) in time $f(d) \cdot |\mathcal{O}|^c$ with f a computable function and c an integer constant; together with our resolution-based calculus, we thus obtain an FPT calculus for reasoning with normalized $\mathcal{ALCI}$ ontologies.

In Section 5 we show that the minimum decomposition width of several commonly used ontologies is much smaller than the respective ontology's size. This suggests that decomposition width provides a "reasonable" measure of ontology complexity, and that our approach might even provide practical tractability guarantees.

Our results can be applied to $\mathcal{SHI}$ ontologies by transforming away role hierarchies and transitivity and normalizing the ontology in a preprocessing step. Such transformations, however, are don't-care nondeterministic, and the minimum decomposition width of the normalization result might depend on the nondeterministic choices. In this paper we thus restrict our attention to normalized $\mathcal{ALCI}$ ontologies, and we leave an investigation of how normalization affects the minimum width for future work.

$$\mathsf{R_1}\ \frac{}{A \sqsubseteq A} \qquad\qquad \mathsf{R_2}\ \frac{K_1 \sqsubseteq M_1 \sqcup A \quad A \sqcap K_2 \sqsubseteq M_2}{K_1 \sqcap K_2 \sqsubseteq M_1 \sqcup M_2}$$

$$\mathsf{R_3}\ \frac{}{K \sqsubseteq M} : K \sqsubseteq M \in \mathcal{O} \qquad\qquad \mathsf{R_4}\ \frac{\begin{array}{rr} & B \sqcap \bigsqcap_i D_i \sqsubseteq \bigsqcup_j E_j \\ \text{or} & \bigsqcap_i D_i \sqsubseteq \bigsqcup_j E_j \end{array}}{A \sqcap \bigsqcap_i C_i \sqsubseteq \bigsqcup_j F_j} : \begin{array}{l} A \sqsubseteq \exists R.B \in \mathcal{O} \\ C_i \sqsubseteq \forall R.D_i \in \mathcal{O} \\ E_j \sqsubseteq \forall R^{-}.F_j \in \mathcal{O} \end{array}$$

Fig. 1. A simple resolution calculus

2 Source of Complexity in DL Reasoning

In order to motivate the results presented in the following sections, in this section we present a very simple calculus that is not FPT, and we discuss the rough idea for making the calculus FPT. The calculus is based on resolution, and is similar to the calculus presented in [9]. Resolution can often provide worst-case optimal calculi whose best case complexity is significantly lower than the worst case complexity; indeed, the calculus from [9] has demonstrated excellent practical performance.

The calculus manipulates *clauses*—expressions of the form $K \sqsubseteq M$, where K is a finite conjunction of atomic concepts, and M is a finite disjunction of atomic concepts. With $\mathsf{sig}(K)$, $\mathsf{sig}(M)$, and $\mathsf{sig}(K \sqsubseteq M)$ we denote the sets of atomic concepts occurring in K, M, and $K \sqsubseteq M$, respectively. We consider two disjunctions (resp. conjunctions) to be the same whenever they mention the same atoms; that is, we disregard the order and the multiplicity of atoms. We write empty K and M as $\top$ and $\bot$, respectively. Furthermore, we say that a clause $K' \sqsubseteq M'$ is a *strengthening* of a clause $K \sqsubseteq M$ if $\mathsf{sig}(K') \subseteq \mathsf{sig}(K)$ and $\mathsf{sig}(M') \subseteq \mathsf{sig}(M)$. We write $K \sqsubseteq M \hat{\in} \mathcal{N}$ if the set of clauses $\mathcal{N}$ contains at least one strengthening of the clause $K \sqsubseteq M$.

Given a normalized ontology $\mathcal{O}$, our calculus constructs a *derivation*—a sequence $\mathcal{S}_0, \mathcal{S}_1, \ldots$ of sets of clauses such that $\mathcal{S}_0 = \emptyset$, and for each $i > 0$, set $\mathcal{S}_i$ is obtained from $\mathcal{S}_{i-1}$ by applying a rule from Fig. 1. Rules $\mathsf{R_1}$ and $\mathsf{R_2}$ implement propositional resolution, and rule $\mathsf{R_3}$ ensures that each clause in $\mathcal{O}$ is taken into account. Rule $\mathsf{R_4}$ handles role restrictions; letter R stands for a role (i.e., R need not be atomic), and $\mathsf{inv}(R)$ is the inverse role of R; finally, note that the atom B in the premise of the rule is optional. Intuitively, the rule says that, if B, D_i, and $\neg E_j$ jointly imply a contradiction, but $A \sqsubseteq \exists R.B$, $C_i \sqsubseteq \forall R.D_i$, and $\neg F_j \sqsubseteq \forall R.\neg E_j$ hold, then A, C_i, and $\neg F_j$ jointly imply a contradiction too. Reasoning with the second premise is analogous.

A *saturation* is defined as $\mathcal{S} := \bigcup_i \mathcal{S}_i$. The calculus *infers* a clause $K \sqsubseteq M$, written $\mathcal{O} \vdash K \sqsubseteq M$, if $K \sqsubseteq M \hat{\in} \mathcal{S}$. It is straightforward to see that the calculus is *sound*: if $\mathcal{O} \vdash K \sqsubseteq M$, then $\mathcal{O} \models K \sqsubseteq M$. Typically, resolution is used as a refutation-complete calculus; however, it is possible to show that the variant of resolution presented here is *complete* in the following stronger sense: if $\mathcal{O} \models K \sqsubseteq M$, then $\mathcal{O} \vdash K \sqsubseteq M$; note that this means that the calculus infers *at least one strengthening* of each clause entailed by $\mathcal{O}$. This stronger notion of completeness can be useful in practice; for example, $\mathcal{O}$ can be classified using a single run of the calculus, which is not the case for calculi (such as tableau) that are only refutationally complete.

Let d be the number of atomic concepts in $\mathcal{O}$. Since each clause is uniquely identified by the atomic concepts that occur in K and/or M, the calculus can derive at most 4^d clauses, which is exponential in $|\mathcal{O}|$. The high complexity of DL reasoning arises because one may have to consider exponentially many combinations of concepts, and this fact fundamentally underpins all DL reasoning algorithms. Clearly, a tractable algorithm should consider only polynomially many combinations. For example, reasoning algorithms for $\mathcal{EL}$ exploit the fact that only polynomially many combinations are "relevant" and that all of them can be constructed deterministically. In the following sections, we ensure tractability of reasoning in a radically different way. Instead of restricting the ontology language, we show that by restricting the structure of the ontology with a suitable parameter one can limit the number of concepts that must be simultaneously considered, which effectively limits the exponent in the above calculation. Since the base of the exponent not depend on $|\mathcal{O}|$, we will thus obtain an FPT reasoning calculus.

3 Reasoning with Decompositions

In this section we develop the notions of decomposition, decomposition admissibility, and the resolution calculus. We start by introducing the notion of decomposition.

Definition 1. *Let $\Sigma = \langle \Sigma_A, \Sigma_R \rangle$ be a DL signature, where Σ_A is a finite set of atomic concepts and Σ_R is a finite set of atomic roles; let $\Sigma_{R^-} = \{R^- \mid R \in \Sigma_R\}$ be the set of inverse roles of Σ_R; and let ϵ be a symbol not contained in $\Sigma_A \cup \Sigma_R \cup \Sigma_{R^-}$.*

A decomposition *of Σ is a labeled graph $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$, where $\mathcal{V}$ is a finite set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times (\Sigma_R \cup \Sigma_R^- \cup \{\epsilon\})$ is a set of directed edges labeled by a role or by ϵ, and $\mathsf{sig} : \mathcal{V} \to 2^{\Sigma_A}$ is a labeling of each vertex with a set of atomic concepts. The* width *of $\mathcal{D}$ is defined as $\mathsf{wd}(\mathcal{D}) := \max_{v \in \mathcal{V}} |\mathsf{sig}(v)|$.*

Note that $\mathcal{D}$ is not defined w.r.t. an ontology, but w.r.t. a signature Σ, and we will establish a link between $\mathcal{D}$ and $\mathcal{O}$ shortly in our notion of admissibility. This is mainly so as to gather all conditions that guarantee completeness in one place. We discuss the intuition behind this definition after presenting the resolution-based calculus.

Definition 2. *Let Σ be a DL signature, let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$ be a decomposition of Σ, and let $\mathcal{O}$ be a normalized $\mathcal{ALCI}$ ontology over Σ. The* resolution calculus *for $\mathcal{D}$ and $\mathcal{O}$ is defined as follows.*

A clause system *for $\mathcal{D}$ is a function $\mathcal{S}$ that assigns to each vertex $v \in \mathcal{V}$ a set of clauses $\mathcal{S}(v)$. A* derivation *of the calculus is a sequence of clause systems $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \ldots$ such that $\mathcal{S}_0(v) = \emptyset$ for each $v \in \mathcal{V}$ and, for each $i > 0$, $\mathcal{S}_i$ is obtained from $\mathcal{S}_{i-1}$ by an application of a derivation rule from Fig. 2; we assume that each derivation is* fair *in the usual sense. The* saturation *is the clause system $\mathcal{S}$ defined by $\mathcal{S}(v) := \bigcup_i \mathcal{S}_i(v)$ for each $v \in \mathcal{V}$. The calculus* infers *a clause $K \sqsubseteq M$ at vertex v, written $\mathcal{O}, v \vdash_{\mathcal{D}} K \sqsubseteq M$, if $K \sqsubseteq M \,\hat{\in}\, \mathcal{S}(v)$; furthermore, the calculus* infers *a clause $K \sqsubseteq M$, written $\mathcal{O} \vdash_{\mathcal{D}} K \sqsubseteq M$, if a vertex $v \in \mathcal{V}$ exists such that $\mathcal{O}, v \vdash_{\mathcal{D}} K \sqsubseteq M$.*

The calculus is complete *(*sound*) if $\mathcal{O} \models K \sqsubseteq M$ implies (is implied by) $\mathcal{O} \vdash_{\mathcal{D}} K \sqsubseteq M$ for each clause $K \sqsubseteq M$ over Σ. Given a set of clauses C over Σ, the calculus is C-*complete *if $\mathcal{O} \models K \sqsubseteq M$ implies $\mathcal{O} \vdash_{\mathcal{D}} K \sqsubseteq M$ for each $K \sqsubseteq M \in C$.*

$$\mathsf{R_1}\ \frac{}{\text{add } A \sqsubseteq A \text{ to } \mathcal{S}(v)} : A \in \mathsf{sig}(v)$$

$$\mathsf{R_2}\ \frac{K_1 \sqsubseteq M_1 \sqcup A \in \mathcal{S}(v) \quad A \sqcap K_2 \sqsubseteq M_2 \in \mathcal{S}(v)}{\text{add } K_1 \sqcap K_2 \sqsubseteq M_1 \sqcup M_2 \text{ to } \mathcal{S}(v)}$$

$$\mathsf{R_3}\ \frac{}{\text{add } K \sqsubseteq M \text{ to } \mathcal{S}(v)} : \begin{array}{l} K \sqsubseteq M \in \mathcal{O} \\ \mathsf{sig}(K \sqsubseteq M) \subseteq \mathsf{sig}(v) \end{array}$$

$$\mathsf{R_4}\ \frac{\begin{array}{rr} & B \sqcap \bigsqcap_i D_i \sqsubseteq \bigsqcup_j E_j \in \mathcal{S}(u) \\ \text{or} & \bigsqcap_i D_i \sqsubseteq \bigsqcup_j E_j \in \mathcal{S}(u) \end{array}}{\text{add } A \sqcap \bigsqcap_i C_i \sqsubseteq \bigsqcup_j F_j \text{ to } \mathcal{S}(v)} : \begin{array}{l} A \sqsubseteq \exists R.B \in \mathcal{O} \\ C_i \sqsubseteq \forall R.D_i \in \mathcal{O} \\ E_j \sqsubseteq \forall \mathsf{inv}(R).F_j \in \mathcal{O} \\ \langle u, v, R \rangle \in \mathcal{E} \\ \mathsf{sig}(A \sqcap \bigsqcap_i C_i \sqsubseteq \bigsqcup_j F_j) \subseteq \mathsf{sig}(v) \end{array}$$

$$\mathsf{R_5}\ \frac{K \sqsubseteq M \in \mathcal{S}(u)}{\text{add } K \sqsubseteq M \text{ to } \mathcal{S}(v)} : \begin{array}{l} \langle u, v, \epsilon \rangle \in \mathcal{E} \\ \mathsf{sig}(K \sqsubseteq M) \subseteq \mathsf{sig}(v) \end{array}$$

Fig. 2. The decomposition calculus

While the simple calculus from Section 2 saturates a single set of clauses, the resolution calculus for $\mathcal{D}$ and $\mathcal{O}$ saturates one set of clauses per decomposition vertex. In particular, for a vertex $v \in \mathcal{V}$, set $\mathcal{S}(v)$ contains only clauses whose propositional atoms are all contained in $\mathsf{sig}(v)$, so v identifies a propositional subproblem of $\mathcal{O}$. Rules $\mathsf{R_1}$–$\mathsf{R_3}$ implement propositional resolution "within" each vertex v. Rule $\mathsf{R_5}$ propagates propositional consequences from vertex u to vertex v connected by an ϵ-labeled edge; thus, the ϵ-labeled edges of $\mathcal{D}$ "connect" the subproblems of $\mathcal{O}$ in accordance with or-branching. Finally, rule $\mathsf{R_4}$ propagates modal consequences from a vertex u to a vertex v connected by an R-labeled edge; thus, the R-labeled edges of $\mathcal{D}$ "connect" the subproblems of $\mathcal{O}$ in accordance with and-branching. A clause is inferred if at least one saturated set $\mathcal{S}(v)$ contains a strengthening of the clause.

Note that rules $\mathsf{R_1}$–$\mathsf{R_3}$ consider only one vertex at a time, whereas rules $\mathsf{R_4}$ and $\mathsf{R_5}$ involve two vertices. Thus, although this was not our initial motivation, the calculus seems to exhibit significant parallelization potential. We leave a thorough investigation of the reasoning problem in terms of parallel complexity classes for future work.

The notion of C-completeness takes into account that one might be interested not only in refutational completeness, but in the derivation of all clauses from some set C. For example, if one is interested in the classification of $\mathcal{O}$, then C would contain all clauses of the form $A \sqsubseteq B$ with A and B atomic concepts occurring in $\mathcal{O}$.

The following proposition determines the complexity of the calculus in terms of the sizes of $\mathcal{D}$ and the number $|\mathcal{O}|$ of axioms in $\mathcal{O}$. It essentially observes two key facts: first, since the clauses in each $\mathcal{S}(v)$ are restricted to atomic concepts in $\mathsf{sig}(v)$, the maximum number of clauses in $\mathcal{S}(v)$ is determined solely by $\mathsf{wd}(\mathcal{D})$; and second, given a node or a pair of nodes, all rules can be applied in time that also depends solely on $\mathsf{wd}(\mathcal{D})$. Once we limit the size of $\mathcal{D}$, this proposition will provide us with an FPT algorithm.

Proposition 1. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$ and $\mathcal{O}$ be as in Definition 2. The saturation of the resolution calculus for $\mathcal{D}$ and $\mathcal{O}$ can be computed in time $O(f(\mathsf{wd}(\mathcal{D})) \cdot (|\mathcal{V}| + |\mathcal{E}|) \cdot |\mathcal{O}|)$, where f is some computable function.*

The rules of our calculus are clearly sound for arbitrary decompositions $\mathcal{D}$ and ontologies $\mathcal{O}$; however, the converse is not true. As a trivial example, note that the decomposition with the empty vertex and edge sets satisfies Definition 1, and that our calculus does not infer any clause using such $\mathcal{D}$. Therefore, we next introduce the notion of *admissibility*, which we later show to be sufficient for completeness.

Definition 3. *Let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$ be a decomposition of a DL signature $\Sigma = \langle \Sigma_A, \Sigma_R \rangle$.*

Let $\mathcal{W} \subseteq \mathcal{V}$ be an arbitrary set of vertices. The signature *of $\mathcal{W}$ is defined as $\mathsf{sig}(\mathcal{W}) := \bigcup_{w \in \mathcal{W}} \mathsf{sig}(w)$. The* ϵ-projection *of $\mathcal{D}$ w.r.t. $\mathcal{W}$ is the undirected graph $\mathcal{D}_{\mathcal{W}}$ that contains the undirected edge $\{u, v\}$ for each $\langle u, v, \epsilon \rangle \in \mathcal{E}$ with $u, v \in \mathcal{W}$. Set $\mathcal{W}$ is* ϵ-connected *if, for all $u, v \in W$, vertices $\{w_0, w_1, \ldots, w_n\} \subseteq \mathcal{W}$ exist such that $w_0 = u$, $w_n = v$, and $\langle w_{i-1}, w_i, \epsilon \rangle \in \mathcal{E}$ for each $1 \leq i \leq n$; furthermore, $\mathcal{W}$ is an* ϵ-component *of $\mathcal{D}$ if $\mathcal{W}$ is ϵ-connected, and each $\mathcal{W}'$ such that $\mathcal{W} \subsetneq \mathcal{W}' \subseteq \mathcal{V}$ is not ϵ-connected.*

Decomposition $\mathcal{D}$ is admissible *for an ontology $\mathcal{O}$ if $\langle u, v, \epsilon \rangle \in \mathcal{E}$ implies $\langle v, u, \epsilon \rangle \in \mathcal{E}$ for all $u, v \in \mathcal{V}$, and if each ϵ-component $\mathcal{W}$ of $\mathcal{D}$ satisfies the following properties:*

(i) *$\mathcal{D}_{\mathcal{W}}$ is an undirected tree;*
(ii) *for each atomic concept $A \in \mathsf{sig}(\mathcal{W})$, the set $\{w \in \mathcal{W} \mid A \in \mathsf{sig}(w)\}$ is ϵ-connected;*
(iii) *for each clause $K \sqsubseteq M \in \mathcal{O}$ such that $\mathsf{sig}(K) \subseteq \mathsf{sig}(\mathcal{W})$, a vertex $w \in \mathcal{W}$ exists such that $\mathsf{sig}(K \sqsubseteq M) \subseteq \mathsf{sig}(w)$;*
(iv) *for each axiom $A \sqsubseteq \exists R.B \in \mathcal{O}$ such that $A \in \mathsf{sig}(\mathcal{W})$, an ϵ-component $\mathcal{U}$ of $\mathcal{D}$ and vertices $w \in \mathcal{W}$ and $u \in \mathcal{U}$ exist such that*
  - *$\langle u, w, R \rangle \in \mathcal{E}$,*
  - *$A \in \mathsf{sig}(w)$,*
  - *$B \in \mathsf{sig}(u)$,*
  - *for each $C \sqsubseteq \forall R.D \in \mathcal{O}$, if $C \in \mathsf{sig}(\mathcal{W})$ then $C \in \mathsf{sig}(w)$ and $D \in \mathsf{sig}(u)$, and*
  - *for each $E \sqsubseteq \forall \mathsf{inv}(R).F \in \mathcal{O}$, if $E \in \mathsf{sig}(\mathcal{U})$ then $E \in \mathsf{sig}(u)$ and $F \in \mathsf{sig}(w)$.*

A clause $K \sqsubseteq M$ is covered *by $\mathcal{D}$ if an ϵ-component $\mathcal{W}$ of $\mathcal{D}$ and a vertex $w \in \mathcal{W}$ exist such that $\mathsf{sig}(K) \cup [\mathsf{sig}(M) \cap \mathsf{sig}(\mathcal{W})] \subseteq \mathsf{sig}(w)$. Decomposition $\mathcal{D}$ is* admissible *for $\mathcal{C}$ if each clause in $\mathcal{C}$ is covered by $\mathcal{D}$.*

Definition 3 incorporates two largely orthogonal ideas. First, each ϵ-component $\mathcal{W}$ of $\mathcal{D}$ reflects the propositional constraints on domain elements of a particular type in a model of $\mathcal{O}$. To deal with or-branching, each $\mathcal{W}$ is a tree decomposition formed by undirected ϵ-labeled edges. Conditions (i)–(iii) are analogous to (T1) and (T2) in Section 1, but (iii) is more general: instead of requiring $\mathsf{sig}(K \sqsubseteq M) \subseteq \mathsf{sig}(w)$ for each $K \sqsubseteq M \in \mathcal{O}$ and some $w \in \mathcal{W}$, Condition (iii) takes into account that, if $\mathsf{sig}(K) \nsubseteq \mathsf{sig}(\mathcal{W})$, then $K \sqsubseteq M$ can be satisfied by making the atomic concepts in $\mathsf{sig}(K) \setminus \mathsf{sig}(\mathcal{W})$ false on the appropriate domain element; thus, $\mathsf{sig}(K \sqsubseteq M) \subseteq \mathsf{sig}(w)$ must hold for some $w \in \mathcal{W}$ only if $\mathsf{sig}(K) \subseteq \mathsf{sig}(\mathcal{W})$. Admissibility for $\mathcal{C}$ uses an analogous idea.

Second, to deal with and-branching, the ϵ-components of $\mathcal{D}$ are interconnected via role-labeled edges. If a concept A occurs in an ϵ-component $\mathcal{W}$ and in an axiom of $\mathcal{O}$ of the form $A \sqsubseteq \exists R.B$, then a domain element corresponding to $\mathcal{W}$ might need to have an R-successor; to reflect that, $\mathcal{D}$ must contain an ϵ-component $\mathcal{U}$, and vertices $w \in \mathcal{W}$ and $u \in \mathcal{U}$ connected by an R-labeled edge must exist such that $A \in \mathsf{sig}(w)$ and $B \in \mathsf{sig}(u)$. Furthermore, in order to address the universal quantifiers over R, if $C \sqsubseteq \forall R.D \in \mathcal{O}$ and $C \in \mathsf{sig}(\mathcal{W})$, then $C \in \mathsf{sig}(w)$ and $D \in \mathsf{sig}(u)$ must hold, and analogously for universals over $\mathsf{inv}(R)$. These conditions ensure that w and u contain all atomic concepts that might be relevant for modal reasoning, which in turn allows our calculus to infer all relevant constrains on atomic concepts.

The following theorem shows that admissibility indeed ensures completeness.

Theorem 1. *Let $\mathcal{O}$ be an ontology, let $\mathcal{C}$ be a set of clauses, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$ be a decomposition that is admissible for $\mathcal{O}$ and $\mathcal{C}$. Then, the resolution calculus for $\mathcal{D}$ and $\mathcal{O}$ is $\mathcal{C}$-complete.*

Ideally, given an ontology $\mathcal{O}$ and a set of clauses $\mathcal{C}$, one would identify a decomposition $\mathcal{D}$ of smallest width and then apply the resolution calculus for $\mathcal{D}$ and $\mathcal{O}$ to obtain an FPT algorithm. The following theorem shows, however, that this idea does not work, since it is not the case that, for each ontology $\mathcal{O}$, there exists a decomposition of minimal width that is admissible for $\mathcal{O}$ and whose size is polynomial in $|\mathcal{O}|$. In order to address this problem, in Section 4 we further restrict the notion of admissibility.

Theorem 2. *A family of $\mathcal{ALCI}$ ontologies $\{\mathcal{O}_n\}$ exists such that each decomposition admissible for $\mathcal{O}_n$ and $\mathcal{C} = \{C \sqsubseteq \bot\}$ of minimal width has size exponential in $|\mathcal{O}_n|$.*

4 Constructing Decompositions of Polynomial Size

In Section 4.3 we present a general method for computing admissible decompositions of polynomial size, for which we obtain the desired FPT result. This method embodies two largely orthogonal ideas, each of which we present separately for didactic purposes. In particular, in Section 4.1 we present an approach for analyzing and-branching, and in Section 4.2 we present an approach for analyzing or-branching.

4.1 Analyzing And-Branching via Deductive Overestimation

In this section we present an approach for analyzing and-branching, which is inspired by the reasoning algorithm for $\mathcal{EL}$ [2]. The approach uses an overestimation of the subsumption relation to construct the decomposition. It manipulates expressions of the form $K \rightsquigarrow A$, where K is a conjunction of atomic concepts, and A is an atomic concept. Given an $\mathcal{ALCI}$ ontology $\mathcal{O}$ and a set of clauses $\mathcal{C}$, the *deductive overestimation* $\rightsquigarrow$ for $\mathcal{O}$ and $\mathcal{C}$ is the relation obtained by exhaustive application of the rules shown in Fig. 3.

Intuitively, $K \rightsquigarrow A$ states that an object whose existence is required to satisfy K can become an instance of A. On $\mathcal{EL}$ ontologies $\rightsquigarrow$ coincides with the subsumption relation, but on more expressive ontologies $\rightsquigarrow$ overestimates the subsumption relation. In order to check whether a clause $K \sqsubseteq M \in \mathcal{C}$ is entailed by $\mathcal{O}$, rule $\mathbf{E_1}$ introduces an instance of all atomic concepts in K. Rule $\mathbf{E_2}$ addresses the fact that, if some object α is an instance of $A_1, \ldots, A_n$ and $\mathcal{O}$ contains a clause $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B_1 \sqcup \ldots \sqcup B_m$, then the

$$\mathsf{E_1}\ \frac{}{K \rightsquigarrow A_1 \quad \dots \quad K \rightsquigarrow A_n} : \begin{array}{l} A_1 \sqcap \dots \sqcap A_n \sqsubseteq B_1 \sqcup \dots \sqcup B_m \in C \\ K = A_1 \sqcap \dots \sqcap A_n \end{array}$$

$$\mathsf{E_2}\ \frac{K \rightsquigarrow A_1 \quad \dots \quad K \rightsquigarrow A_n}{K \rightsquigarrow B_1 \quad \dots \quad K \rightsquigarrow B_m} : A_1 \sqcap \dots \sqcap A_n \sqsubseteq B_1 \sqcup \dots \sqcup B_m \in O$$

$$\mathsf{E_3}\ \frac{K \rightsquigarrow A}{B \rightsquigarrow B} : A \sqsubseteq \exists R.B \in O$$

$$\mathsf{E_4}\ \frac{K \rightsquigarrow A \quad K \rightsquigarrow C}{B \rightsquigarrow D} : \begin{array}{l} A \sqsubseteq \exists R.B \in O \\ C \sqsubseteq \forall R.D \in O \end{array} \qquad \mathsf{E_5}\ \frac{K \rightsquigarrow A \quad B \rightsquigarrow E}{K \rightsquigarrow F} : \begin{array}{l} A \sqsubseteq \exists R.B \in O \\ E \sqsubseteq \forall R^{-}.F \in O \end{array}$$

Fig. 3. Computing the deductive overestimation for O and C

object must be an instance of some B_i. Since a polynomial overestimation method that reasons by case is unlikely to exist, rule $\mathsf{E_2}$ overestimates the subsumption relation by saying that α can be an instance of all $B_1, \dots, B_m$. Rule $\mathsf{E_3}$ takes into account that, given $A \sqsubseteq \exists R.B \in O$, each instance of A needs an R-successor that is an instance of B. Analogously to the $\mathcal{EL}$ reasoning calculus, in order to obtain a polynomial overestimation method, rule $\mathsf{E_3}$ "reuses" the same successor to satisfy multiple existential restrictions to the same concept B. Finally, rules $\mathsf{E_4}$ and $\mathsf{E_5}$ implement modal reasoning.

Having computed $\rightsquigarrow$, we construct the decomposition $\mathcal{D}_{\mathsf{E}} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig} \rangle$ of the symbols occurring in O and C as shown below. Note that $\mathcal{D}_{\mathsf{E}}$ contains no ϵ-labeled edges, as this decomposition method does not analyze or-branching. By Theorems 1 and 3, the resolution calculus for $\mathcal{D}_{\mathsf{E}}$ and O is C-complete.

$$\mathcal{V} := \{v_K \mid K \rightsquigarrow A \text{ for some } A\} \qquad \mathsf{sig}(v_K) := \{A \mid K \rightsquigarrow A\}$$
$$\mathcal{E} := \{\langle v_B, v_K, R \rangle \mid K \rightsquigarrow A \text{ and } A \sqsubseteq \exists R.B \in O\}$$

Theorem 3. *Decomposition $\mathcal{D}_{\mathsf{E}}$ is admissible for O and C.*

4.2 Analyzing Or-Branching via Tree Decomposition

We now present an approach for computing admissible decompositions that analyzes or-branching. The approach handles the clauses in O as explained in Section 1 for SAT, and it imposes additional constraints in order to satisfy condition (iv) of Definition 3.

Given a normalized ontology O and a set of clauses C, we define the hypergraph $G_{O,C} = \langle V, H \rangle$ such that V and H are the smallest sets satisfying the following properties. For each atomic concept A occurring in O or C, we have $A \in V$. For each clause $K \sqsubseteq M \in O$, we have $\mathsf{sig}(K \sqsubseteq M) \in H$. For each $A \sqsubseteq \exists R.B \in O$, set H contains hyperedges $\mathsf{dom}_{A \sqsubseteq \exists R.B}$ and $\mathsf{ran}_{A \sqsubseteq \exists R.B}$ defined as shown below, where $C_i \sqsubseteq \forall R.D_i$, $1 \le i \le n$ and $E_j \sqsubseteq \forall \mathsf{inv}(R).F_j$, $1 \le j \le m$ are all axioms in O of the respective forms:

$$\mathsf{dom}_{A \sqsubseteq \exists R.B} := \{A, C_1, \dots, C_n, F_1, \dots, F_m\},$$
$$\mathsf{ran}_{A \sqsubseteq \exists R.B} := \{B, D_1, \dots, D_n, E_1, \dots, E_m\}.$$

Finally, $\mathsf{sig}(K \sqsubseteq M) \in H$ for each $K \sqsubseteq M \in C$.

Given a tree decomposition $\langle T, L\rangle$ of $G_{O,C}$, we construct (don't-care nondeterministically) a decomposition $\mathcal{D}_{\mathsf{T}} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig}\rangle$ as follows. The vertices of $\mathcal{D}_{\mathsf{T}}$ are the bags of T—that is, $\mathcal{V} := \mathsf{B}(T)$. The signatures of $\mathcal{D}_{\mathsf{T}}$ are the labels of T—that is, $\mathsf{sig} := L$. The ϵ-edges of $\mathcal{D}_{\mathsf{T}}$ are the edges of T—that is, for each $\{u, v\} \in \mathsf{E}(T)$, we have $\langle u, v, \epsilon\rangle \in \mathcal{E}$. Finally, for each $A \sqsubseteq \exists R.B \in O$, choose vertices $u, v \in \mathcal{V}$ such that $\mathsf{ran}_{A\sqsubseteq\exists R.B} \subseteq L(u)$ and $\mathsf{dom}_{A\sqsubseteq\exists R.B} \subseteq L(v)$ and set $\langle u, v, R\rangle \in \mathcal{E}$; such u and v exist due to property (T2) of the definition of tree decompositions in Section 1.

Theorem 4. *Every decomposition $\mathcal{D}_{\mathsf{T}}$ is admissible for O and C.*

4.3 Analyzing And- and Or-Branching Simultaneously

We now show how to combine the approaches for analyzing and- and or-branching to obtain a C-*decomposition* of a normalized $\mathcal{ALCI}$ ontology O and a set of clauses C.

The procedure consists of three steps. First, we compute the relation $\rightsquigarrow$ as described in Section 4.1. This step analyzes the and-branching inherent in O and C.

Second, for all K such that $K \rightsquigarrow A$ for some A, we simultaneously define hypergraphs $G_K = \langle V_K, H_K\rangle$ where $V_K := \{A \mid K \rightsquigarrow A\}$, and H_K are the smallest sets satisfying the following conditions. For each clause $K' \sqsubseteq M' \in O$ with $\mathsf{sig}(K' \sqsubseteq M') \subseteq V_K$, we have $\mathsf{sig}(K' \sqsubseteq M') \in H_K$. For each axiom $A \sqsubseteq \exists R.B \in O$ such that $A \in V_K$, set H_K contains hyperedge $\mathsf{dom}_{K,A\sqsubseteq\exists R.B}$ and set H_B contains hyperedge $\mathsf{ran}_{K,A\sqsubseteq\exists R.B}$ defined below, where $C_i \sqsubseteq \forall R.D_i$, $1 \leq i \leq n$ and $E_j \sqsubseteq \forall \mathsf{inv}(R).F_j$, $1 \leq j \leq m$ are all axioms in O of the respective forms such that $C_i \in V_K$ and $E_j \in V_B$:

$$\mathsf{dom}_{K,A\sqsubseteq\exists R.B} := \{A, C_1, \ldots, C_n, F_1, \ldots, F_m\},$$
$$\mathsf{ran}_{K,A\sqsubseteq\exists R.B} := \{B, D_1, \ldots, D_n, E_1, \ldots, E_m\}.$$

Finally, $[\mathsf{sig}(K \sqsubseteq M) \cap V_K] \in H_K$ for each $K \sqsubseteq M \in C$.

Third, we compute a tree decomposition $\langle T_K, L_K\rangle$ for each hypergraph G_K; without loss of generality we assume that all sets $\mathsf{B}(T_K)$ are disjoint. We then construct the decomposition $\mathcal{D}_{\mathsf{C}} = \langle \mathcal{V}, \mathcal{E}, \mathsf{sig}\rangle$ as follows. The vertices of $\mathcal{D}_{\mathsf{C}}$ are the bags of the tree decompositions—that is, $\mathcal{V} := \bigcup_K \mathsf{B}(T_K)$. The signatures of $\mathcal{D}_{\mathsf{C}}$ are the labels of the tree decompositions—that is, $\mathsf{sig} := \bigcup_K L_K$. The ϵ-edges of $\mathcal{D}_{\mathsf{C}}$ are the edges of the tree decompositions—that is, $\langle u, v, \epsilon\rangle \in \mathcal{E}$ for each $\{u, v\} \in \mathsf{E}(T_K)$. Finally, for each axiom $A \sqsubseteq \exists R.B \in O$ and each K such that $A \in V_K$, choose $u \in \mathsf{B}(V_B)$ and $v \in \mathsf{B}(V_K)$ such that $\mathsf{ran}_{K,A\sqsubseteq\exists R.B} \subseteq L(u)$ and $\mathsf{dom}_{K,A\sqsubseteq\exists R.B} \subseteq L(v)$ and set $\langle u, v, R\rangle \in \mathcal{E}$; such u and v exist due to property (T2) of the definition of tree decompositions in Section 1.

The class of all C-*decompositions* of O and C consists of all decompositions obtained in the way specified above. Note that the first step (computation of $\rightsquigarrow$) is deterministic, but the second step is not as each G_K may admit several tree decompositions. The C-*width* of O and C is the minimal width of any C-decomposition of O and C.

Theorem 5. *Every decomposition $\mathcal{D}_{\mathsf{C}}$ is admissible for O and C.*

To show that DL reasoning is FPT if the C-width is bounded, we next estimate the effort required for computing a C-decomposition of O and C. With $\|O\|$ and $\|C\|$ we denote the sizes of (i.e. the numbers of symbols required to encode) O and C, respectively.

Table 1. Upper bounds on **C**-width for classification

Ontology	$\|\Sigma_A\|$	$\|\Sigma_A^{norm}\|$	$\mathsf{wd}(\mathcal{D}_{\mathbf{E}})$	$\mathsf{wd}(\mathcal{D}_{\mathbf{C}})$
SNOMED CT (http://ihtsdo.org/snomed-ct/)	315,489	516,703	349	100
SNOMED CT-SEP (see [9] for reference)	54,973	149,839	1,196	168
FMA (http://fma.biostr.washington.edu/)	41,700	81,685	1,166	35
GALEN (http://opengalen.org/)	23,136	49,245	646	54
OBI (http://obi-ontology.org/)	2,955	4,296	304	45

Proposition 2. *An algorithm exists that takes as input a positive integer d, a normalized $\mathcal{ALCI}$ ontology O, and a set of clauses C, that runs in time $O(g(d) \cdot (||O|| + ||C||)^5)$ for g a computable function, and that computes a* **C***-decomposition of O and C of width at most d whenever at least one such decomposition exists.*

We can now formulate the main FPT result for **C**-decompositions.

Theorem 6. *Let d be a positive integer, let O be a normalized $\mathcal{ALCI}$ ontology, and let $K \sqsubseteq M$ be a clause. The problem of deciding whether a* **C***-decomposition of O and $C = \{K \sqsubseteq M\}$ of width at most d exists, and if so, whether $O \models K \sqsubseteq M$, is FPT.*

5 Experimental Results

It can be argued that FPT is interesting only if the parameter can be substantially smaller than the input size. In order to judge the “usefulness” of **C**-width as a complexity measure, we measured the **C**-width of several ontologies (listed in Table 1) that are often used for evaluating DL reasoners. We weakened all ontologies to $\mathcal{ALCHI}$ by discarding all unsupported features, we applied the structural transformation from [9], and we eliminated role inclusion axioms by unfolding the role hierarchy into universal restrictions to obtain normalized $\mathcal{ALCI}$ ontologies. Note that there are several different ways of formulating and optimizing structural transformation, and each could produce an ontology of a different **C**-width, so our results are not necessarily optimal.

After normalization, we next computed the deductive overestimation $\rightsquigarrow$ and the decomposition $\mathcal{D}_{\mathbf{E}}$ as described in Section 4.1, we constructed the hypergraphs G_K as described in Section 4.3, and we fed all of them into TreeD[1]—a library for computing tree decompositions—to construct a **C**-decomposition $\mathcal{D}_{\mathbf{C}}$. For each ontology we considered two sets of goal clauses: $C_1 = \{A \sqsubseteq \bot \mid A \in \Sigma_A\}$, which corresponds to checking satisfiability of all atomic concepts, and $C_2 = \{A \sqsubseteq B \mid A, B \in \Sigma_A\}$, which corresponds to classification. In theory, the **C**-width of O and C_1 can be smaller than the **C**-width of O and C_2; however, we have not observed a difference between the two in practice, so we present here only the results for classification. Also, please note that TreeD was able only to produce approximate, rather than exact tree decompositions; hence, our results provide only an upper bound on the **C**-width.

The results of our experiments are shown in Table 1. For each ontology we list the number of atomic concepts in the original ontology ($|\Sigma_A|$), the number of atomic

[1] http://www.itu.dk/people/sathi/treed/

concepts after normalization ($|\Sigma_A^{norm}|$), and the widths of the two decompositions that we constructed. Notice that although some of the tested ontologies contain tens or even hundreds of thousands of concepts, the width of $\mathcal{D}_\mathbf{C}$ rarely exceeds one hundred, and it is always by several orders of magnitude smaller than the total number of concepts in the ontology. This suggests that our notion of a decomposition might even prove to be useful in practice, provided that our resolution algorithm is suitably optimized.

6 Conclusion

We presented a DL reasoning algorithm that is fixed parameter tractable for a suitable notion of the input width. We see two main challenges for our future work. On the theoretical side, our approach should be extended to more complex ontology languages; handling counting seems particularly challenging. On the practical side, our algorithm should be optimized for practical use. A particular challenge is to combine the construction of a decomposition with actual reasoning and thus save preprocessing time.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite Family and Relations. Journal of Artificial Intelligence Research 36, 1–69 (2009)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI 2005). pp. 364–369. Morgan Kaufmann Publishers, Edinburgh, UK (July 30–August 5 2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2nd edn. (August 2007)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. Journal of Automated Reasoning 9, 385–429 (2007)
5. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
6. Gottlob, G., Pichler, R., Wei, F.: Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006). pp. 250–256. AAAI Press, Boston, MA, USA (2006)
7. Gottlob, G., Scarcello, F., Sideri, M.: Fixed-parameter complexity in AI and nonmonotonic reasoning. Artificial Intelligence 138(1–2), 55–86 (2002)
8. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: Proc. of the 12th Int. World Wide Web Conference (WWW 2003). pp. 48–57. ACM Press, Budapest, Hungary (May 20–24 2003)
9. Simančík, F., Kazakov, Y., Horrocks, I.: Consequence-Based Reasoning beyond Horn Ontologies. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011) (July 16–22 2011), to appear
10. Szeider, S.: On Fixed-Parameter Tractable Parameterizations of SAT. In: Giunchiglia, E., Tacchella, A. (eds.) Proc. of the 6th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Selected Revised Papers. LNCS, vol. 2919, pp. 188–202. Springer, Santa Margherita Ligure, Italy (May 5–8 2003)

Repairing Incomplete Reasoners

Giorgos Stoilos and Bernardo Cuenca Grau

Oxford University Computing Laboratory
Wolfson Building, Parks Road, OX1 3QD, Oxford

Abstract. The pressing need for scalable query answering has motivated the development of many *incomplete* ontology-based reasoners. Improving the completeness of such systems without sacrificing their favourable performance would be beneficial to many applications. In this paper, we address the following problem: given a query q, a TBox $\mathcal{T}$ and an incomplete reasoner ans (i.e., one that accepts q and $\mathcal{T}$ as valid inputs but fails to return all query answers to q w.r.t. $\mathcal{T}$ and some dataset), we aim at computing a (hopefully small) TBox $\mathcal{R}$ (a *repair*) which logically follows from $\mathcal{T}$ and such that ans is *provably complete* w.r.t. q and $\mathcal{T} \cup \mathcal{R}$, regardless of the input data. We identify conditions on q, $\mathcal{T}$ and ans that are sufficient to ensure the existence of such repair and present a practical repair generation algorithm. Our experiments suggest that repairs of small size can be computed for well-known ontologies and reasoners.

1 Introduction

In Semantic Web applications, a description logic TBox is frequently used for describing the meaning of data stored in various sources. A query language (based on conjunctive queries) is then used for data access [13]. Unfortunately, when using an expressive ontology language, computing query answers can be costly, and in a Semantic Web setting, datasets may be extremely large.

Motivated by the need for scalable query answering, there has been a growing interest in the development of *incomplete* ontology-based reasoning systems, which are not guaranteed to compute all query answers for some combinations of queries, TBoxes and datasets accepted as valid inputs. Many such systems (e.g., Jena, OWLim, and Oracle's Semantic Store) are based on rule technologies; others are based on approximate reasoning techniques [7, 11, 1].

Incomplete query answers, however, may not be acceptable in some cases, and even if some incompleteness is acceptable, computing as many answers as possible *without affecting performance* would be beneficial to many applications. Not surprisingly, many techniques for improving the completeness of (incomplete) reasoning systems have been proposed in recent years. A widely use approach relies on the materialisation of certain kinds of TBox consequences, prior to accessing the data. TBox materialisation is indeed a convenient and low-cost solution, as it does not rely on modifying the internals of the reasoning system at hand: it can either be included by reasoning systems' implementors as a pre-processing step (e.g., DLEJena [3], PelletDB, TrOWL [11], Minerva [10] and DLDB [6]), or it can be performed by application designers, who have little knowledge of and/or control over the reasoner. This approach, however, exhibits several limitations:

1. TBox materialisation is done blindfold, without taking into account neither the capabilities of the incomplete reasoner under consideration, nor the precise sources of its incompleteness for the application at hand.
2. In order to avoid a blowup in the size of the TBox, materialisation involves only simple atomic subsumption relations, and relevant entailments describing more complex dependencies are typically ignored.
3. It is often difficult, or even infeasible, to estimate the extent to which materialisation has improved the system's completeness for the given application at hand, especially if the data is very large, unknown, or frequently changing.

In this paper, we present a novel approach to TBox materialisation that attempts to address many of these limitations. Given a TBox $\mathcal{T}$ expressed in a language $\mathcal{L}$, a conjunctive query (CQ) q, and a reasoning system ans satisfying certain properties known in advance (e.g., soundness), we aim at computing a (hopefully small) set of TBox axioms $\mathcal{R}$, called a $(q, \mathcal{T})$-*repair* for ans, such that $\mathcal{R}$ logically follows from $\mathcal{T}$ and the system is guaranteed to compute all answers to q w.r.t. $\mathcal{T} \cup \mathcal{R}$(and hence also w.r.t. $\mathcal{T}$), regardless of the input data.

Our TBox materialisation techniques are "guided" by both the input query and TBox and the capabilities of the reasoner at hand, thus limiting the number of materialised consequences. Furthermore, our approach provides a *provable completeness guarantee*: after extending $\mathcal{T}$ with a $(q, \mathcal{T})$-repair, the incomplete reasoner at hand is *indistinguishable* from a complete one w.r.t. q and $\mathcal{T}$, and regardless of the data (which is often unknown or frequently changing). Finally, if a $(q, \mathcal{T})$-repair does not exist for the given reasoner, we also provide means for quantifying the extent to which the reasoner's completeness improved upon materialisation of a given set of TBox consequences.

Our main contributions can be summarised as follows:

– We formalise the notion of a repair for a given CQ, TBox and reasoner.
– We identify sufficient conditions for the existence of a repair. Our conditions establish a bridge between the framework of completeness evaluation [16, 15] and the notion of interpolation in the context of DLs [8, 14, 9].
– We then present a practical algorithm that is guaranteed to compute a repair under certain assumptions. Our algorithm extends well-known query rewriting techniques for DLs [12, 2] with means for computing suitable *interpolants* for each query in the rewriting.
– Although the assumptions required by our algorithm may seem somewhat strict, we provide empirical evidence that repairs can be computed for well-known ontologies and incomplete reasoners. Furthermore, the size of such repairs is surprisingly small and preliminary evaluation has shown that their impact on performance is negligible. Finally, for reasoner which are complete only for a very weak description logic, we show that our techniques can provide "partial repairs", whose impact can be quantified.

Although our main focus is on repairing rule-based reasoners, we feel that our techniques are also highly relevant to recent DL approximation frameworks [11, 1], and provide new insights into the process of approximation.

Proofs for all lemmas and theorems can be found online.[1]

2 Preliminaries

We use DLs that do not allow for nominals and we also assume that only atomic assertions are allowed in ABoxes.

Let $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{I}$ be countable, pairwise disjoint sets of *atomic concepts*, *atomic roles*, and *individuals*. An *$\mathcal{ELHI}$-role* is either an atomic role P or its *inverse* P^-. The set of $\mathcal{ELHI}$-concepts is defined inductively by the following grammar, where $A \in \mathbf{C}$, R is an $\mathcal{ELHI}$-role, and $C_{(i)}$ are $\mathcal{ELHI}$-concepts:

$$C := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists R.C$$

An $\mathcal{ELHI}$-TBox $\mathcal{T}$ is a finite set of GCIs $C_1 \sqsubseteq C_2$ with C_i $\mathcal{ELHI}$-concepts and RIAs $R_1 \sqsubseteq R_2$ with R_i $\mathcal{ELHI}$-roles. An ABox $\mathcal{A}$ is a finite set of assertions $A(a)$ or $P(a,b)$, for $A \in \mathbf{C}$, $P \in \mathbf{R}$, and $a, b \in \mathbf{I}$. An $\mathcal{ELHI}$-ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ consists of an $\mathcal{ELHI}$-TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Moreover, the DLP fragment of $\mathcal{ELHI}$ [5] (DLP-$\mathcal{ELHI}$) is obtained from $\mathcal{ELHI}$ by disallowing concepts of the form $\exists R.C$ on the right-hand side of GCIs.

A *conjunctive query* (CQ) q is an expression of the form

$$q(x_1, \ldots, x_n) \leftarrow \alpha_1, \ldots, \alpha_m$$

where each α_i is a concept or role atom of the form $A(t)$ or $R(t,t')$ (for t, t' function-free terms and A, R atomic) and each x_j is a *distinguished* variable occurring in some α_i. The remaining variables are called undistinguished. We use the standard notion of a certain answer to q w.r.t. $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ and we denote with $\mathsf{cert}(q, \mathcal{O})$ the set of all certain answers to q w.r.t. $\mathcal{O}$. Given q_1, q_2 with the same distinguished variables $\vec{x}$ we say that q_1 *subsumes* q_2 w.r.t. $\mathcal{T}$, written $q_2 \sqsubseteq_{\mathcal{T}} q_1$, if the FO-entailment $\mathcal{T} \models \forall \vec{x}(B_{q_2} \rightarrow B_{q_1})$ holds, with B_{q_i} the formula obtained from the conjunction of all body atoms in q_i by existentially quantifying over all undistinguished variables.

Finally, a *UCQ rewriting u* for a CQ q and TBox $\mathcal{T}$ is a union of conjunctive queries (a set of CQs with the same distinguished variables) that satisfies the following properties for each ABox $\mathcal{A}$ such that $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is consistent [2]: $\mathsf{cert}(q', \mathcal{A}) \subseteq \mathsf{cert}(q, \mathcal{O})$ for each $q' \in u$, and $\mathsf{cert}(q, \mathcal{O}) \subseteq \bigcup_{q' \in u} \mathsf{cert}(q', \mathcal{A})$.

3 Completeness Repairs

We start by recalling from [16, 15] the notions of a CQ answering algorithm, and of completeness for a query q and TBox $\mathcal{T}$.

[1] https://rapidshare.com/files/460900188/main.pdf

Definition 1. *A* CQ answering algorithm ans *for a DL* $\mathcal{L}$ *is a procedure that, for each* $\mathcal{L}$*-ontology* $\mathcal{O}$ *and CQ* q *computes in a finite number of steps a set* $\mathsf{ans}(q, \mathcal{O})$ *of tuples of constants. It is* sound *if* $\mathsf{ans}(q, \mathcal{O}) \subseteq \mathsf{cert}(q, \mathcal{O})$ *for each* $\mathcal{O}$ *and* q*. It is* complete *if* $\mathsf{cert}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O})$ *for each* $\mathcal{O}$ *and* q*. It is* monotonic *if* $\mathsf{ans}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O}')$ *for each* $\mathcal{O}$, $\mathcal{O}'$ *and* q *with* $\mathcal{O} \subseteq \mathcal{O}'$*. It is* invariant under renamings *if, for each* $q, \mathcal{O} = \mathcal{T} \cup \mathcal{A}, \mathcal{O}' = \mathcal{T} \cup \mathcal{A}$ *and isomorphism* ν *between* $\mathcal{A}$ *and* $\mathcal{A}'$ *we have* $\vec{a} \in \mathsf{ans}(q, \mathcal{O})$ *if and only if* $\nu(\vec{a}) \in \mathsf{ans}(q, \mathcal{O}')$*. It is* well-behaved *if it is sound, monotonic and invariant under renamings. We denote with* $\mathcal{C}^{\mathcal{L}}$ *the class of all well-behaved CQ answering algorithms for* $\mathcal{L}$*.*

Finally, we say that ans *is* $(q, \mathcal{T})$-complete *for a query* q *and TBox* $\mathcal{T}$ *if for each ABox* $\mathcal{A}$ *s.t.* $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ *is consistent, we have that* $\mathsf{cert}(q, \mathcal{O}) \subseteq \mathsf{ans}(q, \mathcal{O})$*.*

This general definition allow us to abstract from the specifics of implemented systems. All incomplete reasoning algorithms known to us are well-behaved: they are sound, query answers can only grow if we add axioms to the ontology, and they do not depend on trivial renamings of ABox individuals. Furthermore, a $(q, \mathcal{T})$-complete algorithm, even if incomplete in general, behaves exactly like a complete one w.r.t. the given query q and TBox $\mathcal{T}$, regardless of the data. Consider, as a running example, the following $\mathcal{ELHI}$-TBox $\mathcal{T}_0$ and query q_0:

$$\begin{aligned}
\mathcal{T}_0 \;=\; &\{\exists \mathsf{takes}.\mathsf{Course} \sqsubseteq \mathsf{Student}, \mathsf{GradCo} \sqsubseteq \mathsf{Course}, \\
&\mathsf{GradSt} \sqsubseteq \exists \mathsf{takes}.\mathsf{GradCo}, \mathsf{PhDSt} \sqsubseteq \mathsf{GradSt},\ \mathsf{Student} \sqcap \mathsf{Course} \sqsubseteq \bot\} \\
q_0(x) \leftarrow &\;\mathsf{Student}(x)
\end{aligned}$$

Consider an algorithm ans_0 that first translates DLP-$\mathcal{ELHI}$ GCIs in $\mathcal{T}_0$ into a datalog program using standard transformations (and discarding the remaining GCIs), then saturates the input ABox w.r.t. the program, and finally answers q_0 w.r.t. the saturated ABox. Clearly, ans_0 is not $(q_0, \mathcal{T}_0)$-complete: it returns the empty set for $\mathcal{A} = \{\mathsf{GradSt}(a)\}$, whereas $\mathsf{cert}(q_0, \mathcal{T}_0 \cup \mathcal{A}) = \{a\}$. Computing the missing answer requires axiom $\mathsf{GradSt} \sqsubseteq \exists \mathsf{takes}.\mathsf{GradCo}$, which is discarded. Note, however, that one could dispense with the discarded axiom and still recover the missing answer by extending $\mathcal{T}_0$ with the following DLP-$\mathcal{ELHI}$ TBox $\mathcal{R}_0$:

$$\mathcal{R}_0 = \{\mathsf{GradSt} \sqsubseteq \mathsf{Student}\} \tag{1}$$

Since $\mathcal{T}_0 \models \mathcal{R}_0$, extending $\mathcal{T}_0$ with $\mathcal{R}_0$ does not change the meaning of $\mathcal{T}_0$. The behaviour of ans_0 w.r.t. $\mathcal{T}_0 \cup \mathcal{R}_0$, however, is now different because ans_0 translates $\mathcal{R}_0$ into a datalog clause and hence $\mathsf{ans}_0(q_0, \mathcal{T}_0 \cup \mathcal{R}_0 \cup \mathcal{A}) = \{a\}$.

Note also that adding $\mathcal{R}_0$ allows us to recover missing answers w.r.t. many other ABoxes different from $\mathcal{A}$. For example, given $\mathcal{A}' = \{\mathsf{PhDSt}(b)\}$ and $\mathcal{T}_0 \cup \mathcal{A}'$, we have that b is a certain answer to q_0 that is computed by ans_0 only after extending $\mathcal{T}_0$ with $\mathcal{R}_0$.

Our example suggests that given q, $\mathcal{T}$ and an algorithm ans that is incomplete for q and $\mathcal{T}$, it may be possible to recover all missing answers to q (w.r.t. all possible ABoxes) by "repairing" $\mathcal{T}$ for ans—that is, by materialising a (hopefully small) number of $\mathcal{T}$-consequences that ans can process.

Definition 2. *Let $\mathcal{L}$ be a DL, q a CQ, $\mathcal{T}$ an $\mathcal{L}$-TBox, $\mathcal{A}$ an ABox, and* ans *a CQ answering algorithm for $\mathcal{L}$. A $(q, \mathcal{T}, \mathcal{A})$*-repair *$\mathcal{R}$ for* ans *is an $\mathcal{L}$-TBox such that*

1. *$\mathcal{T} \models \mathcal{R}$; and*
2. *$\mathsf{cert}(q, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(q, \mathcal{T} \cup \mathcal{R} \cup \mathcal{A})$.*

Given a set $\mathbf{A}$ of ABoxes, we say that $\mathcal{R}$ is a $(q, \mathcal{T}, \mathbf{A})$-repair *for* ans *if it is a $(q, \mathcal{T}, \mathcal{A})$-repair for* ans *for each $\mathcal{A} \in \mathbf{A}$. Finally, we say that $\mathcal{R}$ is a $(q, \mathcal{T})$*-repair *for* ans *if it is a $(q, \mathcal{T}, \mathcal{A})$-repair for* ans *for every ABox $\mathcal{A}$ s.t. $\mathcal{T} \cup \mathcal{A}$ is consistent.*

Unfortunately, deciding, on the one hand, whether ans is $(q, \mathcal{T})$-complete and, on the other hand, whether $\mathcal{R}$ is a $(q, \mathcal{T})$-repair for ans may involve computing query answers w.r.t. an unlimited number of ABoxes. In [16, 15], however, it was shown that under certain conditions on q and $\mathcal{T}$ it is possible to decide $(q, \mathcal{T})$-completeness by computing query answers only w.r.t. a finite (and hopefully small) number of ABoxes. Such finite set of ABoxes is called a *testing base*.

Definition 3. *Let $\mathcal{L}$ be a description logic, let $\mathcal{C}$ be a class of CQ answering algorithms for $\mathcal{L}$, let $\mathcal{T}$ be an $\mathcal{L}$-TBox, and let q be a CQ. A $(q, \mathcal{T})$*-testing base *$\mathbf{B}$ for $\mathcal{C}$ is a finite set of ABoxes $\mathcal{A}$ such that the following property holds for each algorithm* ans *in $\mathcal{C}$: if $\mathsf{cert}(q, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(q, \mathcal{T} \cup \mathcal{A})$ for each $\mathcal{A} \in \mathbf{B}$, then $\mathsf{cert}(q, \mathcal{T} \cup \mathcal{A}) \subseteq \mathsf{ans}(q, \mathcal{T} \cup \mathcal{A})$ for each ABox $\mathcal{A}'$ consistent with $\mathcal{T}$.*

In our previous work [16, 15] we have shown how to compute a $(q, \mathcal{T})$-testing base (if one exists) for the class $\mathcal{C}^{\mathcal{L}}$ of well-behaved algorithms.

Consider our running example. The set of ABoxes $\mathbf{B}_0 = \{\mathcal{A}_1, \ldots, \mathcal{A}_5\}$ defined as follows is a $(q_0, \mathcal{T}_0)$-testing base for the class of all well-behaved algorithms:

$$\mathcal{A}_1 = \{\mathsf{Student}(a)\},\ \mathcal{A}_2 = \{\mathsf{takes}(a, b), \mathsf{Course}(b)\},$$
$$\mathcal{A}_3 = \{\mathsf{GradSt}(a)\},\ \mathcal{A}_4 = \{\mathsf{takes}(a, b), \mathsf{GradCo}(b)\},\ \mathcal{A}_5 = \{\mathsf{PhDSt}(a)\}$$

Intuitively, in order to compute a $(q, \mathcal{T})$-repair, we only need to consider the (finitely many) ABoxes in a $(q, \mathcal{T})$-testing base instead of all (infinitely many) possible ABoxes.

Theorem 1. *Let $\mathcal{L}$ be a DL, q a CQ, $\mathcal{T}$ an $\mathcal{L}$-TBox, and $\mathbf{B}$ a $(q, \mathcal{T})$-testing base for $\mathcal{C}^{\mathcal{L}}$. The following property holds for each* $\mathsf{ans} \in \mathcal{C}^{\mathcal{L}}$*: If $\mathcal{R}$ is a $(q, \mathcal{T}, \mathbf{B})$-repair for* ans*, then $\mathcal{R}$ is also a $(q, \mathcal{T})$-repair for* ans*.*

In our running example, clearly, ans_0 only fails to compute certain answers w.r.t. $\mathcal{A}_3$ and $\mathcal{A}_5$. Furthermore, $\mathcal{R}_0$ is a $(q_0, \mathcal{T}_0, \mathbf{B}_0)$-repair for ans_0. But then, Theorem 1 ensures that $\mathcal{R}_0$ is also a $(q_0, \mathcal{T}_0)$-repair. Thus, by simply adding $\mathcal{R}_0$ to $\mathcal{T}_0$, we can guarantee that ans_0 will compute all certain answers, regardless of the data.

4 Computing Repairs

Theorem 1 provides a sufficient condition for the existence of a $(q, \mathcal{T})$-repair for a well-behaved algorithm ans. To apply the theorem, however, we first need to compute a $(q, \mathcal{T})$-testing base $\mathbf{B}$ and then a $(q, \mathcal{T}, \mathbf{B})$-repair $\mathcal{R}$ for ans.

As shown in [15, 16], a $(q, \mathcal{T})$-testing base $\mathbf{B}$ for $\mathcal{C}^{\mathcal{L}}$ can always be computed from a UCQ rewriting u for q and $\mathcal{T}$. UCQ rewritings are guaranteed to exist if $\mathcal{T}$ is expressed in the DL underpinning the OWL 2 QL profile and they may also exist even if $\mathcal{T}$ is expressed in the DLs underpinning the OWL 2 EL profile; this is often the case in many real-world ontologies. Hence, in this paper we will restrict ourselves to TBoxes and queries for which a UCQ rewriting exists.[2]

Given a $(q, \mathcal{T})$-testing base $\mathbf{B}$, however, the existence of a $(q, \mathcal{T}, \mathbf{B})$-repair may also depend on the capabilities of the algorithm under consideration. For instance, in the case of our running example, no $(q_0, \mathcal{T}_0, \mathbf{B}_0)$-repair exists for an algorithm that ignores the TBox and simply matches the query to the data (even though such algorithm is well-behaved).

Our techniques for computing repairs rely on a particular form of *interpolation* [8, 14]. We next make the connection between repairs and interpolation precise, and describe an algorithm for computing interpolants.

4.1 Completeness Repairs and Interpolation

As already discussed, algorithm ans_0 from our running example misses answers w.r.t. $\mathcal{A}_3, \mathcal{A}_5 \in \mathbf{B}_0$, and $\mathcal{R}_0$ from (1) is a $(q_0, \mathcal{T}_0, \mathbf{B}_0)$-repair for ans_0. Furthermore, $\mathbf{B}_0$ can be obtained by "instantiating" queries from the following UCQ rewriting $u = \{q_1, \ldots, q_5\}$ for q_0 and $\mathcal{T}_0$:

$$q_1(x) \leftarrow \mathsf{Student}(x),\ q_2(x) \leftarrow \mathsf{takes}(x, y), \mathsf{Course}(y),$$
$$q_3(x) \leftarrow \mathsf{GradSt}(x),\ \ q_4(x) \leftarrow \mathsf{takes}(x, y), \mathsf{GradCo}(y),\ q_5(x) \leftarrow \mathsf{PhDSt}(x)$$

In particular, $\mathcal{A}_3 = \{\mathsf{GradSt}(a)\}$ is obtained by instantiating q_3. We then say that q_3 is "relevant" to ans_0. Formal definitions of instantiation and relevance are given next.

Definition 4. *Let q be a CQ, B_q the body atoms in q, and π a mapping from all variables of q to individuals. The following ABox is an* instantiation *of q:*

$$\mathcal{A}^q_\pi := \{A(\pi(x)) \mid A(x) \in B_q\} \cup \{R(\pi(x), \pi(y)) \mid R(x, y) \in B_q\}$$

Definition 5. *Let u be a UCQ rewriting for q and an $\mathcal{L}$-TBox $\mathcal{T}$, let* ans *be well-behaved and let $\mathbf{B}$ be a $(q, \mathcal{T})$-testing base. We say that $q_i \in u$ is relevant to* ans *if an instantiation $\mathcal{A}_i$ of q_i exists s.t. $\mathcal{A}_i \in \mathbf{B}$ and $\mathsf{cert}(q, \mathcal{T} \cup \mathcal{A}_i) \not\subseteq \mathsf{ans}(q, \mathcal{T} \cup \mathcal{A}_i)$.*

Note also that q_3 is subsumed by q_0 w.r.t. $\mathcal{T}_0$. We can then identify $\mathcal{R}_0$ as an *interpolant* for the subsumption $q_3 \sqsubseteq_{\mathcal{T}_0} q_0$ since $\mathcal{T}_0 \models \mathcal{R}_0$, $q_3 \sqsubseteq_{\mathcal{R}_0} q_0$ and $\mathcal{R}_0$ only mentions symbols occurring in either q_0 or q_3.

Definition 6. *Let $\mathcal{T}$ be an $\mathcal{L}$-TBox and let q, q' be CQs such that $q' \sqsubseteq_{\mathcal{T}} q$. A TBox $\Im$ expressed in fragment $\mathcal{L}'$ of $\mathcal{L}$ and using only symbols occurring in q or q' is an $\mathcal{L}'$-*interpolant *for $q' \sqsubseteq_{\mathcal{T}} q$ if $\mathcal{T} \models \Im$ and $q' \sqsubseteq_{\Im} q$.*

[2] The notion of a testing base can be extended, and such (extended) testing bases can be computed from *Datalog rewritings*. The study of such extensions is ongoing work.

The following theorem establishes which are the relevant interpolants for computing a repair. Furthermore, if each such interpolants can be expressed in a weak enough language, for which ans is provably complete, we can easily obtain a repair from the union of such interpolants.

Theorem 2. *Let u be a UCQ rewriting for q and an $\mathcal{L}$-TBox $\mathcal{T}$. Let* $\mathsf{ans} \in \mathcal{C}^{\mathcal{L}}$ *be an algorithm that is complete for a fragment $\mathcal{L}'$ of $\mathcal{L}$. Let $q_1, \ldots, q_n$ be those queries in u relevant to* ans*. Finally, for each $1 \leq i \leq n$ let $\Im_i$ be an $\mathcal{L}'$-interpolant for $q_i \sqsubseteq_{\mathcal{T}} q$. Then, $\Im = \bigcup_{1 \leq i \leq n} \Im_i$ is a $(q, \mathcal{T})$-repair for* ans.

4.2 Computing Interpolants

Assumptions First, we restrict ourselves to TBoxes expressed in $\mathcal{ELHI}$. Systems such as REQUIEM can compute a UCQ rewriting w.r.t. an $\mathcal{ELHI}$-TBox, provided that one exists [12].

Second, we observe that most state-of-the-art incomplete systems are based on deductive database and rule technologies. Given $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$and q as input, the ABox $\mathcal{A}$ is deterministically saturated with new assertions using axioms in $\mathcal{T}$, and then q is answered directly w.r.t. the saturated ABox. Furthermore existing systems rarely extend the input ABox with "fresh" individuals and hence cannot handle existential quantification on the right-hand-side of TBox axioms. In fact, many such systems are complete for DLs of the DLP family. Thus, we will consider DLP-$\mathcal{ELHI}$ as our target language for computing interpolants.

In this setting, we also need to restrict the kinds of queries we are considering to guarantee the existence of the relevant interpolants. It is rather straightforward to find $\mathcal{ELHI}$-TBoxes and queries q for which a UCQ rewriting does exist, but the DLP-$\mathcal{ELHI}$ interpolants required by Theorem 2 do not. For example, let $\mathcal{T} = \{A \sqsubseteq \exists R.C\}$ and consider the following queries

$$q(x) \leftarrow R(x, y), C(y) \quad q'(x) \leftarrow A(x)$$

Clearly, $\{q, q'\}$ is a UCQ rewriting for q and $\mathcal{T}$. There is, however, no DLP-$\mathcal{ELHI}$ interpolant for $q' \sqsubseteq_{\mathcal{T}} q$ since such interpolant would need to contain existential quantification on the right-hand-side of GCIs.

Consequently, from now onwards we focus on queries with only distinguished variables. This restriction is not unreasonable in practice since the standard language for querying ontologies on the Web (SPARQL) treats undistinguished variables as if they were distinguished [4].

The Algorithm Algorithm 1 computes a UCQ rewriting u (if one exists) for an $\mathcal{ELHI}$-TBox $\mathcal{T}$ and a query q with no undistinguished variables. Furthermore, for each $q' \in u$, it computes a DLP-$\mathcal{ELHI}$ interpolant for $q' \sqsubseteq_{\mathcal{T}} q$.

Our algorithm extends the rewriting algorithm implemented in the REQUIEM system [12], which first transforms $\mathcal{T}$ and q into a set of clauses (Lines 1, 2 in Algorithm 1) and then uses a resolution calculus to compute u.

Algorithm 1 Extended Resolution-based algorithm

```
Algorithm: extendedUCQRewriting($q, \mathcal{T}$)
Input: $\mathcal{T} \in \mathcal{ELHI}$ and $q$ with no undistinguished vars.
 1: $\mathcal{T}$ := clausify($\mathcal{T}$)
 2: $q$ := clausify($q$)
 3: $\sigma(q) := \emptyset$
 4: for all $\alpha \in \mathcal{T}$ do $\sigma(\alpha) := \{\alpha\}$
 5: $u := \mathcal{T} \cup \{q\}$
 6: repeat
 7:     Pick clauses $C_1, C_2$ from $u$ with $C_1 \neq C_2$
 8:     $C$ := resolve($C_1, C_2$)
 9:     $u := u \cup \{C\}$
10:     $\mathcal{R}_C := \sigma(C_1) \cup \sigma(C_2)$
11:     repeat
12:         Pick clauses $D_1, D_2$ from $\mathcal{R}_C$ with $D_1 \neq D_2$
13:         $\mathcal{R}_C := \mathcal{R}_C \cup \{$resolve($D_1, D_2$)$\}$
14:     until $\mathcal{R}_C$ is saturated
15:     $\sigma(C)$ := prune($\mathcal{R}_C$)
16: until $u$ is saturated
17: $(u,\sigma)$ := ff($u,\sigma$)
18: $(u,\sigma)$ := unfold($u,\sigma$)
19: if $u$ is not a UCQ, return failure
20: return $(u,\sigma)$
```

Interpolants are tracked down during resolution by means of a mapping σ from clauses to sets of clauses. Initially, σ maps q to the empty set (since $q \sqsubseteq_{\emptyset} q$), and each clause from $\mathcal{T}$ to itself (Lines $3, 4$). The rules in the resolution-based calculus from [12] are exhaustively applied in Lines 6–16, and new clauses are generated in Lines 7–9 exactly as in [12]. Then, for each new clause C produced as a resolvent of C_1 and C_2 our algorithm computes its corresponding interpolant $\sigma(C)$. This is done by first saturating $\sigma(C_1) \cup \sigma(C_2)$ (Lines 11–14) and then removing all clauses mentioning a symbol that is neither in C nor in q (Line 15). After u is saturated, the algorithm removes all clauses that contain function symbols (Line 16). Then, at this point, both rewriting and interpolants are given as sets of function-free clauses, so the procedure unfold transforms each $C \in u$ into a datalog rule and "rolls-up" $\sigma(C)$ into a DLP-$\mathcal{ELHI}$ GCI. Finally, the algorithm rejects the input if u is not a UCQ (e.g., a datalog program) and returns both the UCQ rewriting and the interpolant for each query otherwise.

Lemma 1. *If u computed by Algorithm 1 is a UCQ, then u is a UCQ rewriting for q and $\mathcal{T}$. Furthermore, for each $q' \in u$, $\sigma(q')$ is a DLP-$\mathcal{ELHI}$ interpolant for $q' \sqsubseteq_{\mathcal{T}} q$.*

Algorithm 2 computes a repair by considering only those queries and ABoxes for which ans fails. Correctness follows from Theorem 2 and Lemma 1.

Theorem 3. *Algorithm 2 computes a $(q, \mathcal{T})$-repair for an algorithm* ans *that is complete for DLP-$\mathcal{ELHI}$.*

8

Algorithm 2 Computing a Repair

```
Algorithm: computeRepair(ans, $q$, $\mathcal{T}$)
Input: $\mathcal{T}, q$ as in Algorithm 1, ans well-behaved.
 1: $(u,\sigma)$ := extendedUCQRewriting($q, \mathcal{T}$)
 2: $\mathbf{B}$ := computeTestingBase($u$)
 3: Repair := $\emptyset$
 4: for all $\mathcal{A} \in \mathbf{B}$ do
 5:     if cert($q, \mathcal{T} \cup \mathcal{A}$) $\not\subseteq$ ans($q, \mathcal{T} \cup \mathcal{A}$) then
 6:         $q'$ := query in $u$ that generated $\mathcal{A}$
 7:         Repair := Repair $\cup\, \sigma(q')$
 8:     end if
 9: end for
10: return Repair
```

If ans is not complete for DLP-$\mathcal{ELHI}$ (e.g., if it is an RDFS-reasoner), then the TBox $\mathcal{R}$ computed by Algorithm 2 may not be a repair; however, adding $\mathcal{R}$ may still have the desired effect of "mitigating" the reasoner's incompleteness.

5 Evaluation

We have developed a prototype tool based on Algorithms 1 and 2. Our implementation uses REQUIEM [12] and the system described in [16, 15] for computing $(q, \mathcal{T})$-testing bases and relevant queries. Additionally, our tool implements optimisations to eliminate redundancy in the computed repairs.

We have evaluated our techniques first using LUBM's TBox and its 14 test queries (all of which contain only distinguished variables) and then a version of the GALEN TBox together with 4 sample queries for which a UCQ rewriting can be computed using REQUIEM. In each case, we have evaluated the following systems: Sesame 2.3-prl,[3] OWLim,[4] Jena v2.6.3[5] and DLEJena.[6] Our experiments consisted of the following steps:

1. For each TBox $\mathcal{T}$ and test query q we computed a $(q, \mathcal{T})$-testing base and measured the proportion of certain answers that each system is able to compute w.r.t. the ABoxes in the $(q, \mathcal{T})$-testing base (δ_{ini}) [16].
2. For each system and each q for which it was found incomplete (i.e., $\delta_{ini} < 1$), we computed a repair $\mathcal{R}$ (which might be only "partial" for some systems).
3. For each case in Step 2 we have extended $\mathcal{T}$ with the corresponding $\mathcal{R}$, computed a $(q, \mathcal{T} \cup \mathcal{R})$-testing base and obtained a new completeness degree value δ_{fin}.
4. To evaluate the effects that adding TBox axioms had on systems' performance in the LUBM scenario, we used the LUBM performance evaluation

[3] http://www.openrdf.org/
[4] http://www.ontotext.com/owlim/
[5] http://jena.sourceforge.net/
[6] http://lpis.csd.auth.gr/systems/DLEJena/

LUBM													
	Sesame										**OWLim/Jena**		
	Q2	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q12	Q13	Q6	Q8	Q10
$\sharp u_{rel}$	1	4	4	166	34	27	1	166	2	4	1	4	1
$\sharp\mathcal{R}$	0	0	4	1	1	1	0	1	0	4	1	1	1
δ_{ini}	.75	.68	0	.003	.04	.04	0	.001	.25	.2	.99	.98	.99
δ_{fin}	.75	.68	.75	.004	.05	.05	0	.002	.25	.2	1	1	1

GALEN												
	OWLim				**Jena**				**DLEJena**			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
$\sharp u_{rel}$	36	72	72	12	24	48	48	6	36	72	72	6
$\sharp\mathcal{R}$	2	2	2	2	2	2	2	1	2	2	2	1
δ_{ini}	.84	.83	.84	.77	.94	.94	.94	.92	.84	.83	.84	.84
δ_{fin}	1	1	1	1	1	1	1	1	1	1	1	1

Table 1. Experiments for LUBM & GALEN

tool [6] and measured the loading and query answering times using the original and the repaired TBox.

Our results for LUBM are summarised in the upper part of Table 1, where $\sharp u_{rel}$ represents the number of relevant queries in the UCQ rewriting (see Definition 5), and $\sharp\mathcal{R}$ the number of axioms in the computed repair. The only system not included in the table is DLEJena, which was already complete for all queries (w.r.t. the original TBox). OWLim and Jena were found incomplete for three queries and Sesame for 10 of them. As shown in the table, we were able to repair both OWLim and Jena for all queries. This was a reasonable result, since these systems are considered to be complete for DLP-$\mathcal{ELHI}$. Furthermore, all repairs consisted of the same, unique axiom. In contrast, no query could be fully repaired for Sesame. A slight increase in the completeness degree can be observed in 4 queries and a more significant one only in $Q5$. Again, this is unsurprising, since Sesame is essentially an RDFS reasoner. No measurable performance changes were observed for any system after repair, which suggests that completeness can be improved (at least in this scenario) without affecting scalability. Finally, repairs were computed in times ranging between a second and 3 minutes.

Results for GALEN are given in the lower part of the table. As shown in the table, we were able to fully repair OWLim, Jena and DLEJena, and repairs contained at most two axioms in each case (a relatively small number compared to the number of queries in u_{rel}). All repairs could be computed in less than a minute. Finally, Sesame hasn't been included in the table because no improvement could be measured for any query.

6 Conclusions

In this paper, we have studied the problem of *repairing* an incomplete reasoning systems given a query and a TBox. An important feature of our repairs is that they provide *data independent completeness guarantees*: once an incomplete system has been repaired, we can ensure that its output answers will be the same as those of a complete one for the given q and $\mathcal{T}$, regardless of how large and complex the dataset is. Furthermore, our preliminary experiments suggest that repairs can indeed be very small and their effect on systems' performance may even be negligible in some applications. Our results will allow application designers to use highly scalable reasoning systems in their application with the guarantee that they will behave as if they were complete, thus bringing together "the best of both worlds". The extension of our techniques to more expressive DLs and a more extensive evaluation are ongoing work.

Acknowledgments Research supported by project SEALS (FP7-ICT-238975). B. Cuenca Grau is supported by a Royal Society University Research Fellowship.

References

1. Botoeva, E., Calvanese, D., Rodriguez-Muro, M.: Expressive approximations in *DL-Lite* ontologies. In: AIMSA. pp. 21–31 (2010)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
3. G. Meditskos and N. Bassiliades: Combining a DL reasoner and a rule engine for improving entailment-based OWL reasoning. In: Proc. ISWC. pp. 277–292 (2008)
4. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: Proc. of ISWC 2010. pp. 241–256 (2010)
5. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: Proc. of WWW. pp. 48–57 (2003)
6. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics 3(2), 158–182 (2005)
7. Hitzler, P., Vrandecic, D.: Resolution-based approximate reasoning for OWL DL. In: Proc. of ISWC 2005). pp. 383–397 (2005)
8. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. of IJCAI 09. pp. 830–835 (2009)
9. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. Journal of Symbolic Computation 45, 194–228 (2010)
10. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Proc. of ESWC 2006. pp. 125–139 (2006)
11. Pan, J.Z., Thomas, E.: Approximating OWL-DL Ontologies. In: Proc. of AAAI-07. pp. 1434–1439 (2007)
12. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. Journal of Applied Logic 8(2), 186–209 (2010)
13. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal of Data Semantics X, 133–173 (2008)
14. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over dboxes. In: Proc. of IJCAI 09. pp. 923–925 (2009)
15. Stoilos, G., Cuenca Grau, B., Horrocks, I.: Completeness guarantees for incomplete reasoners. In: Proc. of ISWC 2010. LNCS, Springer (2010)
16. Stoilos, G., Cuenca Grau, B., Horrocks, I.: How incomplete is your semantic web reasoner? In: Proc. of AAAI-10. pp. 1431–1436 (2010)

Extracting Finite Sets of Entailments from OWL Ontologies

Samantha Bail, Bijan Parsia, Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{bails,bparsia,sattler@cs.man.ac.uk}

Abstract. The canonical standard description logic reasoning service is classification, that is, the generation of the set of atomic subsumptions which are entailed by some ontology. While this consequence relation is well defined and finite, there is significant variance in the composition of that set. For example, it is common (in tools and in discussion) to exclude some tautologies (e.g., $A \sqsubseteq \top$, $A \sqsubseteq A$). While for many purposes such divergences are harmless, there are many for which precision about what appears in the classification is essential, for example, estimating differences in logical content. In this paper, we propose definitions for different types of finite entailment sets of an OWL ontology based on the transitive closure and transitive reduction of its asserted and inferred class graphs. The purpose of this work is to introduce a flexible and extensible specification for selecting a particular set of entailments, with the aim of ensuring the correctness and replicability of OWL-based applications.

1 Introduction and Motivation

The Web Ontology Language OWL 2 DL is based on the expressive description logic $\mathcal{SROIQ}$ [6]. It is designed to 'facilitate ontology development and sharing via the Web',[1] with OWL development tools aiming at users with little or no knowledge in description logics. *Entailment* is regarded as the 'key inference' of the Semantic Web [12], and while the entailment relation $\models$ is well defined for OWL ontologies [7], misleading nomenclature in ontology tools and anecdotal evidence show that there exist common misconceptions about entailments: first, it is often assumed that the set of entailments of an ontology is finite, and it is possible to extract the set of *all* entailments of an ontology. Second, it is assumed that only non-trivial information is contained in the set of entailments, and tautologies such as $A \sqsubseteq A$ are not entailments. Third, the term entailments is used interchangeably with *inferences*, and the information that is asserted in the ontology is often not considered to be an entailment itself. While the problem of reasoning with and extracting meaningful entailments from inconsistent ontologies has been previously discussed in the literature [8,4], we focus on consistent ontologies for the purpose of this paper, and limit the definitions and examples to atomic subsumptions and equivalences.

[1] http://w3.org/TR/owl2-overview

The ontology editor Protégé 4[2] for instance comes with a 'selected entailments' tab which shows a list of atomic SubClassOf, SubPropertyOf and Type (class assertion) axioms. The tool also offers the option to 'Save inferred axioms as ontology' which saves asserted and inferred axioms as a new OWL ontology. Similarly, Top Braid Composer[3] offers to 'Save [the] inference graph' as a new file. In all cases, it is not clear how the entailments are generated, what the selection criteria is for entailments (inferences), and how the user can modify the ontology to affect these entailments. For example, given an ontology containing only subsumptions and equivalences between named classes, Protégé 4 exports all direct and strict subsumptions between the classes, but offers no options to export the indirect or non-strict subsumptions explicitly. Top Braid Composer, however, does not include (direct or indirect) atomic subsumptions at all in the exported 'inference graph'.

Ontologies that are available on the web may be published as 'compiled' versions, which include the ontology and its entailments of some description. The OWL version of the National Cancer Institute (NCI) Thesaurus, for example, 'includes inferred relationships'.[4] There is, however, no definition of what is regarded as an inferred relationship, how these relationships are determined, and what the selection criteria is. This may leave users wondering what kinds of information they are dealing with, and what implications this has for their understanding of the ontology.

Analytical applications that are based on justifications (minimal subsets of the ontology that are sufficient for the entailment to hold) extract the *entailed atomic subsumptions* of an ontology and compute the justifications therefore [3,5]. Again, transparency of the entailment extraction process is vital for these applications in order to ensure correct and meaningful results. For instance, the number of entailments, as well as the number and properties of their justifications, can be skewed by including or excluding subsumptions caused by unsatisfiable classes, not distinguishing between direct and indirect subsumptions, whether unsatisfiable classes are treated as a subclass or equivalent to bottom, and similar selection criteria. Furthermore, imported ontologies add to the complexity of the problem: computing entailments from the imports closure of an ontology also considers the entailments and justifications of imported ontologies. This may distort the actual number and types of entailments, while also adding a computational overhead to entailment extraction procedures.

The above examples demonstrate how the notion of entailments is widely used in OWL applications, however without a clear understanding of how the entailment relation relates to the set of axioms that is obtained from an ontology. In order to ensure the correctness and replicability of data based on the entailments of an ontology, it is necessary to explicitly specify which (finite) subset of the set of all entailments should be extracted from the ontology. In this paper, we discuss the different aspects of extracting entailments of OWL ontologies and

[2] http://protege.stanford.edu
[3] http://topquadrant.com/products/TB_Composer.html
[4] http://evs.nci.nih.gov/ftp1/NCI_Thesaurus/ReadMe.txt

provide definitions for practical entailment sets. We propose ways of dealing with imported entailments based on the justifications for the entailment. Rather than providing an exhaustive definition for all possible entailment sets of a $\mathcal{SROIQ}$ ontology, the focus of this paper is to encourage clarity when using the term 'entailments' in the context of OWL ontology applications.

2 Applications

2.1 Inferred Ontology Generation in the OWL API

The OWL API[5] provides the convenience class `InferredOntologyGenerator`, which allows users to 'fill' a new ontology with the desired type of entailments, such as inferred atomic SubClass axioms and ClassAssertions. By default, this method only retrieves the *direct* superclasses of a named class when using `InferredSubClassOfAxiomGenerator`. For each class that does not have any direct superclasses other than OWL:Thing, the reasoner returns a node labelled OWL:Thing. While this method provides a common basis for computing the *inferred ontology*, it does not offer any flexibility for the user to specify which relationships should be included in the output. We propose the implementation of more specific and flexible entailment generation methods in the OWL API as an addition to or extension of existing methods, in order to allow users to conveniently extract clearly defined finite entailment sets from an OWL ontology.

2.2 Presenting Entailments to End-Users

The OWL ontology editor Protégé 4 provides a view of 'selected entailments' of the ontology. As the editor offers no further explanation to how these entailments were extracted in the classification process, this view does not support understanding of the ontology. It may even seem surprising to the end-user that some trivial axioms, such as $\mathcal{A} \sqsubseteq \mathsf{OWL{:}Thing}$, are displayed in the panel while others are missing. A more detailed and modifiable view could support users in exploring the class hierarchy when attempting to understand entailment relations in the ontology.

2.3 Explanation of Entailments

Explanation of entailments for the purpose of debugging a description logic ontology has been the focus of research since the early applications of description logics for modelling domain knowledge [9,10,11]. Most OWL ontology editors provide explanation facilities presenting the part of the ontology which causes the entailment to hold. It may be argued that, from a user perspective, a crucial part of understanding why an entailment holds in an ontology is to have a clear understanding of the notion of entailments, while also being able to control the method of extracting the entailment.

[5] http://owlapi.sourceforge.net

2.4 Metrics

Analytical applications that consider the number and type of entailments in order to infer semantic ontology metrics benefit from clearly defined entailment sets in two ways: first, the basis of the measurements, i.e. *what exactly* is measured, is well defined and transparent, therefore ensuring consistent measurements which are independent from a particular implementation or individual modifications of the results provided by the OWL API. Second, greater flexibility allows to extract entailments that are fit for a specific purpose. For instance, when describing the inferential power of an ontology, it is not necessary to consider the asserted entailments as they hold no information value. On the other hand, to explore the justificatory structure of an OWL ontology, we need to consider that there may be non-obvious and possibly complex reasons for entailments that are asserted in the ontology; this makes it necessary to compute the justifications for both inferred and asserted entailments in order to capture these 'hidden' justifications.

3 Extracting and Counting Entailments

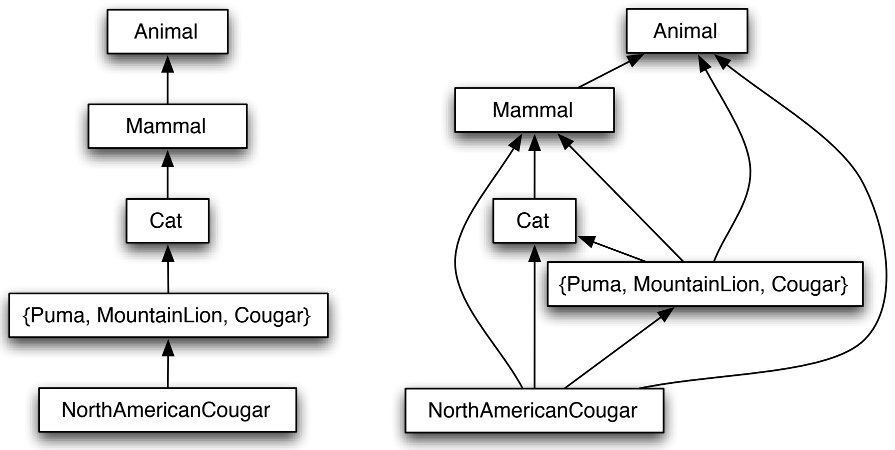
In this section we present different criteria for defining the set of entailments of an OWL ontology. We provide four definitions for finite entailment sets based on the class graph of the ontology, which we then illustrate with examples. In each case, we expect the input to be an OWL 2 DL ontology $\mathcal{O}$, with the output being a set of OWL 2 DL axioms α. Please note that for the purpose of demonstrating our approach, we only focus on *atomic* entailments, i.e. relations between named atomic classes in the ontology.

3.1 Entailments of Description Logic Ontologies

In the remainder of this paper the letters A, B denote class names, C, D (possibly complex) concepts, a an individual, $\mathcal{O} = (\mathcal{T},\mathcal{A})$ a description logic ontology which is the union of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, α an axiom in $\mathcal{O}$, and $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation of an ontology $\mathcal{O}$. The notations for $\top$ and OWL:Thing, and $\bot$ and OWL:Nothing are used interchangeably.

The term *finding entailments* of a DL ontology summarises different reasoning tasks, such as the *subsumption problem*, *equivalence* and *satisfiability* checking with respect to a TBox $\mathcal{T}$, and *instance checking* with respect to an ABox $\mathcal{A}$.

Entailment relations in $\mathcal{SROIQ}$ are defined based on the formal semantics given by an interpretation $\mathcal{I}$ [2]. An ontology $\mathcal{O} = (\mathcal{T},\mathcal{A})$ entails that a (possibly complex) concept C is subsumed by a concept D, written as $\mathcal{O} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. Similarly, $\mathcal{O}$ entails that C is equivalent to D if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. A concept C is entailed to be *unsatisfiable*, i.e. $\mathcal{O} \models C \equiv \bot$ (commonly expressed as $\mathcal{O} \sqsubseteq C \equiv \bot$) if $C^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{O}$. Regarding instance checking for the ABox $\mathcal{A}$, $\mathcal{O}$ entails that an individual



(a) Asserted Class Graph (b) Inferred Class Graph

Fig. 1. Asserted and Inferred Class Graphs

a is an instance of a concept C ($\mathcal{O} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. The set of entailments of an ontology is therefore the set of all axioms α such that $\mathcal{O} \models \alpha$.

3.2 Inferred and Asserted Class Graphs

The asserted class graph of an ontology $\mathcal{O}$ is a labelled directed acyclic graph $G = (V, E, L)$ with nodes labelled with (a non-empty set of) class names, including $\top$ and $\bot$, from the signature of $\mathcal{O}$. The graph is initialised by creating a node u for each class name in the signature of $\mathcal{O}$, with the class name being in the label $L(u)$ of the node. An edge (u, v) is added if it is asserted in $\mathcal{O}$ that $A \sqsubseteq B$ for some $A \in L(u)$, $B \in L(v)$, where A and B are class names, $\top$, or $\bot$. For any two nodes u, v in the graph with $L(u) = \{A\}$, $L(v) = \{B\}$, the nodes are collapsed into a single node w if the ontology contains the two subsumption axioms $A \sqsubseteq B$ and $B \sqsubseteq A$, or the equivalence class axiom $A \equiv B$. This leads to existing edges (u, x), (v, y) for some node x, y in the graph, being replaced by the corresponding edges (w, x), (w, y).

The inferred class graph $G' = (V', E', L')$ of the ontology contains an edge (u, v) if $\mathcal{O} \models A \sqsubseteq B$ for some $A \in L(u)$, $B \in L(v)$. A class name A is in the label $L(u)$ of a node u in the inferred class graph if $\mathcal{O} \models A \equiv B_i$ for all B_i in $L(u)$.

The asserted and inferred class graphs in Figure 1 are based on the following toy ontology:

Example 1 (Toy ontology)

$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Cougar}$	$\mathsf{Mammal} \sqsubseteq \mathsf{Animal}$
$\mathsf{Cougar} \equiv \mathsf{MountainLion}$	$\mathsf{Puma} \equiv \mathsf{Cougar}$
$\mathsf{Puma} \sqsubseteq \mathsf{Cat}$	$\mathsf{Cat} \sqsubseteq \mathsf{Mammal}$

3.3 Generating Axioms From the Transitive Reduction and Transitive Closure

The inferred and asserted class graphs of an ontology are uniquely defined. Retrieving these class graphs may be sufficient for counting entailments, as every subsumption relationship between nodes is represented by a single edge, and the arity of a node label (i.e. the number of distinct class names in the node) is equal to the number of equivalent classes. OWL applications however generally present the asserted and inferred class hierarchy as sets of OWL axioms, which need to be generated from the relationships in the respective class graphs.

Generating axioms from the transitive closure of the inferred class graph is straightforward: a SubClassOf axiom is created for each pair of class names in the label of the sub- and superclass node respectively, and an EquivalentClasses axiom for each pair of class names in the label of a node. Example 2 demonstrates how this method can quickly lead to a large set of entailments. In some situations however it is sufficient and more economical to use a subset of these relations based on the transitive reduction of the graph.

The transitive reduction of a directed acyclic graph is a canonical representation for the paths in the graph [1]. The main challenge here is: how can we express, in one single axiom, an edge between nodes which are labelled with multiple class names? Furthermore, if a node is labelled with several equivalent classes, we would like the number of axioms generated from this node to reflect how many equivalent classes the label contains; therefore, multiple nodes that contain different numbers of class names in their labels cannot be represented by a single EquivalentClasses axiom.

For the purpose of expressing subsumptions in the transitive reduction, a function $Rep(u)$ is introduced which selects a single class name from the label of a node u to act as a representative for the node. This function is intended to be user-defined and may retrieve a randomly selected element, the first element in a lexicographical ordering, or even a freshly generated class name (such as the concatenation of the class names in the node), to name a few examples.

In order to generate EquivalentClasses axioms from a node with arity n, where $n > 2$, we apply a function $Pairwise(u)$ to the node label. This introduces an ordering $<$ on the class names in the label of the node (such as a lexicographical order) and returns a set of pairs of class names (A_i, A_{i+1}) where $A_i < A_{i+1}$. While OWL 2 allows EquivalentClasses axioms with an arity greater than two, we choose to express equivalences in binary axioms, which corresponds to the description logic notation of $A_i \equiv A_{i+1}$.

3.4 Top, Bottom, and Tautologies

While an unsatisfiable class is generally referred to as being a *subclass of Bottom*, the class is in fact equivalent to the bottom node OWL:Nothing, as discussed above. We consider this in our definitions and treat an unsatisfiable named class not as a subsumption, but as an atomic equivalence. Likewise, a *universal* class, i.e. a class that is equivalent to OWL:Thing, is not treated as *superclasses of Top*, but as an equivalent class.

Furthermore, tautologies such as $A \sqsubseteq \top$, $\bot \sqsubseteq A$ and $A \sqsubseteq A$ for all named classes A are not included in the entailment set, as they do not hold any information value.

3.5 Different Types of Entailment Sets

We define the set of inferred atomic entailments of an ontology $\mathcal{O}$ as the union of the inferred atomic subsumptions *Sub* and the inferred atomic equivalences *Equiv* for the asserted and inferred class graphs $G = (V, E, L)$ and $G' = (V', E', L')$ respectively. The following definitions are ordered by two aspects: whether they are based on the transitive closure $Tc(E')$ or transitive reduction $Tr(E')$ of the inferred class graph, and whether they include (A^+) or exclude (A^-) asserted subsumptions and equivalences respectively.

Transitive closure, inferred, including asserted

$$\begin{aligned}\mathsf{SubTcA}^+(\mathcal{O}) :=& \{A \sqsubseteq B \mid \textit{ there is } (u,v) \in E', A \in L'(u), B \in L'(v),\\ & A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTcA}^+(\mathcal{O}) :=& \{A \equiv B \mid \textit{ there is } u \in V', A, B \in L'(u), A \neq B\}\end{aligned}$$

Transitive closure, inferred, not including asserted

$$\begin{aligned}\mathsf{SubTcA}^-(\mathcal{O}) :=& \{A \sqsubseteq B \mid \textit{ there is } (u,v) \in E', A \in L'(u), B \in L'(v)\\ & (u,v) \notin E, A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTcA}^-(\mathcal{O}) :=& \{A \equiv B \mid \textit{ there is } u \in V', A, B \in L'(u), A \neq B,\\ & \textit{there is no } v \in V \textit{ s.t. } A, B \in L(v)\}\end{aligned}$$

Transitive reduction, inferred, including asserted

$$\begin{aligned}\mathsf{SubTrA}^+(\mathcal{O}) :=& \{A \sqsubseteq B \mid \textit{ there is } (u,v) \in TR(E'),\\ & A = \mathit{Rep}(u), B = \mathit{Rep}(v), A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTrA}^+(\mathcal{O}) :=& \{A \equiv B \mid \textit{ there is } u \in V', (A, B) \in \mathit{Pairwise}(u), A \neq B\}\end{aligned}$$

Transitive reduction, inferred, not including asserted

$$\begin{aligned}\mathsf{SubTrA}^{-}(\mathcal{O}) :=& \{A \sqsubseteq B \mid \textit{ there is } (u,v) \in TR(E'),\\ & A = \mathit{Rep}(u), B = \mathit{Rep}(v), (u,v) \notin E, A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTrA}^{-}(\mathcal{O}) :=& \{A \equiv B \mid \textit{ there is } u \in V', (A,B) \in \mathit{Pairwise}(u), A \neq B,\\ & \textit{there is no } v \in V \textit{ s.t. } A, B \in L(v)\}\end{aligned}$$

3.6 Examples

The properties of different entailment sets are demonstrated using the above toy ontology as an example.

Example 2 (Transitive closure, including asserted, 18 axioms)

$\mathsf{Cougar} \equiv \mathsf{MountainLion}$	$\mathsf{Cougar} \equiv \mathsf{Puma}$
$\mathsf{MountainLion} \equiv \mathsf{Puma}$	$\mathsf{Puma} \sqsubseteq \mathsf{Cat}$
$\mathsf{Puma} \sqsubseteq \mathsf{Mammal}$	$\mathsf{Puma} \sqsubseteq \mathsf{Animal}$
$\mathsf{MountainLion} \sqsubseteq \mathsf{Cat}$	$\mathsf{MountainLion} \sqsubseteq \mathsf{Mammal}$
$\mathsf{MountainLion} \sqsubseteq \mathsf{Animal}$	$\mathsf{Cougar} \sqsubseteq \mathsf{Cat}$
$\mathsf{Cougar} \sqsubseteq \mathsf{Mammal}$	$\mathsf{Cougar} \sqsubseteq \mathsf{Animal}$
$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Puma}$	$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{MountainLion}$
$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Cougar}$	$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Cat}$
$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Mammal}$	$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Animal}$

The transitive closure, including asserted, makes explicit the relationships between every single class in the ontology. It is the largest finite entailment set to be extracted from the class graph. The alternative variant of this set excluding asserted entailments simply discards the axioms that occur in the original ontology, yielding a set of 12 axioms.

Example 3 (Transitive reduction, including asserted, 6 axioms)

$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Cougar}$	$\mathsf{Cougar} \equiv \mathsf{MountainLion}$
$\mathsf{MountainLion} \equiv \mathsf{Puma}$	$\mathsf{Puma} \sqsubseteq \mathsf{Cat}$
$\mathsf{Cat} \sqsubseteq \mathsf{Mammal}$	$\mathsf{Mammal} \sqsubseteq \mathsf{Animal}$

The entailment set based on the transitive reduction of the class graph uses representative elements from each node to produce a minimal representation of the class hierarchy. In this example, the function selects the class names that are asserted to be in SubClassOf relationships in the ontology, otherwise it selects a random class name from the node. The function *Pairwise*(u) applies a lexicographical ordering on the class names in each node, as described above.

3.7 Counting Entailments

Having translated the class graph into a set of OWL axioms based on the above definitions, the number of entailments can be computed in an unambiguous way. By choosing a representative class name for each node in the transitive reduction, the number of entailed atomic subsumptions is equal to the number of edges in the graph, i.e. one edge in the graph is represented by one axiom.

For the transitive closure of the class graph, the number of entailed subsumption axioms is the sum of all $n(u) * n(v)$ for each edge (u, v) in the graph, with $n(u)$ the number of class names in a node u. The number of entailed binary equivalence class axioms is $n(u) * (n(u) - 1)/2$ for each node u in the graph.

3.8 Implementation

We have implemented the entailment extractor methods using the OWL API.[6] The code is intended to be used with any OWL reasoner that is compatible with the current version of the OWL API. Preliminary tests with large ontologies such as the NCI Thesaurus show that all types of entailment sets can be extracted in practical time.

4 Dealing with Imports

Another issue that needs to be dealt with when extracting and counting entailments from an ontology is its import structure. An OWL ontology $\mathcal{O}$ (the 'root' ontology) that imports another OWL ontology $\mathcal{O}$' can have different kinds of entailments: those that hold in $\mathcal{O} \setminus \mathcal{O}$' (*native* entailments), those that are entirely from the imported ontology, i.e. they hold in $\mathcal{O}$' $\setminus$ $\mathcal{O}$ (*imported* entailments), and those that hold in $\mathcal{O} \cup \mathcal{O}$' but not in $\mathcal{O} \setminus \mathcal{O}$' (*mixed* entailments). When performing analytical tasks on the root ontology such as analysing its inferential power, it may be considered misleading to include the number of imported entailments. Furthermore, if the imported ontology itself imports another ontology (and so on), we will almost certainly obtain data that is not relevant to our original root ontology. While this may not be problematic for some tasks, the origin of entailments needs to be at least made obvious in a way such that the user can make their own judgements on how to handle them.

4.1 Classification of Imported Entailments

We propose a classification of these three types of entailments in an ontology which is based on the notion of justifications for an entailment. A justification is a minimal subset of the ontology that is sufficient for the entailment to hold [10]. The 'origin' of an entailment given an ontology imports structure is determined by the set of its justifications.

[6] The source code is available for download and modification at http://code.google.com/p/owl-entailment-extractor.

Type 1: Native entailments We may want to restrict the entailment extraction to the root ontology and discard all entailments that originate partly or entirely from the imported ontologies. In this case, the class graph construction and reasoning process is limited to the root ontology axioms only. Computing justifications is not necessary for this type of entailments.

Type 2: Imported entailments While the entailments that originate purely from the imported ontology may not be relevant to application, an analysis of the type and numbers provides information about the computational overhead they may cause when not excluding them from the entailment set. Type 2 entailments are extracted by computing the inferred class graph for axioms that are contained in the imported ontology only. Computing justifications is not necessary for this type of entailments.

Type 3: Mixed entailments In this type, we gather all entailments that are considered 'mixed' for at least one of the following reasons: first, an entailment that has at least one justification which contains axioms from both $\mathcal{O}$ and $\mathcal{O}$' is considered mixed. Second, an entailment that has some justification that comprises axioms from $\mathcal{O}$, and some justification that comprises axioms from $\mathcal{O}$'.[7] Type 3 entailments are computed by extracting all entailments from the union of the imports closure of the root ontology, then sequentially generating justifications for these entailments. If the set of justifications contains axioms from both the root and the import ontologies, the entailment is marked as 'mixed' and no further justifications need to be found.

5 Conclusions and Future Work

Due to the ambiguous use of the term 'entailments' in the OWL community, it is necessary to explicitly specify the selection criteria for entailments in both analytical and user-oriented applications. The methods for extracting entailments from OWL ontologies currently provided by the OWL API and ontology development tools provide little flexibility and do not support understanding of the entailment relationships in the ontology. We have presented well-founded and extensible definitions for different types of entailment sets of an OWL ontology, based on its class graph. Depending on the purpose, users can extract entailments from the transitive reduction or the transitive closure of the ontology, and decide whether the asserted entailments should be included. We have also introduced different ways of dealing with entailments that are partly or entirely caused by imported ontologies. The proposed methods offer flexibility and transparency when handling entailments, which may support ontology understanding as well as clarify analytical tasks.

[7] While this distinction is not relevant to the classification of entailments, it does matter in the context of analysing the structure of justifications in an ontology.

Thus far, we have discussed definitions and examples for entailed subsumptions between atomic classes. In order to capture the wide range of entailments from an expressive description logic such as $\mathcal{SROIQ}$ and to provide extensive information about an ontology, these definitions can be extended in two directions: first, to cover the expressivity of OWL 2 DL ontologies beyond subsumptions between named classes, such as class assertions, object property hierarchies and data property hierarchies. Second, we may also want to capture non-atomic entailments, such as literals (disjointness of classes) and subsumptions and equivalences between complex class expressions (e.g. existential and universal restrictions on atomic class names). In the case of complex class expressions, it will be necessary to identify which complex entailments are of interest to users, depending on the needs of a particular application.

References

1. A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.
2. F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, 2003.
3. S. Bail, B. Parsia, and U. Sattler. The justificatory structure of OWL ontologies. In *Proc. of OWLED-10*, 2010.
4. P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. volume 3729 of *Lecture Notes in Computer Science*, pages 353–367. Springer, 2005.
5. M. Horridge, B. Parsia, and U. Sattler. The state of bio-ontologies. In *To be published in Proc. of ISMB-11*, 2011.
6. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proc. of KR-06*, pages 57–67, 2006.
7. I. Horrocks and P. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.
8. Z. Huang, F. Van Harmelen, and A. Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI-05*, volume 19, page 454. Citeseer, 2005.
9. D. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proc. of IJCAI-95*, volume 14, pages 816–821, 1995.
10. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW-05*, pages 633–640, 2005.
11. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI-03*, pages 355–362, 2003.
12. E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53, 2007.

Concept Model Semantics for DL Preferential Reasoning

Katarina Britz[1,2], Thomas Meyer[1,2], and Ivan Varzinczak[1,2]

[1] CSIR Meraka Institute, Pretoria, South Africa
[2] University of KwaZulu-Natal, Durban, South Africa
{arina.britz,tommie.meyer,ivan.varzinczak}@meraka.org.za

Abstract. The preferential and rational consequence relations first studied by Lehmann and colleagues play a central role in non-monotonic reasoning, not least because they provide the foundation for the determination of the important notion of rational closure. Although they can be applied directly to a large variety of logics, these constructions suffer from the limitation that they are largely propositional in nature. One of the main obstacles in moving beyond the propositional case has been the lack of a formal semantics which appropriately generalizes the preferential and ranked models of Lehmann et al. In this paper we propose a semantics to fill that gap for description logics, an important class of decidable fragments of first-order logic. Our semantics replaces the propositional valuations used in the models of Lehmann et al. with structures we refer to as *concept models*. We prove representation results for the description logic $\mathcal{ALC}$ for both preferential and rational consequence relations. We argue that our semantics paves the way for extending preferential and rational consequence, and therefore also rational closure, to a whole class of logics that have a semantics defined in terms of first-order relational structures.

1 Introduction

There has by now been quite a substantial number of attempts to incorporate defeasible reasoning in logics other than propositional logic. One such endeavor, and the broad focus of this paper, has been to extend the influential version of preferential reasoning first studied by Lehmann et al. [7, 9] to logics beyond the propositional. A stumbling block to this end has been that research on preferential reasoning has really only reached maturity in a propositional context, whereas many logics of interest have more structure. A generally accepted semantics for first-order preferential reasoning, with corresponding syntactic proof system or characterization, does not yet exist. The first tentative exploration of preferential predicate logics by Lehmann et al. didn't fly (pun intended), primarily because propositional logic was sufficiently expressive for the non-monotonic reasoning community at the time, and first-order logic introduced too much complexity [8]. But this changed with the surge of interest in description logics as knowledge representation formalism. Description logics (DLs) [1] are decidable fragments of first-order logic, and are ideal candidates for the kind of extension to preferential reasoning we have in mind: the notion of subsumption present in all DLs is a natural candidate for defeasibility, while at the same time, the restricted expressivity of DLs ensures that attempts to introduce preferential reasoning are not hampered by the complexity of full first-order logic. The aim of this paper is to extend the work of Lehmann et al. [7, 9] beyond propositional logic without moving to full first-order logic. We restrict our attention to the description logic $\mathcal{ALC}$ here, but the results are broadly applicable to other DLs, as well as other similarly structured logics such as logics of action and logics of knowledge and belief.

The central question answered in this paper is how the existing semantics for both preferential and rational propositional non-monotonic consequence relations should be generalized to languages with more structure, and in particular, to $\mathcal{ALC}$. Specifically, what is the meaning of a preferential (or rational) subsumption statement $C \precsim D$ — what properties should it have, and what is its corresponding formal semantics? The main results of this paper, and our answers to these questions, are the two representation results presented in Theorems 3 and 4, respectively. Key to the establishment of these results are the notions of a *concept model*, which gives a reading of the meaning of concepts suitable for our purposes, and a *DL preferential model*, giving meaning to non-monotonic subsumption statements. The latter generalizes the notion of a propositional preferential model in terms of concept models.

The rest of the paper is structured as follows. In Section 2 we give a brief account of the work on preferential and rational subsumption for the propositional case as developed by Lehmann and colleagues. Section 3 is the heart of the paper in which we define the semantics for both preferential and rational subsumption for $\mathcal{ALC}$ and prove representation results for both. Importantly, the representation results provided here are with respect to the corresponding *propositional* properties. From this we conclude that the semantics we present here forms the foundation of a semantics for preferential and rational consequence for a whole class of DLs and related logics and provides a natural and intuitive semantic framework on which to base such work. In Section 4 we use the fundamental results of the previous section to show that the notions of propositional preferential entailment and rational closure can be 'lifted' to the case for DLs, specifically $\mathcal{ALC}$. In Section 5 we discuss related results. We conclude with Section 6 in which we also discuss future work.

We assume that the reader is familiar with description logics. For more details on description logics in general, and the description logic $\mathcal{ALC}$ in particular, the reader is referred to the DL handbook [1].

2 Propositional Preferential Consequence

In this section we give a brief introduction to propositional preferential and rational consequence, as initially defined by Kraus et al. [7]. A propositional defeasible consequence relation $\mid\!\sim$ is defined as a binary relation on formulas

$\alpha, \beta, \gamma, \ldots$ of an underlying (possibly infinitely generated) propositional logic equipped with a standard propositional entailment relation $\models$ [7]. $\mid\!\sim$ is said to be *preferential* if it satisfies the following set of properties:

$$\text{(Ref)}\ \alpha \mid\!\sim \alpha \qquad \text{(LLE)}\ \frac{\alpha \equiv \beta, \alpha \mid\!\sim \gamma}{\beta \mid\!\sim \gamma} \qquad \text{(And)}\ \frac{\alpha \mid\!\sim \beta, \alpha \mid\!\sim \gamma}{\alpha \mid\!\sim \beta \wedge \gamma}$$

$$\text{(RW)}\ \frac{\alpha \mid\!\sim \beta, \beta \models \gamma}{\alpha \mid\!\sim \gamma} \qquad \text{(Or)}\ \frac{\alpha \mid\!\sim \gamma, \beta \mid\!\sim \gamma}{\alpha \vee \beta \mid\!\sim \gamma} \qquad \text{(CM)}\ \frac{\alpha \mid\!\sim \beta, \alpha \mid\!\sim \gamma}{\alpha \wedge \beta \mid\!\sim \gamma}$$

The semantics of (propositional) preferential consequence relations is in terms of *preferential models*; these are partially ordered structures with states labeled by propositional valuations. We shall make this terminology more precise in Section 3, but it essentially allows for a partial order on states, with states lower down in the order being more preferred than those higher up. Given a preferential model $\mathscr{P}$, a pair $\alpha \mid\!\sim \beta$ is in the consequence relation defined by $\mathscr{P}$ iff the minimal states (according to the partial order) of all those states labeled by valuations that are propositional models of α, are also labeled by propositional models of β. The representation theorem for preferential consequence relations then states:

Theorem 1 (Kraus et al. [7]). *A defeasible consequence relation is a preferential consequence relation iff it is defined by some preferential model.*

If, in addition to the properties of preferential consequence, $\mid\!\sim$ also satisfies the following Rational Monotony property, it is said to be a *rational* consequence relation:

$$\text{(RM)}\ \frac{\alpha \mid\!\sim \beta, \alpha \not\mid\!\sim \neg \gamma}{\alpha \wedge \gamma \mid\!\sim \beta}$$

The semantics of rational consequence relations is in terms of *ranked* preferential models, i.e., preferential models in which the preference order is *modular*:

Definition 1. *Given a set S, $\prec \subseteq S \times S$ is modular i ff$\prec$ is a partial order on S, and there is a ranking function $rk : S \mapsto \mathbb{N}$ s.t. for every $s, s' \in S$, $s \prec s'$ iff $rk(s) < rk(s')$.*

The representation theorem for rational consequence relations then states:

Theorem 2 (Lehmann and Magidor [9]). *A defeasible consequence relation is a rational consequence relation iff it is defined by some ranked model.*

3 Semantics for DL Preferential Consequence

It has been argued elsewhere that description logics are ideal candidates for the extension of propositional preferential consequence since the notion of subsumption in DLs lends itself naturally to defeasibility [3, 6, 4]. The basic idea is to reinterpret defeasible consequence of the form $\alpha \mid\!\sim \beta$ as *defeasible subsumption* of the form $C \precsim D$, where C and D are DL concepts, and classical entailment $\models$ as DL subsumption $\sqsubseteq$. The properties of preferential consequence from Section 2 are then immediately applicable.

Definition 2. *A subsumption relation $\precsim \subseteq \mathcal{L} \times \mathcal{L}$ is a preferential subsumption relation iff it satisfies the properties (Ref), (LLE), (And), (RW), (Or), and (CM), with propositional entailment replaced by classical DL subsumption. $\precsim$ is a rational subsumption relation iff in addition to being a preferential subsumption relation, it also satisfies the property (RM).*

However, up until now it has not been clear how to best generalize the propositional semantics for the DL case. Since DLs have a standard first-order semantics, the obvious generalization from a technical perspective is to replace the propositional valuations in preferential models with first-order interpretations. Intuitively, this also turns out to be a natural generalization of the propositional setting, with the notion of normal first-order interpretation characterizing a given concept replacing the propositional notion of normal worlds satisfying a given proposition. Formally, our semantics is based on the notion of a *concept model*, which is analogous to that of a Kripke model in modal logic [2]:

Definition 3 (Concept Model). *A concept model is a tuple $\mathscr{M} = \langle W, R, V \rangle$ where W is a set of possible worlds, $R = \langle R_1, \ldots, R_n \rangle$, where each $R_i \subseteq W \times W$, $1 \leq i \leq |\mathsf{N}_{\mathscr{R}}|$, and $V \colon W \mapsto 2^{\mathsf{N}_{\mathscr{C}}}$ is a valuation function.*

Observe that the valuation function V can be viewed as a propositional valuation with propositional atoms replaced by concept names. From the definition of satisfaction in a concept model below it is then clear that, within the context of a concept model, a world occurring in that concept model is a proper generalization of a propositional valuation.

Definition 4 (Satisfaction). *Given $\mathscr{M} = \langle W, R, V \rangle$ and $w \in W$:*

- $\mathscr{M}, w \Vdash \top$;
- $\mathscr{M}, w \Vdash A$ *iff* $A \in V(w)$;
- $\mathscr{M}, w \Vdash C \sqcap D$ *iff* $\mathscr{M}, w \Vdash C$ *and* $\mathscr{M}, w \Vdash D$;
- $\mathscr{M}, w \Vdash \neg C$ *iff* $\mathscr{M}, w \nVdash C$;
- $\mathscr{M}, w \Vdash \exists r_i.C$ *iff there is* $w' \in W$ *s.t.* $(w, w') \in R_i$ *and* $\mathscr{M}, w' \Vdash C$.

Let $\mathscr{U}$ denote the set of all pairs $(\mathscr{M}, w)$ where $\mathscr{M} = \langle W, R, V \rangle$ is a concept model and $w \in W$.

Worlds are, loosely speaking, interpreted DL objects. And while this correspondence holds technically (from the correspondence between $\mathcal{ALC}$ and multimodal logic K [13]), a possible worlds reading of the meaning of a concept is also more intuitive in the current context, since this leads to a preference order on rich first-order structures, rather than on interpreted objects. This is made precise below.

Let S be a set, the elements of which are called *states*. Let $\ell : S \mapsto \mathscr{U}$ be a *labeling function* mapping every state to a pair $(\mathscr{M}, w)$ where $\mathscr{M} = \langle W, R, V\rangle$ is a concept model s.t. $w \in W$. Let $\prec$ be a binary relation on S. Given $C \in \mathcal{L}$, we say that $s \in S$ *satisfies* C (written $s \models\!\!\equiv C$) i ff$\ell(s) \Vdash C$, i.e., $\mathscr{M}, w \Vdash C$. We define $\widehat{C} = \{s \in S \mid s \models\!\!\equiv C\}$. $\widehat{C}$ is *smooth* iff each $s \in \widehat{C}$ is either $\prec$-minimal in $\widehat{C}$, or there is $s' \in \widehat{C}$ s.t. $s' \prec s$ and s' is $\prec$-minimal in $\widehat{C}$. We say that S satisfies the smoothness condition iff for every $C \in \mathcal{L}$, $\widehat{C}$ is smooth.

We are now ready for our definition of preferential model.

Definition 5 (Preferential Model). *A preferential model is a triple $\mathscr{P} = \langle S, \ell, \prec\rangle$ where S is a set of states satisfying the smoothness condition, ℓ is a labeling function mapping states to elements of $\mathscr{U}$, and $\prec$ is a strict partial order on S, i.e., $\prec$ is irreflexive and transitive.*

These formal constructions closely resemble those of Kraus et al. [7] and of Lehmann and Magidor [9], the difference being that propositional valuations are replaced with elements of the set $\mathscr{U}$.

Definition 6 (Preferential Subsumption). *Given concepts $C, D \in \mathcal{L}$ and a preferential model $\mathscr{P} = \langle S, \ell, \prec\rangle$, we say that C is preferentially subsumed by D in $\mathscr{P}$ (denoted $C \sqsubsim_{\mathscr{P}} D$) iff every $\prec$-minimal state $s \in \widehat{C}$ is s.t. $s \in \widehat{D}$.*

We are now in a position to prove one of the central results of this paper.

Theorem 3. *A defeasible subsumption relation is a preferential subsumption relation iff it is defined by some preferential model.*

The significance of this is that the representation result is proved with respect to the same set of properties used to characterize propositional preferential consequence. We therefore argue that preferential models, as we have defined them, provide the foundation for a semantics for preferential (and rational) subsumption for a whole class of DLs and related logics. We do not claim that this is *the* appropriate notion of preferential subsumption for $\mathcal{ALC}$, but rather that it describes the basic framework within which to investigate such a notion.

In order to obtain a similar result for rational subsumption, we restrict ourselves to those preferential models in which $\prec$ is a modular order on states (cf. Definition 1):

Definition 7 (Ranked Model). *A ranked model $\mathscr{P}_r$ is a preferential model $\langle S, \ell, \prec\rangle$ in which $\prec$ is modular.*

Since ranked models are preferential models, the notion of rational subsumption is as in Definition 6. We can then state the following result:

Theorem 4. *A defeasible subsumption relation is a rational subsumption relation iff it is defined by some ranked model.*

4 Rational Closure

One of the primary reasons for defining non-monotonic consequence relations of the kind we have presented above is to get at a notion of *defeasible entailment*: Given a set of subsumption statements of the form $C \precsim D$ or $C \sqsubseteq D$, which other subsumption statements, defeasible and classical, should one be able to derive from this? It can be shown that classical subsumption statements of the form $C \sqsubseteq D$ can be encoded as defeasible subsumption statements of the form $C \sqcap \neg D \precsim \bot$. For the remainder of this paper we shall therefore concern ourselves only with *finite* sets of defeasible subsumption statements, and refer to these as *defeasible TBoxes*, denoted $\mathcal{T}$. We permit ourselves the freedom to include classical subsumption statements of the form $C \sqsubseteq D$ in a defeasible TBox, with the understanding that it is an encoding of the defeasible subsumption statement $C \sqcap \neg D \precsim \bot$.

Our aim in this section is to show that the results for the propositional case [9] with respect to the question above can be 'lifted' to $\mathcal{ALC}$. We provide here appropriate notions of preferential entailment and rational closure. It must be emphasized that the results obtained in this section rely heavily on similar results obtained by Lehmann and Magidor [9] for the propositional case, and the semantics for preferential and rational subsumption presented in Section 3. Similar to the results of that section, our claim is not that the versions of preferential and rational closure here are *the* appropriate ones for $\mathcal{ALC}$. In fact, our conjecture is that they are *not*, due to their propositional nature. However, we claim that they provide the appropriate springboard from which to investigate more appropriate versions, for $\mathcal{ALC}$, as well as for other DLs and related logics.

The version of rational closure defined here provides us with a strict generalization of classical entailment for $\mathcal{ALC}$ TBoxes in which the expressivity of $\mathcal{ALC}$ is enriched with the ability to make defeasible subsumption statements. For example, consider the defeasible $\mathcal{ALC}$ TBox:

$$\mathcal{T} = \{BM \sqsubseteq M, VM \sqsubseteq M, M \precsim \neg F, BM \precsim F\},$$

where BM abbreviates the concept *BacterialMeningitis*, M stands for *Meningitis*, VM for *viralMeningitis*, and F abbreviates *FatalDisease*. One should be able to conclude that viral meningitis is usually non-fatal ($VM \precsim \neg F$). On the other hand, we should not conclude that fatal versions of meningitis are usually bacterial ($F \sqcap M \precsim BM$), nor, for that matter, that fatal versions of meningitis are usually *not* bacterial ones ($F \sqcap M \precsim \neg BM$).

Armed with the notion of a preferential model (cf. Section 3) we define preferential entailment for $\mathcal{ALC}$ as follows.

Definition 8. *$C \precsim D$ is preferentially entailed by a defeasible TBox $\mathcal{T}$ iff for every preferential model $\mathscr{P}$ in which $E \precsim_{\mathscr{P}} F$ for every $E \precsim F \in \mathcal{T}$, it is also the case that $C \precsim_{\mathscr{P}} D$.*

Firstly, we can show that preferential entailment is well-behaved and coincides with *preferential closure* under the properties of preferential subsumption (i.e., the intersection of all preferential subsumption relations containing a defeasible TBox). More precisely, let $\mathcal{T}$ be a defeasible TBox. Then the set of defeasible subsumption statements preferentially entailed by $\mathcal{T}$, viewed as a binary relation on concepts, is a preferential subsumption relation. Furthermore, a defeasible subsumption statement is preferentially entailed by $\mathcal{T}$ iff it is in the preferential closure of $\mathcal{T}$.

From this it follows that if we use preferential entailment, the meningitis example can be formalized by letting $\mathcal{T} = \{BM \sqsubseteq M\ VM \sqsubseteq M,\ M \precsim \neg F,\ BM \precsim \neg F\}$. However, $VM \precsim \neg F$ is *not* preferentially entailed by $\mathcal{T}$ above (we cannot conclude that viral meningitis is usually not fatal) and preferential entailment is thus generally too weak. We therefore move to rational subsumption relations.

The first attempt to do so is to use a definition similar to that employed for preferential entailment: $C \precsim D$ is rationally entailed by a defeasible TBox $\mathcal{T}$ iff for every ranked model $\mathscr{P}_r$ in which $E \precsim_{\mathscr{P}_r} F$ for every $E \precsim F \in \mathcal{T}$, it is also the case that $C \precsim_{\mathscr{P}_r} D$. However, this turns out to be *exactly* equivalent to preferential entailment. Therefore, if the set of defeasible subsumption statements obtained as such is viewed as a binary relation on concepts, the result is a preferential subsumption relation and is not, in general, a rational consequence relation.

The above attempt to define rational entailment is thus not acceptable. Instead, in order to arrive at an appropriate notion of (rational) entailment we first define a preference ordering on rational subsumption relations, with relations further down in the ordering interpreted as more preferred.

Definition 9. *Let $\precsim_0$ and $\precsim_1$ be rational subsumption relations. $\precsim_0$ is preferable to $\precsim_1$ (written $\precsim_0 \ll \precsim_1$) iff*

- *there is $C \precsim D \in \precsim_1 \setminus \precsim_0$ s.t. for all E s.t. $E \sqcup C \precsim_0 \neg C$ and for all F s.t. $E \precsim_0 F$, we also have $E \precsim_1 F$; and*
- *for every $E, F \in \mathcal{L}$, if $E \precsim F$ is in $\precsim_0 \setminus \precsim_1$, then there is an assertion $G \precsim H$ in $\precsim_1 \setminus \precsim_0$ s.t. $G \sqcup H \precsim_1 \neg H$.*

Space considerations prevent us from giving a detailed motivation for $\ll$ here, but it is essentially the motivation for the same ordering for the propositional case provided by Lehmann and Magidor [9]. Given a defeasible TBox $\mathcal{T}$, the idea is now to define rational entailment as the most preferred (w.r.t. $\ll$) of all those rational subsumption relations which include $\mathcal{T}$.

Lemma 1. *Let $\mathcal{T}$ be a (finite) defeasible TBox and let $\mathcal{R}$ be the class of all rational subsumption relations which include $\mathcal{T}$. There is a unique rational subsumption relation in $\mathcal{R}$ which is preferable to all other elements of $\mathcal{R}$ w.r.t. $\ll$.*

This puts us in a position to define an appropriate form of (rational) entailment for defeasible TBoxes:

Definition 10. *Let $\mathcal{T}$ be a defeasible TBox. The rational closure of $\mathcal{T}$ is the (unique) rational subsumption relation which includes $\mathcal{T}$ and is preferable (w.r.t. $\ll$) to all other rational subsumption relations including $\mathcal{T}$.*

It can be shown that $VM \precsim \neg F$ is in the rational closure of $\mathcal{T}$ (we can conclude viral meningitis is usually not fatal), but that neither $F \sqcap M \precsim BM$ nor $F \sqcap M \precsim \neg BM$ is.

We conclude this section with a result which can be used to define an algorithm for computing the rational closure of a defeasible TBox $\mathcal{T}$. For this we first need to define a ranking of concepts w.r.t. $\mathcal{T}$ which, in turn, is based on a notion of exceptionality. A concept C is said to be *exceptional* for a defeasible TBox $\mathcal{T}$ iff $\mathcal{T}$ *preferentially* entails $\top \precsim \neg C$. A defeasible subsumption statement $C \precsim D$ is exceptional for $\mathcal{T}$ if and only if its antecedent C is exceptional for $\mathcal{T}$.

It turns out that checking for exceptionality can be reduced to classical subsumption checking.

Lemma 2. *Given a defeasible TBox $\mathcal{T}$, let $\mathcal{T}^{\sqsubseteq}$ be its classical counterpart in which every defeasible subsumption statement of the form $D \precsim E$ in $\mathcal{T}$ is replaced by $D \sqsubseteq E$. C is exceptional for $\mathcal{T}$ iff $\top \sqsubseteq \neg C$ is classically entailed by $\mathcal{T}^{\sqsubseteq}$.*

Let $E(\mathcal{T})$ denote the subset of $\mathcal{T}$ containing statements that are exceptional for $\mathcal{T}$. We define a non-increasing sequence of subsets of $\mathcal{T}$ as follows: $\mathcal{E}_0 = \mathcal{T}$, and for $i > 0$, $\mathcal{E}_i = E(\mathcal{E}_{i-1})$. Clearly there is a smallest integer k s.t. for all $j \geq k$, $\mathcal{E}_j = \mathcal{E}_{j+1}$. From this we define the *rank* of a concept w.r.t. $\mathcal{T}$: $r_{\mathcal{T}}(C) = k - i$, where i is the smallest integer s.t. C is not exceptional for $\mathcal{E}_i$. If C is exceptional for $\mathcal{E}_k$ (and therefore exceptional for all $\mathcal{E}$s), then $r_{\mathcal{T}}(C) = 0$. Intuitively, the lower the rank of a concept, the more exceptional it is w.r.t. the TBox $\mathcal{T}$.

Theorem 5. *Let $\mathcal{T}$ be a defeasible TBox. The* rational closure *of $\mathcal{T}$ is the set of defeasible subsumption statements $C \precsim D$ s.t. either $r_{\mathcal{T}}(C) > r_{\mathcal{T}}(C \sqcap \neg D)$, or $r_{\mathcal{T}}(C) = 0$ (in which case $r_{\mathcal{T}}(C \sqcap \neg D) = 0$ as well).*

From this result it is easy to construct a (naïve) decidable algorithm to determine whether a given defeasible subsumption statement is in the rational closure of a defeasible TBox $\mathcal{T}$. Also, if checking for exceptionality is assumed to take constant time, the algorithm is quadratic in the size of $\mathcal{T}$. Given that exceptionality reduces to subsumption checking in $\mathcal{ALC}$ which is ExpTime-complete, it immediately follows that checking whether a given defeasible subsumption statement is in the rational closure of $\mathcal{T}$ is an ExpTime-complete problem. This result is closely related to a result by Casini et al. [4] which we refer to again in the next section.

5 Related Work

Quantz and Ryan [11, 12] were probably the first to consider the lifting of non-monotonic reasoning formalisms to a DL setting. They propose a general framework for Preferential Default Description Logics (PDDL) based on an $\mathcal{ALC}$-like language by introducing a version of default subsumption and proposing a semantics for it. Their semantics is based on a simplified version of standard DL interpretations in which all domains are assumed to be finite and the unique name assumption holds for object names. Their framework is thus much more restrictive than ours. They focus on a version of entailment which they refer to as preferential entailment, but which is to be distinguished from the version of preferential entailment we have presented in this paper. We shall refer to their version as *Q-preferential entailment.*

Q-preferential entailment is concerned with what ought to follow from a set of classical DL statements, together with a set of default subsumption statements, and is parameterised by a fixed partial order on (simplified) DL interpretations. They prove that any Q-preferential entailment satisfies the properties of a preferential consequence relation and, with some restrictions on the partial order, satisfies Rational Monotony as well. Q-preferential entailment can therefore be viewed as something in between the notions of preferential consequence and preferential entailment we have defined for DLs. It is also worth noting that although the Q-preferential entailments satisfy the properties of a preferential consequence relation, Quantz and Ryan do not prove that Q-preferential entailment provides a *characterisation* of preferential consequence.

Britz et al. [3] and Giordano et al. [6] use typicality orderings on *objects* in first-order domains to define versions of defeasible subsumption for $\mathcal{ALC}$ and extensions thereof. Both approaches propose specific non-monotonic consequence relations, and hence their semantic constructions are special cases of the more general framework we have provided here. In contrast, we provide a general semantic framework which is relevant to all logics with a possible worlds semantics. This is because our preference semantics is not defined in terms of orders on interpreted DL objects relative to given concepts, but rather in terms of a single order on relational structures. Our semantics for defeasible subsumption yields a single order at the meta level, rather than ad hoc relativized orders at the object level.

Casini and Straccia [4] recently proposed a syntactic operational characterization of rational closure in the context of description logics, based on classical entailment tests only, and thus amenable to implementation. Their work is based on that of Lehmann and Magidor [9], Freund [5] and Poole [10], and represents an important building block in the extension of preferential consequence to description logics. However, this work lacks a semantics, and we can only at present conjecture that the rational closure produced by their algorithm coincides with the notion of the rational closure of a defeasible TBox presented in this paper.

6 Conclusion and Future Work

The main contribution of this paper is the provision of a natural and intuitive formal semantics for preferential and rational subsumption for the description logic $\mathcal{ALC}$. We claim that our semantics provides the foundation for extending preferential reasoning in at least three ways. Firstly, as we have seen in Section 4, it allows for the 'lifting' of preferential entailment and rational closure from the propositional case to the case for $\mathcal{ALC}$. Without the semantics such a lifting may be possible in principle, but will be very hard to prove formally. Secondly, it paves the way for defining similar results for other DLs, as well as other similarly structured logics, such as logics of action and belief. We are at present investigating similar notions for logics of action. And thirdly, it provides the tools to tighten up the versions of preferential and rational subsumption for $\mathcal{ALC}$ presented in this paper in order to truly move beyond the propositional. The latter point is the obvious one to pursue first when it comes to future work. Below we provide some initial ideas on moving beyond propositional properties. The value added by the semantics is the ability it provides to test whether appropriate constraints on the orderings in ranked models can be found that matches the new properties.

Consider the following defeasible TBox, which is a slightly modified version of our previous meningitis example:

$$\mathcal{T} = \{BM \sqsubseteq M, M \precsim \neg F, BM \precsim F, \top \sqsubseteq \forall cm.F\},$$

where BM, M, and F are as before, and cm abbreviates the role *causaMortis*. The last statement encodes the classical subsumption statement that all causes of death are fatal.

It is easily verified that $\exists pc.BM \sqsubseteq \exists pc.M$ is in the rational closure of $\mathcal{T}$ (where pc abbreviates the role *potentiallyCauses*), and so it should be since it is entailed by $BM \sqsubseteq M$. We would also expect to conclude $\exists pc.M \precsim \exists pc.\neg F$ from $\mathcal{T}$ since it contains $M \precsim \neg F$. However, there is no propositional property to guarantee the latter. This prompts us to consider the following property:

$$\frac{C \precsim D}{\exists r.C \precsim \exists r.D} \text{ (Naïve Role Introduction)}$$

Arguably, then, if meningitis is usually non-fatal, then potential causes of meningitis are usually potential causes of something non-fatal.

But there are problems with this reasoning. The following example makes this explicit: From $M \precsim \neg F$, Naïve Role Introduction also allows us to conclude that $\exists cm.M \precsim \exists cm.\neg F$. So usually, fatal cases of meningitis are fatal cases of something non-fatal. This is clearly counter-intuitive. Intuitively, cm usually relates to an abnormal type of meningitis, such as bacterial meningitis, which is usually fatal. An additional blocking mechanism is therefore needed to prevent the rule from being applied when the entire range of the role r is abnormal with respect to C. In order to provide such a mechanism, we need to go beyond $\mathcal{ALC}$, and include the ability to express *role inverses*.[3] We then have the following Role Introduction property:

$$\text{(RI)} \ \frac{C \precsim D, \quad C \not\precsim \forall r^{-}.\bot}{\exists r.C \precsim \exists r.D}$$

The effect of the premise $C \not\precsim \forall r^{-}.\bot$ is to block application of the rule if C is normally disjoint from the range of r. On the other hand, if normally C overlaps with the range of r, it follows that $\exists r.C \precsim \exists r.D$.

[3] Given a role name r, the role inverse of r is denoted by r^{-}. For an interpretation $\mathcal{I}$, $(r^{-})^{\mathcal{I}} = \{(y, x) \mid (x, y) \in r^{\mathcal{I}}\}$.

Now consider again the example above. The intention of role cm is modeled by $\top \sqsubseteq \forall cm.F$, the intuition being that only something fatal can be the cause of death. It then follows classically that $\neg F \sqsubseteq \forall cm^{-}.\bot$, and by (RW) that $M \precsim \forall cm^{-}.\bot$. This property (RI) is therefore blocked for the statement $M \precsim \neg F$. Note that (RI) applied to pc is not blocked, as we do not have that $M \precsim \forall pc^{-}.\bot$. (RI) applied to cm is also not blocked for $BM \precsim F$. The interesting thing about (RI) is that it does *not* hold for the preferential closure of a TBox, whereas it *does* hold for the rational closure.

This illustrates that there are intuitively appealing properties characterizing rational DL-entailment that merit further investigation.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge, 2 edn. (2007)
2. Blackburn, P., van Benthem, J., Wolter, F.: Handbook of Modal Logic. Elsevier (2006)
3. Britz, K., Heidema, J., Meyer, T.: Semantic preferential subsumption. In: Proc. KR. pp. 476–484. AAAI Press/MIT Press (2008)
4. Casini, G., Straccia, U.: Rational closure for defeasible description logics. In: Proc. of JELIA. pp. 77–90. Springer-Verlag (2010)
5. Freund, M.: Preferential reasoning in the perspective of Poole default logic. Artificial Intelligence 98, 209–235 (1998)
6. Giordano, L., Olivetti, N., Gliozzi, V., Pozzato, G.: $\mathcal{ALC}+T$: a preferential extension of description logics. Fund. Informatica 96(3), 341–372 (2009)
7. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)
8. Lehmann, D., Magidor, M.: Preferential logics: the predicate calculus case. In: Proc. of TARK. pp. 57–72 (1990)
9. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)
10. Poole, D.: A logical framework for default reasoning. Artificial Intelligence 36, 27–47 (1988)
11. Quantz, J.J.: A Preference Semantics for Defaults in Terminological Logics. In: Proc. of KR. pp. 294–305 (1992)
12. Quantz, J., Ryan, M.: Preferential Default Description Logics. Tech. rep., TU Berlin (1993), `www.tu-berlin.de/fileadmin/fg53/KIT-Reports/r110.pdf`
13. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. of IJCAI. pp. 466–471 (1991)

Goal-oriented Query Rewriting for OWL 2 QL

Alexandros Chortaras, Despoina Trivela, and Giorgos Stamou

School of Electrical and Computer Engineering,
National Technical University of Athens,
Zografou 15780, Athens, Greece
{achort,gstam}@cs.ntua.gr, despoina@image.ntua.gr

Abstract. We present an optimized query rewriting algorithm for OWL 2 QL that computes the rewriting set of a user query by avoiding unnecessary inferences and extended clause subsumption checks. The evaluation shows a significant performance improvement in comparison to other similar approaches. Alternatively, instead of a rewriting set, the algorithm can produce an equivalent non-recursive datalog program.

1 Introduction

The problem of answering *conjunctive queries* (CQ) over expressive DL ontologies suffers from high worst-case complexity. The DL-Lite$_R$ language [1], which underpins the OWL 2 QL profile, overcomes this problem by allowing a limited expressivity. In DL-Lite$_R$, the CQ answering problem is tractable from the data point of view, and can be solved by splitting the answering procedure in two steps [5, 1, 6]: the query *rewriting*, in which the CQ is expanded into a union of CQs (UCQ), and the *execution* of the UCQ over the database. Apart from having the advantage of using the mature relational database technology, rewriting can be based on first order resolution-based reasoning algorithms [4]. The main restriction is that, for large terminologies and/or large queries, the exponential complexity in the query size may result in a very large number of rewritings.

Several CQ answering algorithms for DL-Lite$_R$ have been proposed. In [2, 7], the rewriting strategy is based on reformulating the conjuncts of the query according to the taxonomic information of the ontology. Although the strategy is effective, some of the ontology axioms must be rewritten in terms of auxiliary roles. This restriction is lifted in [4], which proposes a resolution-based rewriting strategy, called RQR. However, its strategy may get tangled in long inference paths leading to unnecessary or to non function free rewritings. Such rewritings are discarded in the end, but their participation in the inference process and the increased number of required subsumption checks degrades performance. Another approach, called Presto, is proposed in [6] which, instead of a UCQ, computes a non-recursive datalog program, deferring thus part of the complexity to the database system, where it can be handled using disjunctive views.

In this paper we present Rapid$_f$, a goal-oriented query rewriting algorithm for queries posed over DL-Lite$_R$ ontologies. Instead of exhaustively performing resolution, it performs a restricted sequence of inferences that lead directly to rewriting sets with, hopefully, no unnecessary rewritings. In this way, we avoid a large number of blind inference paths and the need for extended query subsumption checks. Rapid$_f$ improves the Rapid algorithm [3] by supporting the full syntactic expressivity of DL-Lite$_R$ (i.e. axioms of the form $A \sqsubseteq \exists P.B$, $\exists P \sqsubseteq \exists S$, $\exists P \sqsubseteq \exists R.B$), all types of queries, and by further refining the unfolding step and reducing the need for subsumption checks. We describe also a simple modification of Rapid$_f$, called Rapid$_d$, which, instead of the complete set of rewritings, outputs an equivalent non recursive datalog program, similarly to [6].

2 Preliminaries

A DL-Lite$_R$ *ontology* is a tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the *terminology* and $\mathcal{A}$ the assertional knowledge. Formally, $\mathcal{T}$ is a set of axioms of the form shown in Table 1, where A, B are atomic concepts and P, S atomic roles. $\mathcal{A}$ is a finite set of *assertions* of the form $A(a)$ or $P(a, b)$, where a, b are individuals.

A CQ Q has the form $\mathbf{A} \leftarrow \mathcal{B}$, where atom $\mathbf{A}$ is the *head*, and the set of atoms $\mathcal{B}$ (seen as a conjunction) is the *body* of Q. We denote $\mathcal{B}$ by $\mathsf{body}\, Q$, and $\mathbf{A}$ by $\mathsf{head}\, Q$. A CQ Q is *posed* over an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ if the predicates of all atoms $\mathbf{B} \in \mathsf{body}\, Q$ are entities of $\mathcal{T}$ and have arities 1 or 2, if the entity is a concept or a role, respectively. Hence, $\mathbf{B}$ is a *concept atom* $B(t)$ or a *role atom* $S(t, s)$. $\mathsf{terms}\, \mathbf{B}$ ($\mathsf{vars}\, \mathbf{B}$, $\mathsf{cons}\, \mathbf{B}$) are the sets of terms (variables, constants) that appear in $\mathbf{B}$. For a set of atoms $\mathcal{B}$ we have $\mathsf{terms}\, \mathcal{B} = \bigcup_{\mathbf{B} \in \mathcal{B}} \mathsf{terms}\, \mathbf{B}$, for a CQ Q we have $\mathsf{terms}\, Q = \mathsf{terms}\, (\{\mathsf{head}\, Q\} \cup \mathsf{body}\, Q)$ and similarly for $\mathsf{vars}\, Q$, $\mathsf{cons}\, Q$. An atom or CQ is *function free* if it contains no functional terms. User queries are always function free. Given a function free CQ Q, a term $t \in \mathsf{terms}\, Q$ is called *distinguished* if it appears in $\mathsf{head}\, Q$, and *non distinguished* otherwise; *bound* if it is a constant, or a distinguished variable, or a variable that appears at least twice in $\mathsf{body}\, Q$, and *unbound* otherwise. We denote the set of bound terms, bound and unbound variables of Q by $\mathsf{terms}^{\mathsf{B}}\, Q$, $\mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}^{\mathsf{UB}}\, Q$, respectively. For an atom $\mathbf{A}$ we also write $\mathsf{vars}^{\mathsf{B}}\, \mathbf{A}$ and $\mathsf{vars}^{\mathsf{UB}}\, \mathbf{A}$ instead of $\mathsf{vars}\, \mathbf{A} \cap \mathsf{vars}^{\mathsf{B}}\, Q$ and $\mathsf{vars}\, \mathbf{A} \setminus \mathsf{vars}^{\mathsf{B}}\, Q$, respectively, if it is clear that $\mathbf{A} \in \mathsf{body}\, Q$, for some Q.

A tuple of constants $\boldsymbol{a}$ is a *certain answer* of a CQ Q posed over the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ iff $\Xi(\mathcal{O}) \cup \{Q\} \models C(\boldsymbol{a})$, where C is the predicate of $\mathsf{head}\, Q$ and $\Xi(\mathcal{O})$ the clausification of $\mathcal{O}$ into first order clauses (see Table 1, where it is assumed that each axiom introduces a distinct function f). The set that contains all answers of Q over $\mathcal{O}$ is denoted by $\mathsf{cert}\, (Q, \mathcal{O})$. It has been proved [5, 1] that for any CQ Q and DL-Lite$_R$ ontology $\mathcal{O}$, there is a set $\mathcal{Q}$ of function free CQs (called *query rewritings*) such that $\mathsf{cert}(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \bigcup_{Q' \in \mathcal{Q}} \mathsf{cert}(Q', \langle \emptyset, \mathcal{A} \rangle)$.

Formally, a function free CQ Q' is a *rewriting* of a CQ Q posed over ontology $\mathcal{O}$, i ff$Q$ and Q' have the same head predicate and $\Xi(\mathcal{O}) \cup \{Q\} \models Q'$. Nevertheless, not all possible rewritings are needed for the complete computation of $\mathsf{cert}\, (Q, \mathcal{O})$, since some of them may be equivalent or subsumed by others. We say that a CQ Q *subsumes* a CQ Q' (or Q' *is subsumed by* Q) and write $Q \rhd Q'$, iff there is a substitution θ such that $\mathsf{head}\, (Q\theta) = \mathsf{head}\, Q'$ and $\mathsf{body}\, (Q\theta) \subseteq \mathsf{body}\, Q'$. If Q and Q' are mutually subsumed, they are *equivalent*. If $\mathcal{Q}$ is a set of CQs and for some

CQ Q there is a $Q' \in \mathcal{Q}$ equivalent to Q, we write $Q \hat{\in} \mathcal{Q}$. A set $\mathsf{rewr}\,(Q,\mathcal{O})$ is a *rewriting set* of the CQ Q over $\mathcal{O}$ iff for each rewriting Q' of Q over $\mathcal{O}$, either $Q' \hat{\in} \mathsf{rewr}\,(Q,\mathcal{O})$ or there is a $Q'' \in \mathsf{rewr}\,(Q,\mathcal{O})$ such that $Q'' \rhd Q'$. Given a CQ Q, let Q' be the CQ $\mathsf{head}\,Q \leftarrow \{\mathbf{B}\}_{\mathbf{B}\in\mathcal{B}}$ for some $\mathcal{B} \subseteq \mathsf{body}\,Q$. If $\mathcal{B}$ is a minimal subset of $\mathsf{body}\,Q$ such that $Q \rhd Q'$, Q' is called *condensed* or a *condensation* of Q, and is denoted by $\mathsf{cond}\,Q$. Since a CQ is equivalent to its condensation, we can find $\mathsf{cert}\,(Q,\mathcal{O})$ by computing a rewriting set of Q that contains only condensed rewritings and no two rewritings Q, Q' such that $Q \rhd Q'$. Hence, we say that Q' is a *core rewriting* of a CQ Q over $\mathcal{O}$, iff it is a rewriting of Q over O, it is condensed, and there is no (non equivalent) rewriting Q'' of Q over $\mathcal{O}$ such that $Q'' \rhd Q'$. The *core rewriting set* $\mathsf{rewr}^{\mathsf{C}}\,(Q,\mathcal{O})$ of Q over $\mathcal{O}$ is the set of all the core rewritings of Q over $\mathcal{O}$.

Axiom	Clause	Axiom	Clause
$A \sqsubseteq B$	$B(x) \leftarrow A(x)$		
$S \sqsubseteq P$	$P(x,y) \leftarrow S(x,y)$	$S \sqsubseteq P^-$	$P(y,x) \leftarrow S(x,y)$
$S^- \sqsubseteq P$	$P(x,y) \leftarrow S(y,x)$	$S^- \sqsubseteq P^-$	$P(y,x) \leftarrow S(y,x)$
$\exists S \sqsubseteq A$	$A(x) \leftarrow S(x,y)$	$\exists S^- \sqsubseteq A$	$A(x) \leftarrow S(y,x)$
$A \sqsubseteq \exists P$	$P(x,f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P^-$	$P(f(x),x) \leftarrow A(x)$
$A \sqsubseteq \exists P.B$	$P(x,f(x)) \leftarrow A(x)$ $B(f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P^-.B$	$P(f(x),x) \leftarrow A(x)$ $B(f(x)) \leftarrow A(x)$
$\exists S \sqsubseteq \exists P$	$P(x,f(x)) \leftarrow S(x,y)$	$\exists S \sqsubseteq \exists P^-$	$P(f(x),x) \leftarrow S(x,y)$
$\exists S^- \sqsubseteq \exists P$	$P(x,f(x)) \leftarrow S(y,x)$	$\exists S^- \sqsubseteq \exists P^-$	$P(f(x),x) \leftarrow S(y,x)$
$\exists S \sqsubseteq \exists P.B$	$P(x,f(x)) \leftarrow S(x,y)$ $B(f(x)) \leftarrow S(x,y)$	$\exists S \sqsubseteq \exists P^-.B$	$P(f(x),x) \leftarrow S(x,y)$ $B(f(x)) \leftarrow S(x,y)$
$\exists S^- \sqsubseteq \exists P.B$	$P(x,f(x)) \leftarrow S(y,x)$ $B(f(x)) \leftarrow S(y,x)$	$\exists S^- \sqsubseteq \exists P^-.B$	$P(f(x),x) \leftarrow S(y,x)$ $B(f(x)) \leftarrow S(y,x)$

Table 1. Translation of DL-Lite$_R$ axioms into clauses of $\Xi(\mathcal{O})$.

3 Goal-oriented Query Rewriting

Rapid$_f$ computes $\mathsf{rewr}^{\mathsf{C}}\,(Q,\mathcal{O})$ for a user query Q in an efficient way. Its strategy is based on the distinguishing property of the bound variables, namely that whenever a CQ Q is used as the main premise in a resolution rule in which an atom $\mathbf{A} \in \mathsf{body}\,Q$ unifies with the head of the side premise and the mgu θ contains a binding v/t for some variable $v \in \mathsf{vars}^{\mathsf{B}}\,Q$, the application of θ affects several atoms of the query apart from $\mathbf{A}$.

Rapid$_f$ consists of the following steps: (1) The *clausification* step, in which O is transformed into $\Xi(\mathcal{O})$. (2) The *shrinking* step, in which the clauses of $\Xi(\mathcal{O})$ are selectively used as side premises in resolution rule applications in order to compute rewritings which differ from the user query Q in that they do not contain one or more variables in $\mathsf{vars}^{\mathsf{B}}\,Q$, because the application of the resolution rule led to their unification with a functional term which subsequently was eliminated. (3) The *unfolding* step, which uses the results of the previous step to compute the remaining rewritings of Q, by applying the resolution rule without that the bound variables of the main premise are affected. In principle, only unbound variables are eliminated or introduced at this step. However, some bound variables of the main premise may also be eliminated, not through the introduction and subsequent elimination of functional terms, but while condensing the conclusion. Obviously, the same can happen at the shrinking step. (4) The *subsumption check* step, in which subsumed rewritings are removed.

For efficienty, Rapid$_f$ does not implement the shrinking and unfolding steps by applying directly the resolution rule. Instead, a shrinking and unfolding inference rule are defined, which combine a series of several successful resolution rule application steps into one. In this way, the resolution rule is used only if it eventually leads to a function free and hopefully also non subsumed rewriting, and a large number of unnecessary inferences is avoided. The rules of Rapid$_f$ make use of the unfolding and function sets of atom.

3.1 Atom Unfolding Sets

The saturation of $\Xi(\mathcal{O})$ w.r.t. the resolution rule contains clauses of the form

$$\begin{array}{lll} A(x) \leftarrow B(x), & A(x) \leftarrow S(x,y), & P(x,y) \leftarrow S(x,y), \\ P(x,f(x)) \leftarrow B(x), & & P(x,f(x)) \leftarrow S(x,y), \\ P(g(x),f(g(x))) \leftarrow B(x), & & P(g(x),f(g(x))) \leftarrow S(x,y), \\ P(g(h(x)),f(g(h(x)))) \leftarrow B(x), \ \dots & & P(g(h(x)),f(g(h(x)))) \leftarrow S(x,y), \ \dots \end{array}$$

as well as the respective clauses with the role atom arguments inverted. We note that in the clauses of the first two rows, the non functional terms of the head appear also in the body. Based on this remark, and given that in the unfolding step we want the bound variables not to unify with functional terms but be preserved in the conclusion, we define the unfolding of an atom as follows:

Definition 1. *Let* $\mathbf{A}$ *be a function free atom and* T *a subset of* $\mathsf{terms}\,\mathbf{A}$*. Atom* $\mathbf{B}\theta'$ *is an* unfolding *of* $\mathbf{A}$ *w.r.t.* T *iff* $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ *for some substitution* θ *on a (possibly empty) subset of* $\mathsf{vars}\,\mathbf{A} \setminus T$ *to functional terms, where* θ' *is a renaming of* $\mathsf{vars}\,\mathbf{B} \setminus T$ *such that for* $v \in \mathsf{vars}\,\mathbf{B} \setminus T$ *we have* $v\theta' \notin \mathsf{vars}\,\mathbf{A}$*.*

In the above, $\vdash_{\mathcal{R}}$ denotes derivability under the first-order resolution rule. Essentially, $\mathbf{B}\theta'$ is an unfolding of $\mathbf{A}$ w.r.t. T if it is the body of a clause inferrable from $\Xi(\mathcal{O})$ that has in its head an atom $\mathbf{A}'$ (of the same predicate as $\mathbf{A}$), and both $\mathbf{B}$ and $\mathbf{A}'$ contain unaltered all terms in T (which should contain the bound terms in $\mathbf{A}$). Since the variable renaming θ' contains no essential information, we collect all unfoldings and define the *unfolding set* of atom $\mathbf{A}$ for T w.r.t. $\Xi(\mathcal{O})$ as the set $\mathcal{D}(\mathbf{A};T) = \{\mathbf{B} \mid \Xi(\mathcal{O}) \cup \{\mathbf{A}\} \vdash_{\mathcal{J}(T)} \mathbf{B}\}$, where $\mathcal{J}(T)$ are the inference rules shown in Table. 2, in the form $\frac{\mathbf{A}\ C}{\mathbf{B}}$. Given T, $\mathbf{A}$ (the main premise) and a clause $C \in \Xi(\mathcal{O})$ (the side premise), by applying the respective rule we get atom $\mathbf{B}$ (the conclusion). We also define the set $\hat{\mathcal{D}}(\mathbf{A};T) = \mathcal{D}(\mathbf{A};T) \cup \{\mathbf{A}\}$. It is easy to prove that given $\mathbf{A}$ and $T \neq \emptyset$ we have $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} \mathbf{A}\theta \leftarrow \mathbf{B}$ iff $\mathbf{B}\theta' \in \mathcal{D}(\mathbf{A};T)$, for θ,θ' as defined in Def. 1.

T	rule	T	rule
$\{\}$, $\{t\}$	$\dfrac{A(t) \quad A(x) \leftarrow B(x)}{B(t)}$	$\{\}$	$\dfrac{A(t) \quad A(f(x)) \leftarrow B(x)}{B(z)}$
$\{\}$	$\dfrac{A(t) \quad A(f(x)) \leftarrow S(x,y)}{S(z,w)}$	$\{\}$	$\dfrac{A(t) \quad A(f(x)) \leftarrow S(y,x)}{S(w,z)}$
$\{\}$, $\{t\}$	$\dfrac{A(t) \quad A(x) \leftarrow S(x,y)}{S(t,z)}$	$\{\}$, $\{t\}$	$\dfrac{A(t) \quad A(x) \leftarrow S(y,x)}{S(z,t)}$
$\{\}$, $\{t\}$	$\dfrac{P(t,v) \quad P(x,f(x)) \leftarrow B(x)}{B(t)}$	$\{\}$, $\{t\}$	$\dfrac{P(v,t) \quad P(f(x),x) \leftarrow B(x)}{B(t)}$
$\{\}$, $\{t\}$	$\dfrac{P(t,v) \quad P(x,f(x)) \leftarrow S(x,y)}{S(t,z)}$	$\{\}$, $\{t\}$	$\dfrac{P(v,t) \quad P(f(x),x) \leftarrow S(x,y)}{S(t,z)}$
$\{\}$, $\{t\}$	$\dfrac{P(t,v) \quad P(x,f(x)) \leftarrow S(y,x)}{S(z,t)}$	$\{\}$, $\{t\}$	$\dfrac{P(v,t) \quad P(f(x),x) \leftarrow S(y,x)}{S(z,t)}$
$\{t\}$, $\{s\}$, $\{\}$, $\{t,s\}$	$\dfrac{P(t,s) \quad P(x,y) \leftarrow S(x,y)}{S(t,s)}$	$\{t\}$, $\{s\}$, $\{\}$, $\{t,s\}$	$\dfrac{P(t,s) \quad P(x,y) \leftarrow S(y,x)}{S(s,t)}$

Table 2. The $\mathcal{J}(T)$ inference rules (w and z are any newly introduced variables).

3.2 Atom Function Sets

As we have already seen, the closure of $\Xi(\mathcal{O})$ contains clauses of the form $P(x, f(x)) \leftarrow B(x)$, $P(f(x), x) \leftarrow B(x)$ and $A(f(x)) \leftarrow B(x)$, as well as of the form $P(g(x), f(g(x))) \leftarrow B(x)$ and $P(g(x), f(g(x))) \leftarrow S(x, y)$. Unlike in the unfolding case, now we are interested in the behavior of the functional term $f(x)$, which appears in the head but not in the body, because if $f(x)$ appears in the body of a rewriting, it may be possible to eliminate it by using such clauses. Let $\mathsf{funcs}\,\Xi(\mathcal{O})$ be the set of all functions in $\Xi(\mathcal{O})$. According to Table 1, each DL-Lite$_R$ axiom that has an existential quantifier in the RHS introduces a distinct function f. Hence, each function $f \in \mathsf{funcs}\,\Xi(\mathcal{O})$ is uniquely associated with the concept or role that appears in the LHS of the axiom that introduces f. Let $\mathsf{atom}\,f[x]$ denote the atom that (a) has as predicate the entity associated with f, (b) has the variable x in the place of the bound variable of the respective axiom of Table 1 which introduces f, and (c) has a distinct variable (not elsewhere used, as needed) in the place of the unbound variable, if any. E.g. if the axiom $A \sqsubseteq \exists P.B$ introduces function f_1 then $\mathsf{atom}\,f_1[x]$ is the atom $A(x)$, and if the axiom $\exists S^- \sqsubseteq \exists P.B$ introduces function f_2 then $\mathsf{atom}\,f_2[x]$ is the atom $S(z, x)$, where z is some variable not used elsewhere. We define the set of all functions that may appear in the place of a bound variable v of an atom $\mathbf{A}$ when resolving any of its unfoldings with a non function free clause in $\Xi(\mathcal{O})$ as follows:

Definition 2. *Let* $\mathbf{A}$ *be a function free atom,* T *a non empty subset of* $\mathsf{terms}\,\mathbf{A}$ *and* v *a variable in* $\mathsf{vars}\,\mathbf{A} \cap T$*. The* function set $\mathcal{F}_v(\mathbf{A}; T)$ *of all functions associated with* $\mathbf{A}$ *in variable* v *w.r.t.* T *is defined as follows:*

$$\mathcal{F}_v(\mathbf{A}; T) = \begin{array}{l} \{f \mid B(v) \in \hat{\mathcal{D}}(\mathbf{A}; T) \text{ and } B(f(x)) \leftarrow \mathsf{atom}\,f[x] \in \Xi(\mathcal{O})\} \cup \\ \{f \mid S(v,t) \in \hat{\mathcal{D}}(\mathbf{A}; T) \text{ and } S(f(x),x) \leftarrow \mathsf{atom}\,f[x] \in \Xi(\mathcal{O})\} \cup \\ \{f \mid S(t,v) \in \hat{\mathcal{D}}(\mathbf{A}; T) \text{ and } S(x,f(x)) \leftarrow \mathsf{atom}\,f[x] \in \Xi(\mathcal{O})\} \end{array}$$

It follows that (a) if $\mathbf{A} \equiv P(v,t)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} P(f(t),s) \leftarrow \mathsf{atom}\, f[t]$, (b) if $\mathbf{A} \equiv P(t,v)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} P(s,f(t)) \leftarrow \mathsf{atom}\, f[t]$, where in both cases $s=t$ if $t \in T$ otherwise either $s=t$, or $s=g(f(t))$ for some function g, and (c) if $\mathbf{A} \equiv A(v)$ then $f \in \mathcal{F}_v(\mathbf{A};T)$ iff $\Xi(\mathcal{O}) \vdash_{\mathcal{R}} A(f(t)) \leftarrow \mathsf{atom}\, f[t]$. Again, T stands for the set of bound terms in $\mathbf{A}$.

3.3 Query Shrinking

The shrinking step computes rewritings that can be inferred from the user query Q by eliminating one or more of its bound variables through their unification with a functional term. Given that the rewritings in $\mathsf{rewr}\,(Q,\mathcal{O})$ are function free, if a function is introduced in some rewriting during the standard resolution-based inference process, subsequently it must be eliminated. However, we know that each function appears in at most two clauses of $\Xi(\mathcal{O})$, both of which have as body the atom $\mathsf{atom}\, f[x]$. Now, the functional term $f(x)$ can be introduced in a CQ only if some inference led to the substitution of a bound variable v by $f(x)$. Hence, in order for $f(x)$ to be eliminated, all atoms in which $f(x)$ has been introduced must contain f in their function sets, for the appropriate argument. Moreover, if Q contains the terms say $P(x,v)$ and $P(v,y)$ and v is eliminated this way by unifying with $f(x)$, variables x and y must be unified. If in place of x, y there are constants, these should coincide in order for the inference to be possible. This is the intuition behind the following shrinking inference rule:

Definition 3. *Let Q be a CQ and v a non distinguished bound variable of Q. Write Q in the form $\mathbf{A} \leftarrow \mathbf{B}_1,\ldots,\mathbf{B}_k,\mathbf{C}_1,\ldots,\mathbf{C}_n$, where $\mathbf{B}_i$ are the atoms in* $\mathsf{body}\, Q$ *that contain v, and $\mathbf{C}_i$ the remaining atoms. Let also $\mathcal{C} = \bigcup_{i=1}^{k} \mathsf{cons}\, \mathbf{B}_i$ and $\mathcal{X} = \bigcup_{i=1}^{k} (\mathsf{vars}^{\mathsf{B}}\, \mathbf{B}_i) \setminus v$. The* shrinking rule *$\mathcal{S}$ on Q is defined as follows:*

$$\frac{\mathbf{A} \leftarrow \mathbf{B}_1,\ldots,\mathbf{B}_k,\mathbf{C}_1,\ldots,\mathbf{C}_n \quad f \in \bigcap_{i=1}^{k} \mathcal{F}_v(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i) \wedge |\mathcal{C}| \leq 1}{\mathsf{cond}\,(\mathbf{A}\theta \leftarrow \mathsf{atom}\, f[t], \mathbf{C}_1\theta,\ldots,\mathbf{C}_n\theta)}$$

where $\theta = \bigcup_{x \in \mathcal{X}} \{x/t\}$, and $t=a$ if $\mathcal{C}=\{a\}$ otherwise t is a variable $\notin \mathsf{vars}\, Q$.

3.4 Query Unfolding

Let $\mathcal{S}^*(Q)$ be the closure of $\mathsf{cond}\, Q$ under application of the inference rule $\mathcal{S}$, for any CQ Q. By construction, $\mathcal{S}^*(Q)$ contains a 'representative' for all query structures that can result from Q by eliminating one or more variables in $\mathsf{vars}^{\mathsf{B}}\, Q$ by using functional terms. This representative can be considered as a 'top' query, in the sense that in can produce several more CQs with no further structural changes due to bindings of bound variables with functional terms. Hence, the remaining rewritings can be obtained by computing, for each $Q' \in \mathcal{S}^*(Q)$, all CQs that can be inferred from Q' by replacing one or more of its atoms by one of their unfoldings. In this way we can eventually compute all rewritings of Q. This can be achieved by applying the following unfolding inference rule:

Definition 4. *An* unfolding *of CQ $Q : \mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, is the conclusion of any application of the following* unfolding rule $\mathcal{W}$*:*

$$\frac{\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n \qquad \mathbf{C}_i \in \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i) \text{ for } i = 1 \ldots n}{\mathsf{cond}\,(\mathbf{A} \leftarrow \mathbf{C}_1\gamma_1, \ldots, \mathbf{C}_n\gamma_n)}$$

where γ_i is a renaming of $\mathsf{vars}^{\mathsf{UB}}\, \mathbf{C_i}$ such that $x\gamma_i \notin \bigcup_{j=1, j\neq i}^{n} \mathsf{vars}\,(\mathbf{C}_j\gamma_j)$ for all $x \in \mathsf{vars}^{\mathsf{UB}}\, \mathbf{C}_i$.

Let $\mathcal{W}^*(Q)$ be the closure of a $\mathsf{cond}\, Q$ under application of the inference rule $\mathcal{W}$, for any Q. The strategy by which Rapid$_f$ computes the core rewriting set of a user query Q is justified by the following theorem:

Theorem 1. *Let Q be a CQ over a DL-Lite$_R$ ontology $\mathcal{O}$.*
If $Q' \in \bigcup_{Q'' \in \mathcal{S}^*(Q)} \mathcal{W}^*(Q'')$ then $Q' \hat{\in} \mathsf{rewr}\,(Q, \mathcal{O})$ (soundness), and if $Q' \in \mathsf{rewr}^{\mathsf{C}}\,(Q, \mathcal{O})$ then $Q' \hat{\in} \bigcup_{Q'' \in \mathcal{S}^*(Q)} \mathcal{W}^*(Q'')$ (completeness).

3.5 Query Unfolding Optimization

If we apply exhaustively the $\mathcal{W}$ rule in order to compute $\mathcal{W}^*(Q)$, we may end up with many subsumed rewritings. Because the subsumption check operation needed to remove them is very costly, Rapid$_f$ applies $\mathcal{W}$ in a cleverer way, so as to get as few as possible subsumed rewritings. In fact, it restates the unfolding problem as follows: Given a CQ Q of the form $\mathbf{A} \leftarrow \mathbf{B}_1, \ldots, \mathbf{B}_n$, find the non subsumed CQs that are conclusions of all possible applications of $\mathcal{W}$ on Q. For convenience, define $\mathcal{B}_i = \hat{\mathcal{D}}(\mathbf{B}_i; \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i)$, so that we get the sequence of the possibly non disjoint unfolding sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$. For simplicity, we drop the substitutions γ_i in Def. 4 by assuming that each time a rule of $\mathcal{J}(T)$ that introduces a new variable is applied, this variable does not appear elsewhere.

For any $\mathbf{B} \in \bigcup_{i=1}^{n} \mathcal{B}_i$, define the set $\mathsf{ind}\, \mathbf{B} = \{j \mid \mathbf{B} \in \mathcal{B}_j\}$ of the indices of all unfolding sets that contain $\mathbf{B}$. We call the set $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ with $k \leq n$ a *selection* for Q iff (a) $\bigcup_{i=1}^{k} \mathsf{ind}\, \mathbf{C}_i = \mathbb{N}_n$, where $\mathbb{N}_n \doteq \{1, \ldots, n\}$, and (b) $\mathsf{ind}\, \mathbf{C}_i \setminus \mathsf{ind}\, \mathbf{C}_j \neq \emptyset$ for all $i, j \in \mathbb{N}_k$, i.e. if $\mathcal{C}$ contains at least one atom from each unfolding set and no two sets $\mathsf{ind}\, \mathbf{C}_i$ overlap fully. Clearly, a selection corresponds to an unfolding of Q, in particular to $\mathbf{A} \leftarrow \mathcal{C}$. However, of interest are the *minimal selections*, which can produce non subsumed rewritings. We call a selection $\mathcal{C}$ for Q minimal, iff there is no selection $\mathcal{C}'$ for Q such that $\mathcal{C}' \subset \mathcal{C}$, i.e. if condition (b) above is replaced by the stronger condition $\mathsf{ind}\, \mathbf{C}_i \setminus \left(\bigcup_{j=1, j\neq i}^{k} \mathsf{ind}\, \mathbf{C}_j \right) \neq \emptyset$ for all $i \in \mathbb{N}_k$, i.e. if all atoms $\mathbf{C}_i$ need to be present in set $\mathcal{C}$ in order for $\bigcup_{i=1}^{k} \mathsf{ind}\, \mathbf{C}_i = \mathbb{N}_n$ to hold. If this were not the case, we could form the selection $\mathcal{C}' = \{\mathbf{C}_1, \ldots, \mathbf{C}_{i-1}, \mathbf{C}_{i+1}, \mathbf{C}_k\} \subset \mathcal{C}$, hence $\mathcal{C}$ would not be minimal.

In this computation of minimal selections only equality between the elements of the sets $\mathcal{B}_i$ is taken into account, and not subsumption relations. However, an unfolding set may contain an atom with an unbound variable (e.g. $P(x, *)$, where $*$ is unbound) which unifies with an atom of another unfolding set that contains only bound variables (e.g. $P(x, y)$). The unfoldings of a CQ Q resulting after such unifications are made may subsume or be subsumed by several of the unfoldings given directly by other minimal selections for Q. In order to take into account atom subsumption relations, we compute all possible bindings for the unbound variables that appear in the sets $\mathcal{B}_i$ in advance and enrich the respective sets $\mathcal{B}_i$ with the respective atoms, before computing the minimal selections. In particular, if for some $i, j \in \mathbb{N}_n$ we have $\mathbf{C} \in \mathcal{B}_i$ and $\mathbf{C}' \in \mathcal{B}_j$ and there is a substitution θ on $\mathsf{vars}^{\mathsf{UB}}\, \mathbf{C}'$ such that $\mathbf{C}'\theta = \mathbf{C}$, we add $\mathbf{C}$ to $\mathcal{B}_j$. The presence of any such two atoms $\mathbf{C}'$ and $\mathbf{C}$ in any pair $\mathcal{B}_i, \mathcal{B}_j$, regardless of whether they were present from the beginning or introduced at the enrichment phase, establishes a *parent-child* relationship between $\mathbf{C}'$ and $\mathbf{C}$. Let $\mathsf{parents}\, \mathbf{C}$ and $\mathsf{children}\, \mathbf{C}$ denote the set of parent and child atoms of atom $\mathbf{C}$ across all the sets $\mathcal{B}_i$. Obviously, a child can have several parents in different unfolding sets, possibly distinct between each other, and the same holds for the children of a parent.

In order to avoid the production of subsumed rewritings due to such atom subsumptions, for each candidate unfolding $Q : \mathbf{A} \leftarrow \mathcal{C}$ obtained from a minimal selection $\mathcal{C}$, Rapid$_f$ performs two checks: (1) For each parent $\mathbf{C}$ of an atom in $\mathcal{C}$, it constructs a candidate rewriting with body $\mathcal{C}' = \{\mathbf{C}\} \cup (\mathcal{C} \setminus \mathsf{children}\, \mathbf{C})$, i.e it replaces all children of $\mathbf{C}$ by their parent. If $\mathcal{C}'$ is a minimal selection, then Q is discarded because it is subsumed by $\mathbf{A} \leftarrow \mathcal{C}'$. E.g. $Q(x) \leftarrow S(x, y), T(x, y)$ is subsumed by $Q(x) \leftarrow S(x, *), T(x, z)$ where $S(x, y)$ is a child of $S(x, *)$. (2) For each atom $\mathbf{C}$ that is a child of an atom in $\mathcal{C}$ it constructs the candidate body $\mathcal{C}' = \{\mathbf{C}\} \cup \{\mathbf{D} \mid \mathbf{D} \in \mathcal{C} \text{ and } \mathsf{ind}\, \mathbf{D} \nsubseteq \mathsf{ind}\, \mathbf{C}\}$, i.e. it replaces the parent by its child $\mathbf{C}$ and keeps all the remaining atoms of $\mathcal{C}$ that are not 'covered' (in terms of their indices) by $\mathbf{C}$. If $\mathsf{cond}\, (\mathbf{A} \leftarrow \mathcal{C}') \rhd Q$ then Q is discarded because it is subsumed. E.g. $Q(x, y) \leftarrow R(x, y), S(y, w), T(y, z), S(v, z)$ is subsumed by $Q(x, y) \leftarrow R(x, y), S(y, z)$, where $S(y, z)$ is child of both $S(y, w)$ and $S(v, z)$.

The only case an unfolding Q' of Q obtained in this way may subsume another unfolding of Q is when the condensation of Q' does not contain one or more of the variables in $\mathsf{vars}^{\mathsf{B}}\, Q$; this implies that a structural change has happened to $\mathsf{cond}\, Q'$. To cover this case, we always compute the condensation of each unfolding given by the above procedure. If the condensation does not contain a bound variable of Q it is marked as *impure*, otherwise as *pure*. Given that the unfolding step is executed for each rewriting produced by the shrinking step, the final step is the check for subsumed rewritings within the results of the entire unfolding process. The check is done after first grouping the results into sets that are known not to contain subsumed rewritings. As explained, these are the sets of pure unfoldings obtained during the unfolding step for each rewriting given by the shrinking step. Each impure unfolding is considered to be a separate set.

The overall structure of Rapid$_f$ for a user query Q is shown in Algorithm 1. Procedure SHRINK computes $\mathcal{S}^*(Q)$, by iteratively applying Def 3. For each rewriting computed by SHRINK, procedure UNFOLD computes its minimal selections and discards any subsumed unfoldings as described above. Finally, the unfoldings, grouped into sets of pure unfoldings and singleton sets of impure unfoldings, are processed by procedure CHECKSUBSUMPTION, which checks for subsumptions across sets only and removes any subsumed rewritings.

Algorithm 1 The Rapid$_f$ algorithm

```
procedure RAPID$_f$(CQ $Q$, ontology $\mathcal{O}$)
    $\mathcal{Q}_f = \emptyset$
    for all $Q_s \in$ SHRINK$(Q, \mathcal{O})$ do
        $\mathcal{Q}_t \leftarrow \emptyset$
        for all $Q' \in$ UNFOLD$(Q_s, \mathcal{O})$ do
            if $\mathsf{vars}^{\mathsf{B}}\, Q \subseteq \mathsf{vars}\,(\mathsf{cond}\, Q')$ then
                $Q_t \leftarrow Q_t \cup \{Q'\}$
            else
                $Q_f \leftarrow Q_f \cup \{\{\mathsf{cond}\, Q'\}\}$
            end if
        end for
        $Q_f \leftarrow Q_f \cup \{Q_t\}$
    end for
    return CHECKSUBSUMPTION$(Q_f)$
end procedure
```

3.6 Rapid$_d$: Rewritings as a non-recursive Datalog program

The side premises $\mathbf{C}_i \in \hat{\mathcal{D}}(\mathbf{B}_i, \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i)$ of the unfolding rule $\mathcal{W}$ may be seen as the clauses $\mathbf{C}_i \leftarrow \mathbf{D}_i$ for some $\mathbf{D}_i \in \hat{\mathcal{D}}(\mathbf{B}_i, \mathsf{terms}^{\mathsf{B}}\, \mathbf{B}_i)$. Hence, similarly to [6], given a user CQ Q, instead of applying exhaustively the unfolding rule on $\mathcal{S}^*(Q)$, in order to produce all unfoldings and then check for subsumptions among them so as to get $\mathsf{rewr}^{\mathsf{C}}\,(Q, \mathcal{O})$, we can produce a non-recursive datalog program $\mathcal{P}_Q$, which contains the rewritings produced at the shrinking step plus the side premises of the $\mathcal{W}$ rules that can possibly be applied. Rapid$_d$ works exactly this way: The clausification and shrinking steps are as in Rapid$_f$, but the unfolding and subsumption check steps are replaced by a single step which rewrites the unfolding of all atoms that appear in the body of the rewritings in $\mathcal{S}^*(Q)$ in the form of a set of clauses $\mathcal{U}$, which are then appended to the set of rewritings obtained at the shrinking step so that $\mathcal{P}_Q$ is produced. Before doing this however, the rewritings in $\mathcal{S}^*(Q)$ need to be modified in two ways, as in [6].

First, we can remove from the body of the several $Q' \in \mathcal{S}^*(Q)$ the atoms that will certainly produce only subsumed rewritings (in the case we were to apply exhaustively the $\mathcal{W}$ rule, as before, in order to compute all rewritings); this happens if there are two atoms $\mathbf{A}, \mathbf{B} \in \mathsf{body}\, Q'$ and a θ such that $\mathbf{A} = \mathbf{C}\theta$ for some $\mathbf{C} \in \mathcal{D}(\mathsf{terms}^{\mathsf{B}}\, \mathbf{B})$, i.e. if $\mathbf{A}$ is subsumed by $\mathbf{C}$. In this case we just remove $\mathbf{B}$ from $\mathsf{body}\, Q'$. Let $\mathsf{rr}\,(\mathcal{S}^*(Q))$ be the set of clauses obtained in this way from $\mathcal{S}^*(Q)$ and also after removing from it any subsumed clauses.

Next, we must construct the set of clauses $\mathcal{U}$. This is straightforward, but we must take into account the different bindings that bound and unbound variables can have during the unfolding. So, for all atoms $\mathbf{A}$ that appear in the clauses of $\mathsf{rr}\,(\mathcal{S}^*(Q))$ we compute the set $\mathcal{D}(\mathbf{A}; \mathsf{vars}^{\mathsf{B}}\, \mathbf{A})$ and then we construct from $\mathbf{A}$ a new atom $\mathbf{A}'$ by removing from the arguments of $\mathbf{A}$ all unbound variables and replacing the predicate p of $\mathbf{A}$ by a new predicate p_{t_1} or $p_{t_1 t_2}$, if $\mathbf{A}$ is a concept or role atom, respectively, and $t_i = 0$ if the i-th argument of $\mathbf{A}$ is unbound and

$t_i = 1$ otherwise. Finally we *normalize* the clauses in $\mathsf{rr}\,(\mathcal{S}^*(Q))$ by replacing all appearances of $\mathbf{A}$ by $\mathbf{A}'$, and add to $\mathcal{U}$ the clause $\mathbf{A}' \leftarrow \mathbf{A}$ as well as the clause $\mathbf{A}' \leftarrow \mathbf{D}$ for all $\mathbf{D} \in \mathcal{D}(\mathbf{A}; \mathsf{vars}^{\mathsf{B}}\,\mathbf{A})$.

$\mathcal{P}_Q$ is the union of $\mathcal{U}$ and the normalized version of all clauses in $\mathsf{rr}\,(\mathcal{S}^*(Q))$.

4 Evaluation

We evaluated Rapid$_f$ by comparing it with Rapid and Requiem, the implementation of RQR. We used the same datasets as in [4], namely the V, S, U, A, P5, UX, AX, P5X ontologies. (V models European history, S European financial institutions, and A information about abilities, disabilities and devices. U is a DL-Lite$_R$ version of the LUBM benchmark ontology. P5 is synthetic and models graphs with paths of length 5. UX, AX and P5X are obtained by rewriting U, A and P5 without qualified existential restrictions). The results are shown in Table 3. T is the rewriting computation time and R the number of rewritings. For Rapid$_f$ (Rap$_f$), Rapid (Rap) and Requiem (Req), one number is given for the rewritings, since all these algorithms compute the same core rewriting set. For Rapid$_d$ (Rap$_d$), the column R is the number of clauses in $\mathcal{P}_Q$.

		Rap$_f$	Rap	Req		Rap$_d$				Rap$_f$	Rap	Req		Rap$_d$	
$\mathcal{O}$	Q	T	T	T	R	T	R	$\mathcal{O}$	Q	T	T	T	R	T	R
	1	.001	.001	.001	15	.001	16		1	.001	.001	.001	6	.001	7
	2	.001	.001	.001	10	.001	13		2	.001	.001	.015	10	.001	16
V	3	.001	.015	.016	72	.001	29	**P5**	3	.001	.001	.047	13	.001	19
	4	.015	.031	.062	185	.001	44		4	.015	.015	.688	15	.001	21
	5	.016	.016	.015	30	.001	13		5	.015	.015	16.453	16	.001	22
	1	.001	.001	.001	6	.001	7		1	.001	.001	.001	14	.001	15
	2	.001	.001	.062	2	.001	3		2	.001	.001	.031	25	.001	31
S	3	.001	.001	.515	4	.001	5	**P5X**	3	.016	.031	.297	58	.001	34
	4	.001	.001	1.047	4	.001	5		4	.078	.172	7.375	179	.001	36
	5	.001	.001	17.984	8	.001	7		5	1.234	2.625	3:48.690	718	.001	37
	1	.001	.001	.001	2	.001	4		1	.001	.001	.001	5	.001	7
	2	.001	.001	.047	1	.001	2		2	.001	.001	.078	1	.001	2
U	3	.001	.001	.109	4	.001	8	**UX**	3	.001	.001	1.125	12	.001	10
	4	.001	.001	2.031	2	.001	3		4	.001	.001	19.375	5	.001	6
	5	.001	.001	7.781	10	.001	8		5	.001	.015	57.672	25	.001	11
	1	.001	.001	.047	27	.001	54		1	.001	.015	.063	41	.001	69
	2	.001	.001	.047	50	.001	33		2	.109	.141	2.781	1,431	.001	51
A	3	.016	.016	.063	104	.001	33	**AX**	3	.375	.469	29.109	4,466	.001	57
	4	.015	.031	.156	224	.001	60		4	.265	.641	23.516	3,159	.001	85
	5	.062	.078	.610	624	.001	38		5	3.375	49.984	1:56:21.585	32,921	.001	72

Table 3. Evaluation results. The times T are in hh.mm.ss.msec format The results for Requiem are for its greedy modality, which applies forward query subsumption, dependency graph pruning and greedy unfolding.

The results show clearly the efficiency of Rapid$_f$. It is always faster than Rapid, and much faster than Requiem; in several cases the improvement is significant. The most striking case is ontology *AX* and query 5, in which Rapid$_f$ completes the computation of the 32,921 core rewritings in less than 4 seconds, while Rapid needs 50 seconds and Requiem about 2 hours. The more detailed study of this particular case showed that Rapid$_f$ computes directly the final core rewriting set and performs no subsumption checks at all. On the other hand, Rapid spends about 45 seconds checking for subsumptions and Requiem about 1.5 hours.

Table 3 also shows, as expected, that Rapid$_d$ is always much faster than any of the other algorithms, since it does not include the unfolding step, which is the main source of complexity, even for the optimized Rapid$_f$ algorithm. For the same ontologies and query pairs tested in [6], similar times and numbers of rewritings are reported. Note, however that the rewriting sizes do not coincide, because Rapid$_d$ and Presto do not produce the same datalog programs. This is due to the fact that the *Split* and *EliminateEJVars* steps of Presto are performed in a different way by the shrinking step of Rapid$_d$. The expansion of the datalog program to a UCQ is of course the same, for both algorithms.

5 Conclusions

We presented Rapid$_f$, an efficient algorithm for the computation of the core rewriting set of queries posed over DL-Lite$_R$ ontologies. Rapid$_f$ optimizes the inference process by replacing the application of the first order resolution rule by specialized shrinking and unfolding rules, which save the algorithm from many unnecessary rewritings, subsumption checks and blind inference paths. We presented also Rapid$_d$ a modification of Rapid$_f$, which does not unfold the rewritings, but encodes the unfoldings into a datalog program similarly to [6]. The experimental evaluation of Rapid$_f$ showed a significant performance benefit if compared to RQR and Rapid, which in several practical cases can alleviate the exponential behavior. The performance of Rapid$_d$ is similar to Presto, but Rapid$_d$ supports the full syntactic expressivity of DL-Lite$_R$.

References

1. A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev. The DL-Lite family and relations. J. of Artificial Intelligence Research, 36:1–69, (2009).
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite Family. J. of Automated Reasoning, 39:385–429, (2007).
3. A. Chortaras, D. Trivela, G. Stamou. Optimized query rewriting for OWL 2 QL, In Procs of CADE 2011 (*accepted*), (2011).
4. H. Perez-Urbina, I. Horrocks, B. Motik. Efficient query answering for OWL 2. In Procs of ISWC 2009, LNCS 5823:489–504, (2009).
5. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati. Linking data to ontologies. J. on Data Semantics, 10:133–173, (2008).
6. R. Rosati, A. Almatelli. Improving query answering over DL-Lite ontologies. In Procs of KR 2010, pp. 290–300, (2010)
7. M. Stocker, M. Smith. Owlgres: A scalable OWL reasoner. In Procs of OWLED 2008, CEUR-WS.org Vol-432, (2008).

Collective Classification in Semantic Mapping with a Probabilistic Description Logic

Fabiano R. Correa[1], Fabio G. Cozman[2], Jun Okamoto Jr[2]

Department of Civil Construction, University of Sao Paulo, Brazil
`fabiano.correa@usp.br`
[2] Department of Mechatronic Engineering, University of Sao Paulo, Brazil
`{fgcozman, jokamoto}@usp.br`

Abstract. Sensor data classification is very dependent on which features represent primitives. We consider line segments extracted from laser points as primitives, and focus on their collective classification into door or wall objects, so as to build semantic maps. Because features may have non-trivial characteristics, and sensor primitives may be inter-related in complex ways, we represent features of spatial relationships using a probabilistic description logic.

1 Introduction

Recent successes have raised expectations concerning the behavior of mobile robots in dynamic environments [21]. State-of-the-art applications construct precise spatial maps of static environments; however, autonomous robots need more than accurate spatial information when dealing with people or objects that display dynamic change. “Semantic mapping” focuses on the representation of an hierarchy of general objects in the environment, with their individual properties and inter-relationships. Broadly speaking, semantic maps must compactly encode rich information in a scalable manner.

Although there is no unique or precise definition for semantic maping in robotics, in the last five years many researchers have turned to spatial representations tagged with information like: "This segment of laser data is a door" or "This area of the occupancy grid is a room". As such, a semantic map typically means a labeled spatial map, and not really a map interwoved with deep semantic information. Few proposals really include semantic information in their robotic architecture by means of an ontology that relates objects in the environment. Clearly a more detailed look at semantic mapping is worth the study, because through semantic information we may expect to create more natural ways for robots to interact with humans and its environments.

Semantic mapping could deal with different sensor data inputs and output several different representations. Cameras could be employed to construct a map representation based in clusters of images representing different rooms [24]. Other proposals use 3D laser sensors to produce point cloud representations, that allows for object recognition [22], environment segmentation in floor and walls [1], and even the construction of a real map [15]. One interesting application relies in laser sensors that obtains horizontal slices at a fixed height of the environment to create bidimensional spatial maps. Semantic mapping in this context envolves the classification of line segments from already constructed 2D maps of indoor environments. This scenario was firstly proposed by Limketkai *et al.* [12] and latter was also approached by Wang and Domingos [19]. It is a scenario where one can explore different kinds of dependencies between the data, including spatial relationships and appearance. Both previous cited work employed models that combines first-order or relational logic with probabilities to produce line segments classification. The use of probabilities is justified because there is considerable uncertainty in associating dependencies with the possible classes of line segments, both due to the uncertain process of creating line segments from laser points and to changes in sensed objects [20]. And first-order logic is an expressive language that allows for a rich representation of complex relationships between different object in a compact way.

In this paper we focus exactly on the problem of laser data classification, using a combination of logic and probability to represent information extracted from sensor data. At the moment, we provide only probabilistic reasoning in our model while logic elements are used to describe the scenario and to obtain an ontology that could be explored in future applications. We chose to model this problem in a probabilistic description logic called CR$\mathcal{ALC}$, as it seems to provide a reasonable balance between flexibility and computational cost, to be explored in further developments. The next section briefly describes the probabilistic description logic CR$\mathcal{ALC}$. In Section 3, semantic mapping is discussed. Experiments are detailed in Section 4, followed by our conclusions.

2 Credal $\mathcal{ALC}$

A probabilistic description logic, called Credal $\mathcal{ALC}$ (CR$\mathcal{ALC}$), has been proposed recently [4,5,16], in a wave of related efforts [14]. In fact, the literature brings a variety of probabilistic description logics [7,9,10,11,13,3,18]; CR$\mathcal{ALC}$ is based on the popular $\mathcal{ALC}$ logic, adopts an interpretation-based semantics and resorts to the theory of Bayesian networks to allow for judgements of stochastic independence and to obtain inference algorithms.

The vocabulary of CR$\mathcal{ALC}$ contains *individuals*, *concepts*, and *roles*. Concepts and roles are combined to form new concepts using a set of *constructors* from $\mathcal{ALC}$ [17]: *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$) and *value* restriction ($\forall r.C$). A *concept inclusion* is denoted by $C \sqsubseteq D$ and a *concept definition* is denoted by $C \equiv D$, where C and D are concepts; we assume in both cases that C is a concept name. We then say that C *directly uses* D; the relation *uses* is the transitive closure of *directly uses*. Also, the concept $\top$ denotes $C \sqcup (\neg C)$ for some concept C. As in $\mathcal{ALC}$, the semantics is given by a domain $\mathcal{D}$, a set of elements, and an interpretation mapping $\mathcal{I}$ that assigns an element to an individual, a set of elements to a concept, and a binary relation to a role. An interpretation mapping must also comply with constructs of the language; for instance, the interpretation of concept $C \sqcap D$ is $\mathcal{I}(C) \cap \mathcal{I}(D)$, while the interpretation of concept $\forall r.C$ is $\{x \in \mathcal{D} \mid \forall\, y : (x, y) \in \mathcal{I}(r) \to y \in \mathcal{I}(C)\}$. Additionally, CR$\mathcal{ALC}$ accepts ***probabilistic inclusions*** as follows. A probability inclusion reads

$$P(C|D) \in [\alpha_1, \alpha_2],$$

where D is a concept and C is a concept name. The semantics of such a probabilistic inclusion is, informally:

$$\forall x : P(C(x)|D(x)) \in [\alpha_1, \alpha_2], \tag{1}$$

where it is understood that probabilities are over the set of all interpretation mappings $\mathcal{I}$ for a domain $\mathcal{D}$. If D is the concept $\top$ then we write $P(C) \in [\alpha_1, \alpha_2]$. Probabilistic inclusions are required to only have concept names in their conditioned concept (that is, inclusions such as $P(\forall r.C|D)$ are not allowed). Yet another type of probabilistic assessement is possible in CR$\mathcal{ALC}$: for a role r, we can have $P(r) \in [\beta_1, \beta_2]$ to be made for roles, with semantics:

$$\forall x, y : P(r(x, y)) \in [\beta_1, \beta_2], \tag{2}$$

where again the probabilities are over the set of all interpretation mappings for a given domain.

Every ontology is assumed *acyclic*; that is, a concept does not use itself. If we write down an ontology as a directed graph where each node is a concept or role, and arcs go from concepts that are directly used to concepts that directly use them, we obtain that this graph must be acyclic. We refer to such a graph as an *ontology graph*. For instance, consider concepts A, B, C and the role r. C is a concept inclusion defined by $C \equiv A \sqcap \exists r.B$. In Figure 1.a we have the ontology graph for this example. Note that exists a node for general role $r\,(x, y)$ and another for the instantiation with concept $B\,(x)$, $\exists r.B\,(x)$. Concept inclusion $C\,(x)$ is composed by $A\,(x)$ and $\exists r.B\,(x)$.

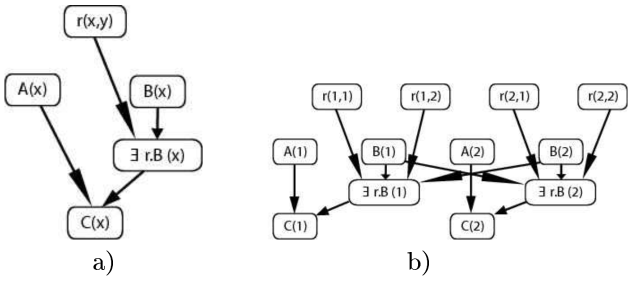


Fig. 1. Ontology graph

In short, the sentences written in the underlying description logic (with added probabilistic features) induce directed dependencies between instantiations of concepts. Under some additional restrictions (unique-names assumption, known and finite domain), any ontology expressed in CR$\mathcal{ALC}$ can be grounded into a Bayesian network, possibly with attached probability intervals [4,5,16]. That is, grounding an ontology with a finite and known domain leads to a *credal network* [6]. In Figure 1.b we have the grounded network for the ontology described in the previous paragraph, for a domain with only 2 individuals. Note that the entire ontology graph is repeated for each individual of the domain, with each concept instantiated for each individual and each role is instantiated with each pair of individuals. The probabilities of each sentence composes the CPT (*Conditional Probability Table*) of a particular node in the Bayesian network.

3 Semantic mapping with CR$\mathcal{ALC}$

We have used CR$\mathcal{ALC}$ previously to model some aspects of robotic semantic mapping. In [2] we proposed to segment robotic sensor data (odometry, gyro and distance measures) obtained from navigation through an indoor environment, based on the objects found in each different area. Rooms and Corridors are examples of possible areas to be found in an indoor environment. Such segmentation of the sensor data provides a scalable way to map larger environments, as each area could be mapped independently: as a result, several smaller areas are mapped and then merged together to construct the map.

The main limitation of that approach was that CR$\mathcal{ALC}$ models areas of the environment with relation to full objects detected by a image processing algorithm - inference does not start from sensor data itself. In our previous work, sensor data consisted of images that were processed by SIFT algorithm to detect objects whose signatures were trained previously.

But real robotic tasks must deal directly with uncertain sensor data. To do so, we wish to explore the flexibility and relatively low cost of CR$\mathcal{ALC}$; however, we do face some challenges to do so. In CR$\mathcal{ALC}$ we face a difficulty because the language models concepts (set of individuals) and a hierarchy over them, and not relations between individuals. There is no direct way to include a probabilistic dependency between two arbitrary constants or individuals. Some description logic languages that accepts *nominals*, allow us to specify individuals, like 'Brazil' and 'France'. But in semantic mapping domain it is impossible to consider in advance all the constants in the environment (all points, lines or planes that may exist).

The solution to this problem came from ideas presented in [19]. The trick is to include in the model, individuals or constants that are created from the combination of two segments; for example, there are two distinct segment lines, a and b. Then, if one is near the other, the constant ab is created (ba could also be created, but is identical to the first one). One way to specify in the model the conditional independences using the description logic language, is to create those kind of constants only when there is some dependency between the constants. Thus, it is not necessary to instatiate all possible combinations of segments. With that modification, it is necessary to differentiate concepts with

Table 1. Impact of features on the performance of classifier (extracted from [12]).

Environment	Lengths	Lengths+Neighbours	All
1	62.6%	88.5%	90.7%
2	58.7%	63.0%	93.5%
3	59.0%	79.2%	89.7%
4	51.8%	96.5%	97.7%
5	60.0%	68.5%	77.9%

primitive constant and concepts with composed constants. We now consider our application in more detail.

The scenario of interest is to take a bidimensional metric map, constructed using a SLAM algorithm, based on distances from a laser sensor, and to classificate each segment of the map in door or wall segments. Segments are extracted from laser data points following [8]. We do not propose to classify laser segments in real time as the robot constructs the map and localizes itself. Inferences are done offline, after the map has been obtained.

The trivial way to do that is to consider the length of each segment: doors tend to be of the same size, and walls have very variable lengths. But to make a robust classification, we need to consider further features of the segments. For instance, we should include dependences related to spatial relationships: points or line segments produced by laser sensor that are near one from another likely has the same classification.

To illustrate the importance of some features in the classification results, Table 1 lists the percentage of correct classification in five different environments, using only *Length*, *Length+Neighbours*, and all features together. As representative features are added (for instance spatial relationships), results are improved.

Some of the spatial features used by [23] are considered in our model and listed in Table 2.

Figure 2 depicts a Bayesian network constructed around two segments near each other and aligned along a line. Dashed line separates variables belonging to each of the segments. White nodes represents hidden variables; gray and black nodes represent observable variables; black nodes are continuous observable variables that must be discretized and gray nodes are discrete. Each segment is represented by *SegType* variable. Each has the *Length*, *Depth*, *SingleAligned* and *SharpTurn* properties. The relationships *Neighbours*, *Consecutive* and *Aligned* appears between each possible pair of segments. Beyond these properties and relationships, each segment could be attached to a line composed of aligned segments of the same type. In the scenario, only Wall objects could align to form a corridor. Each segment or aggregate of segments are represented by a discrete variable that contains its type (in the case of figure is *LineType*). *StarLine*, *EndLine*, *PreviousAligned*, *NextAligned* and *PartOf* characterize the properties of a segment inside a line.

This model, once implemented in CR$\mathcal{ALC}$, generates a large Bayesian network including all line segments extracted from the laser sensor, and considering all

Table 2. Spatial features.

SegType :	the segment is of the type, *Door*, *Wall*, or *Other*
LineType :	the line is of the type, *Door*, *Wall* or *Other*
PartOf :	the segment is part of the line
StartLine :	the segment is the start of a line
EndLine :	the segment is the end of a line
PreviousAligned :	there are segments aligned to this segment (preceding it)
NextAligned :	there are segments aligned to this segment (following it)
Aligned :	the angle between two segments is below some threshold, and so is the perpendicular distance between them
Neighbors :	the distance between the nearest end points of two segments is below some threshold
Consecutive :	there is no other segment's initial point between the initial points of the two segments
SingleAligned :	the angle between the segment and the average line it belongs to is below some threshold
SharpTurn :	the distance between the segment and its neighbor is below some threshold, and it is almost perpendicular to the average line
Length :	the length of the segment
Depth :	the depth of the segment, i.e., the signed perpendicular distance of the segment's midpoint to the nearest line

possible relationships between each two segments whose proximity is below some threshold. Classification is done through probabilistic inference in the graph using a MAP-based algorithm to promote collective classification. Recall that in collective classification, the class of each segment is decided based on the class of its neighbours.

An inherent problem in spatial mapping is the size of indoor environments. As each wall could be formed by a dozen line segments, the number of constants to be considered, and consequently the number of spatial relations to put in the model, are prohibitive. In our experiments, we have decided to partition the dataset in smaller sets, so we could handle the problem with the tools available. Figure 3 shows a corridor extracted from a map. The corridor is formed by a set of segments that must be classificated in doors, walls and others.

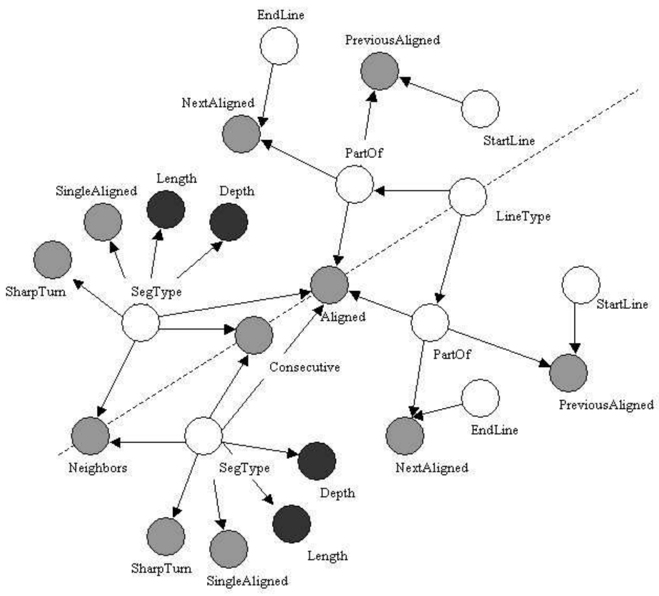


Fig. 2. A sample diagram.

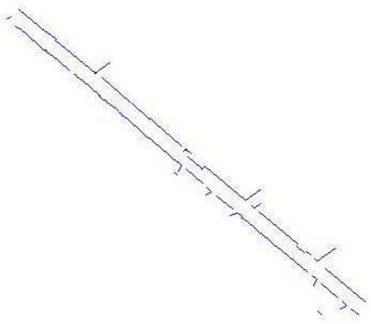


Fig. 3. Example of the scenario of classification of line segmentes.

4 Experiments

The experiments consist of teleoperation of a robot with a laser sensor through an indoor environment. As the robot navigates, laser readings and odometry are collected to be processed later, so as to produce a metric map. Any standard algorithm could be used to produce a consistent map. Basically, it is necessary to transform relative measures, obtained as the robot traverses the environment, into a global coordinate system, by dealing with uncertainty measures of the laser and the robot position. It is important to have line segments formed by laser points adequately positioned in the world, because CR$\mathcal{ALC}$ does not deal with uncertainty regarding spatial coordinates.

Although we collect some data with our own robot (Figure 4), and tested our model with it to determine the parameters, we have decided to report results for a dataset available online in the Radish repository, as other works that approached that same problem, using instead RMN and MLN models, used that dataset.

Fig. 4. Robot Pioneer 3-AT used in the experiments.

A restriction found in probabilistic models that incorporates logic elements is the type of random variables that are allowed. Often a continuous random variable for the length of the segment constructed from laser points must be discretized in a finite set of possible lengths. In our model, it was necessary to turn some numerical quantities into discrete values, as with variable *Length*. We considered six different values of lengths for doors and walls, based on our observed data.

The values for our conditional probability tables (the parameters of our model), were determined experimentally using our own experience in this kind of problem. These values are listed in Table 3.

Inferences were performed using the package *SamIam* (available at the address *http://reasoning.cs.ucla.edu/samiam/*). We selected MAP-based explanations generated by an approximate algorithm. Through MAP, we produced collective classification, and decided on each line segment label considering the labels of its neighbours.

Table 4 shows results obtained by MAP inference in a scenario with 70 line segments. In each column represents a range (i.e., 1-10 or 11-20) in the line segments considered. Rows indicate an exact line segment inside the range of the respective column. Observing the results, we have around of 75% accuracy, considering only *Length*, and *Neighboor* and *Aligned* features. It is hard to make a quantitative comparison between our results with Limketkai's RMN and Wang's MLN, because the features used in their experiments are not clearly given; nevertheless, qualitatively, results with CR$\mathcal{ALC}$ are similar to the ones obtained with their probabilistic logic models.

Table 3. CPTs used in the model.

	door	wall	other
length_1	0.0	0.2	0.25
length_2	0.1	0.15	0.15
length_3	0.8	0.15	0.2
length_4	0.1	0.15	0.15
length_5	0.0	0.15	0.25
length_6	0.0	0.2	0.0

door	0.3
wall	0.4
other	0.3

Roles

Neighbour									
s_1	door			wall			other		
s_2	door	wall	other	door	wall	other	door	wall	other
true	0.6	0.8	0.6	0.6	0.6	0.6	0.7	0.8	0.5
false	0.4	0.2	0.4	0.4	0.4	0.4	0.3	0.2	0.5

Aligned									
s_1	door			wall			other		
s_2	door	wall	other	door	wall	other	door	wall	other
true	0.5	0.3	0.3	0.3	0.5	0.3	0.3	0.3	0.5
false	0.5	0.7	0.7	0.7	0.5	0.7	0.7	0.7	0.5

Table 4. Inference results.

Calculated/Correct	1-10	11-20	21-30	31-40	41-50	51-60	61-70
1	**wall/wall**	**wall/wall**	wall/door	**other/other**	door/wall	**door/door**	**door/door**
2	other/wall	door/wall	**wall/wall**	wall/door	**door/door**	**wall/wall**	wall/other
3	**door/door**	wall/other	**wall/wall**	**wall/wall**	**wall/wall**	**wall/wall**	**door/door**
4	**wall/wall**	**wall/wall**	**wall/wall**	wall/other	**door/door**	wall/other	**wall/wall**
5	**wall/wall**	**wall/wall**	**wall/wall**	**door/door**	**wall/wall**	**wall/wall**	**wall/wall**
6	wall/other	**door/door**	**door/door**	door/wall	**door/door**	**door/door**	**wall/wall**
7	wall/other	**door/door**	door/wall	other/wall	**wall/wall**	**wall/wall**	**wall/wall**
8	**door/door**	other/wall	**door/door**	wall/other	**wall/wall**	**wall/wall**	**wall/wall**
9	**wall/wall**	**door/door**	**other/other**	**wall/wall**	**wall/wall**	**wall/wall**	**door/door**
10	**wall/wall**	wall/other	**wall/wall**	**door/door**	**door/door**	**wall/wall**	wall/other

5 Conclusion

This article proposes semantic mapping techniques based on the classification of line segments from a metric map into *Doors*, *Walls*, or *Others* elements, using CR$\mathcal{ALC}$ as a representation language. Metric maps are constructed by a standard SLAM algorithm, so as to obtain a precise spatial positioning of each line segment and then to determine features. To do so, we used constants formed by the combination of two simple constants. With these new constants, we included features of neighborhoods and properties of alignments.

We chose a probabilistic description logic due to its compact encoding of the needed knowledge; as a result less parameters must be specified. Collective classification proceeds as inference over an instantiated probabilistic graph using approximate reasoning; all labels are decided together in a single run.

Preliminary results obtained with CR$\mathcal{ALC}$ show that it can handle classification of robotic sensor data. The next step is to further extend this labeling to create an automatic topological map starting for the labels of the metric map, and also to use the same technique to create 3d maps. Besides that, we are trying to introduce DL reasoning in the model through extension of $\mathcal{ALC}$ to some description logic that accepts spatial reasoning.

References

1. D. Anguelov, Ben Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. NG. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
2. Fabiano Corrêa, Rodrigo B. Polastro, Fabio G. Cozman, and Jun Okamoto Jr. Dealing with semantic knowledge in robotics with a probabilistic description logic. In *Argentine Symposium on Artificial Intelligence*, 2010.
3. Paulo C. G. Costa and Kathryn B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Conference on Formal Ontology in Information Systems*, 2006.
4. Fabio G. Cozman and Rodrigo B. Polastro. Loopy propagation in a probabilistic description logic. In Sergio Greco and Thomas Lukasiewicz, editors, *Second International Conference on Scalable Uncertainty Management*, Lecture Notes in Artificial Intelligence (LNAI 5291), pages 120–133. Springer, 2008.
5. Fabio G. Cozman and Rodrigo B. Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.
6. C. P. de Campos and Fabio G. Cozman. Computing lower and upper expectations under epistemic independence. *International Journal of Approximate Reasoning*, 44:244–260, 2007.
7. Zhongli Ding, Yun Peng, and Rong Pan. BayesOWL: Uncertainty modeling in semantic web ontologies. In *Soft Computing in Ontologies and Semantic Web*, volume 204 of *Studies in Fuzziness and Soft Computing*, pages 3–29. Springer, Berlin/Heidelberg, 2006.
8. J. S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1999.
9. J. Heinsohn. Probabilistic description logics. In *Conference on Uncertainty in Artificial Intelligence*, pages 311–318, 1994.
10. Manfred Jaeger. Probabilistic reasoning in terminological logics. In *Principles of Knowledge Representation (KR)*, pages 461–472, 1994.
11. Daphne Koller, Alon Y. Levy, and Avi Pfeffer. P-CLASSIC: A tractable probablistic description logic. In *AAAI*, pages 390–397, 1997.
12. B. Limketkai, L. Liao, and Dieter Fox. Relational object maps for mobile robots. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1:1471–1476, 2005.

13. Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172:852–883, 2008.
14. Thomas Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.
15. A. Nuchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann. 3D mapping with semantic knowledge. In *RoboCup Internation Symposium*, 2005.
16. Rodrigo B. Polastro and Fabio G. Cozman. Inference in probabilistic ontologies with attributive concept descriptions and nominals. In *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 7th International Semantic Web Conference (ISWC)*, Karlsruhe, Germany, 2008.
17. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.
18. Fabrizio Sebastiani. A probabilistic terminological logic for modelling information retrieval. In W.B. Croft and C.J. van Rijsbergen, editors, *17th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 122–130, Dublin, Ireland, 1994. Springer-Verlag.
19. Sheng sheng Wang and Da you Liu. Spatial description logic and its application in geospatial semantic web. In *International Multisymposiums on Computer and Computacional Sciences*, 2008.
20. Sebastian Thrun. Robotic mapping: A survey. Technical report, Carnegie Mellow University, Computer Science Department, 2002.
21. Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, and Pascal Stang. Stanley: the robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.
22. Rudolph Triebel, Kristian Kersting, and Wolfram Burgard. Robust 3D scan point classification using associative Markov networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2603–2608, 2006.
23. Jue Wang and Pedro Domingos. Hybrid Markov logic networks. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 2, pages 1106–1111, 2008.
24. Zoran Zivkovic, Olaf Booij, and Ben Kröse. From images to rooms. *Robotics and Autonomous Systems*, 55:411–418, 2007.

The Modular Structure of an Ontology: Atomic Decomposition towards Applications

Chiara Del Vescovo

The University of Manchester, Oxford Road, Manchester, M13 9PL, UK
delvescc@cs.man.ac.uk

1 Introduction

Modularity in ontologies Modern ontologies can get quite large as well as complex, which poses challenges to tools and users when it comes to processing, editing, analyzing them, or reusing their parts. This suggests that exploiting modularity of ontologies might be fruitful, and research into this topic has been an active area for ontology engineering. Much recent effort has gone into developing *logically sensible* modules, that is, parts of an ontology which offer strong logical guarantees for intuitive modular properties. One such guarantee is called *coverage*. It means that a module captures all the ontology's knowledge about a given set of terms (signature). A module in this sense is a subset of an ontology's axioms that provides coverage for a signature, and each possible signature determines such a module. The minimal modules to provide coverage for a signature are those based on Conservative Extensions (CEs) [2], that are however not feasible to be computed for many expressive languages. Modules based on syntactic locality [5] also provide coverage because they are efficiently computable approximations of CEs; however, such modules are not in general minimal.

The extraction of such a module given a set of terms (*signature*) is well understood and starting to be deployed in standard ontology development environments, such as Protégé 4,[1] and online.[2] Locality-based modules have already been effectively used for ontology reuse [14] and as a subservice for incremental reasoning [3]. However, we think that by investigating the family $\mathfrak{F}_{\mathcal{O}}$ of all locality-based modules we can obtain information about topicality, connectedness, structure, superfluous parts of an ontology, or agreement between actual and intended modeling.

Previous work In [6] we investigated the number of (locality-based) modules that an ontology can have. There are examples of artificial ontologies with exponentially many w.r.t. their size: for example, the ontology $\mathcal{O} = \{\mathtt{A}_i \sqsubseteq \mathtt{B} \,|\, i = 1, \ldots, n\}$, where each subset of the ontology is a module. However, we tried to understand if real ontologies generate an exponential family $\mathfrak{F}_{\mathcal{O}}$ of modules. To this aim, we selected some ontologies from the TONES ontology repository[3] and tried to extract all of their modules. The results we obtained made us tend towards rejecting the hypothesis, but they were not strong enough for a clear rejection.

[1] http://www.co-ode.org/downloads/protege-x
[2] http://owl.cs.manchester.ac.uk/modularity
[3] http://owl.cs.manchester.ac.uk/repository/browser

In [7] we introduced a different approach to look at modules: we noticed that an ontology can be partitioned into building blocks, called *atoms*, that do not split over two modules. Atoms are interesting because they are in 1-1 correspondence with *genuine modules*, i.e., modules that are not the union of two uncomparable (w.r.t. set inclusion) modules. Moreover, they are computable in polynomial time (provided that the extraction of a locality-based module is polynomial). The set of atoms is called *Atomic Decomposition* (AD). The AD of an ontology is stable, in the sense that is well-defined given any (suitable) notion of locality-based module. This suggests us that we could use it for performing several tasks of interest for users of ontologies, as the estimation of the number of modules of an ontology. We started exploiting this matter in [8].

Following [7], in this paper we describe more extensively the AD of an ontology: its definition, its properties, its generation. We then introduce some tasks of interest for ontology engineers that could be performed by means of the ADs. For each such task, we discuss with the help of several examples some issues about the suitability of the sole AD to perform it. Hence, we introduce a family of refinements of AD, called *Labelled Atomic Decomposition* (LAD), useful to solve the issues raised. In particular, these issues are deeply discussed w.r.t. one such specific task: the *Fast Module Extraction* (FEM), t.i., the extraction of a single module without loading the ontology.

2 Preliminaries

We assume the reader to be familiar with Description Logics [1], and only briefly sketch here some of the central notions around locality-based modularity. We use $\mathcal{L}$ for a Description Logic, e.g., $\mathcal{SHIQ}$, and $\mathcal{O}, \mathcal{M}$, etc., for a knowledge base, i.e., a finite set of axioms. Moreover, we respectively use $\widetilde{\alpha}$ or $\widetilde{\mathcal{O}}$ for the signature of an axiom α or of an ontology $\mathcal{O}$, i.e., the set of concept, role, and individual names used in α or in $\mathcal{O}$.

Conservative extensions (CEs) capture the above described encapsulation of knowledge. They are defined as follows.

Definition 1. *Let $\mathcal{L}$ be a DL, $\mathcal{M} \subseteq \mathcal{O}$ be $\mathcal{L}$-ontologies, and Σ be a signature.*

1. *$\mathcal{O}$ is a* deductive Σ-conservative extension (Σ-dCE) *of $\mathcal{M}$ w.r.t. $\mathcal{L}$ if for all axioms α over $\mathcal{L}$ with $\widetilde{\alpha} \subseteq \Sigma$, it holds that $\mathcal{M} \models \alpha$ if and only if $\mathcal{O} \models \alpha$.*
2. *$\mathcal{M}$ is a* dCE-based module for Σ *of $\mathcal{O}$ if $\mathcal{O}$ is a Σ-dCE of $\mathcal{M}$ w.r.t. $\mathcal{L}$.*

Unfortunately, CEs are hard or even impossible to decide for many DLs [11,15,17]. Therefore, approximations have been devised. We focus on *syntactic locality* (here for short: locality). Given an ontology $\mathcal{O}$, and a set of terms, called *seed signature*, $\Sigma \subseteq \widetilde{\mathcal{O}}$, we say that an axiom $\alpha \in \mathcal{O}$ is $\bot$*-local* w.r.t. Σ if we can clearly identify it as a tautology when all the terms not in Σ are substituted by $\bot$ (the formal definition can be found in [5]). An analogous definition can be made for $\top$-locality. Then, a locality-based module is recursively computed as follows: starting from an empty set $\mathcal{M}$, each axiom $\alpha \in \mathcal{O}$ is tested whether it is local w.r.t. Σ; if not, α is added to $\mathcal{M}$, the signature Σ is extended with all terms in $\widetilde{\alpha}$, and the test is re-run against the extended signature. Then, $\mathcal{M} \subseteq \mathcal{O}$ and all axioms in $\mathcal{O} \setminus \mathcal{M}$ being local w.r.t. $\Sigma \cup \widetilde{M}$ is sufficient for $\mathcal{O}$ to be a Σ-dCE of $\mathcal{M}$. By alternating the extraction of $\bot$- and $\top$-module over the previously extracted module, we obtain various notion of modules. When the fixpoint is reached, the resulting notion is called $\top\bot^*$*-locality*. Locality-based modules can be efficiently computed and provide coverage; that is, they capture *all* the relevant entailments, but not necessarily *only* those [5,13]. A module extractor is implemented in the OWL API.[4]

Given a module notion $x \in \{\top, \bot, \top\bot^*\}$, we denote by x-mod$(\Sigma, \mathcal{O})$ the x-module of $\mathcal{O}$ w.r.t. Σ. The following properties of locality-based modules will be of interest for our modularization [5,17].

Proposition 2. *Let $\mathcal{O}$ be an ontology, Σ a signature and $x \in \{\top, \bot, \top\bot^*\}$. Then the following properties hold:*
(a) for any Σ', x-mod$(\Sigma, \mathcal{O}) \subseteq x$-mod$(\Sigma \cup \Sigma', \mathcal{O})$ (monotonicity)
(b) for Σ' with $\Sigma \subseteq \Sigma' \subseteq \Sigma \cup \widetilde{\mathcal{M}}$, x-mod$(\Sigma', \mathcal{O}) = x$-mod$(\Sigma, \mathcal{O})$ (self-containedness)
(c) each axiom α entailed by $\mathcal{O} \setminus x$-mod$(\Sigma, \mathcal{O})$ and such that $\widetilde{\alpha} \subseteq \Sigma$ is a tautology (depletingness).

Proposition 3. *Any notion of locality-based modules satisfying the properties in Prop. 2 is such that any given signature generates a unique module.*

From now on, we focus on the $\top\bot^*$ notion of locality-based modules. However, we want to underline that what we discuss in the rest of the paper can be carried out for each notion of module satisfying monotonicity, self-containedness, and depletingness.

3 Atomic Decomposition

In [7] we introduced a new approach to represent the whole family $\mathfrak{F}_{\mathcal{O}}$ of locality-based modules of an ontology $\mathcal{O}$. The key point is observing that some axioms appear in a module only if other axioms do. In this spirit, we defined a notion of "logical dependence" between axioms: the idea is that an axiom α depends on another axiom β if whenever α occurs in a module $\mathcal{M}$ then β also belongs to $\mathcal{M}$.

To keep the formalization clean, we remove from the ontology *syntactic tautologies*, i.e. always-local axioms, and *global axioms*, i.e. axioms that belong to all modules. We can always remove these unwanted axioms and consider them separately. Then, for each axiom α is well-defined the *smallest* module containing it.

Proposition 4. *The module $\top\bot^*$-mod$(\widetilde{\alpha}, \mathcal{O})$ is the smallest containing α.*

Proof. We recall that $\top\bot^*$-mod satisfies the properties as in Prop. 2. Then:

1. α is non-local w.r.t. $\widetilde{\alpha}$ (because is not a syntactic tautology), hence $\mathcal{M}_\alpha$ is not empty
2. $\mathcal{M}_\alpha$ is the unique and thus smallest module for the seed signature $\widetilde{\alpha}$
3. by monotonicity, enlarging the seed signature $\widetilde{\alpha}$ results in a superset of $\mathcal{M}_\alpha$
4. $\mathcal{M}' = \top\bot^*$-mod$(\widetilde{\mathcal{M}'}, \mathcal{O}) = \top\bot^*$-mod$(\widetilde{\mathcal{M}'} \cup \widetilde{\alpha}, \mathcal{O}) \supseteq \top\bot^*$-mod$(\widetilde{\alpha}, \mathcal{O})$ by self-containedness and monotonicity, thus any module $\mathcal{M}'$ that contains α needs to contain also $\mathcal{M}_\alpha$. □

Definition 5. *The module $\mathcal{M}_\alpha = \top\bot^*$-mod$(\widetilde{\alpha}, \mathcal{O})$ as in Prop. 4 is called* α-module.

[4] `http://owlapi.sourceforge.net/`

The dependency between axioms allows us to identify clumps of highly interrelated axioms that never split over two or more modules [7]; these clumps are called *atoms*.

Definition 6. *An* atom *is a maximal disjoint subset of an ontology such that their axioms either appear always together in modules, or none of them does.*

Definition 7. *The family of atoms of an ontology $\mathcal{O}$ is denoted by $\mathcal{A}(\mathfrak{F}_{\mathcal{O}})$ and is called* Atomic Decomposition (AD).

The AD is evidently a partition of the ontology, thus is linear w.r.t. the size of the ontology. Moreover, atoms are the building blocks of all modules [10].

Proposition 8. *Each module is the union of suitable atoms.*

We summarize in the following table the ontologies' fragments described so far.

Structure	$\mathcal{O}$	$\mathfrak{F}_{\mathcal{O}}$	$\mathcal{A}(\mathfrak{F}_{\mathcal{O}})$
Elements	axioms α	modules $\mathcal{M}$	atoms $\mathfrak{a}, \mathfrak{b}, \ldots$
Maximal size	baseline	exponential	linear
Mathem. object	set	family of sets	poset

Proposition 9. *Let $\mathfrak{a}$ be an atom in the AD $\mathcal{A}(\mathfrak{F}_{\mathcal{O}})$ of an ontology $\mathcal{O}$; then, for any selection of axioms $\mathcal{S} = \{\alpha_1, \ldots, \alpha_\kappa\} \subseteq \mathfrak{a}$ we have that $\top\bot^*$-mod$(\widetilde{\mathcal{S}}, \mathcal{O}) = \mathcal{M}_\alpha$. In particular, for each $\alpha_i \in \mathfrak{a}$, $\mathcal{M}_{\alpha_i} = \mathcal{M}_\alpha$.*

Proof. Let $\alpha \in \mathfrak{a}$ be an axiom, and let us consider the module $\mathcal{M}_\alpha$. Then:

1. $\mathfrak{a} \subseteq \mathcal{M}_\alpha$ by the definition of atoms
2. as a consequence, $\mathcal{M}_\alpha \supseteq \top\bot^*$-mod$(\widetilde{\mathcal{S}}, \mathcal{O})$ for every selection $\mathcal{S}$ of axioms from $\mathfrak{a}$
3. by Prop. 4, the inverted inclusion $\top\bot^*$-mod$(\widetilde{\alpha}_i, \mathcal{O}) \supseteq \mathcal{M}_\alpha$ also holds. □

A module $\mathcal{M}_{\mathfrak{a}} = \top\bot^*$-mod$(\widetilde{\mathfrak{a}}, \mathcal{O})$ is called *compact*. From Prop. 9 it is clear that the set of compact modules coincides with the one of α- modules. Hence, we can denote by $\mathcal{M}_{\mathfrak{a}}$ the module $\mathcal{M}_\alpha$ for each $\alpha \in \mathfrak{a}$. Now, we are ready to extend the definition of "logical dependency" to atoms.

Definition 10. *Let $\mathfrak{a}$ and $\mathfrak{b}$ be two distinct atoms of an ontology $\mathcal{O}$. Then:*

- $\mathfrak{a}$ *is* dependent *on* $\mathfrak{b}$ *(written $\mathfrak{a} \succeq \mathfrak{b}$) if $\mathcal{M}_{\mathfrak{b}} \subseteq \mathcal{M}_{\mathfrak{a}}$*
- $\mathfrak{a}$ *and* $\mathfrak{b}$ *are* independent *if $\mathcal{M}_{\mathfrak{a}} \cap \mathcal{M}_{\mathfrak{b}} = \emptyset$*
- $\mathfrak{a}$ *and* $\mathfrak{b}$ *are* weakly dependent *if, they are neither independent nor dependent; in this case, there exists an atom $\mathfrak{c}$ which both $\mathfrak{a}$ and $\mathfrak{b}$ are dependent on.*

Thanks to Def. 10, the AD inherites the mathematical structure of partially ordered set, thus can be represented by means of a Hasse diagram.

Proposition 11. *The binary relation " $\succeq$" as in Def. 10 is a partial order over the set $\mathcal{A}(\mathfrak{F}_{\mathcal{O}})$ of atoms of an ontology $\mathcal{O}$.*

Proof. This is true because $\succeq$ satisfies reflexivity, antisymmetry, and transitivity. □

Algorithm 1 Atomic decomposition

```
Input: An ontology $\mathcal{O}$.
Output: The set $\mathfrak{G}$ of $\alpha$-modules; the poset of atoms $(\mathcal{A}(\mathfrak{F}_\mathcal{O}), \succeq)$; the set of generating axioms GenAxs; for $\alpha \in$ GenAxs, the cardinality CardAtom($\alpha$) of its atom.

ToDoAxs $\leftarrow \top\bot^*$-mod$(\widetilde{\mathcal{O}}, \mathcal{O}) \setminus \top\bot^*$-mod$(\emptyset, \mathcal{O})$
GenAxs $\leftarrow \emptyset$
for each $\alpha \in$ ToDoAxs do
    Module($\alpha$) $\leftarrow \top\bot^*$-mod$(\widetilde{\alpha}, \mathcal{O})$        $\{\neq \emptyset\}$
    $new \leftarrow true$
    for each $\beta \in$ GenAxs do
        if Module($\alpha$) = Module($\beta$) then
            Atom($\beta$) $\leftarrow$ Atom($\beta$) $\cup \{\alpha\}$
            CardAtom($\beta$) $\leftarrow$ CardAtom($\beta$) + 1
            $new \leftarrow false$
        end if
    end for
    if $new = true$ then
        Atom($\alpha$) $\leftarrow \{\alpha\}$
        CardAtom($\alpha$) $\leftarrow 1$
        GenAxs $\leftarrow$ GenAxs $\cup \{\alpha\}$
    end if
end for
for each $\alpha \in$ GenAxs do
    for each $\beta \in$ GenAxs do
        if $\beta \in$ Module($\alpha$) then
            Atom($\beta$) $\preceq$ Atom($\alpha$)
        end if
    end for
end for
$\mathcal{A}(\mathfrak{F}_\mathcal{O}) \leftarrow \{$Atom($\alpha$) $\mid \alpha \in$ GenAxs$\}$
$\mathfrak{G} \leftarrow \{$Module($\alpha$) $\mid \alpha \in$ GenAxs$\}$
return $[(\mathcal{A}(\mathfrak{F}_\mathcal{O}), \succeq)$,$\mathfrak{G}$,GenAxs,CardAtom($\cdot$)]
```

Name	#logical axioms	#α-mods	#Con. comp.	#max. mod.	#max. atom
Koala	42	23	5	18	7
Mereology	44	17	2	11	4
University	52	31	11	20	11
People	108	26	1	77	77
miniTambis	173	129	85	16	8
OWL-S	277	114	1	57	38
Tambis	595	369	119	236	61
Galen	4,528	3,340	807	458	29

Table 1. Experiments summary

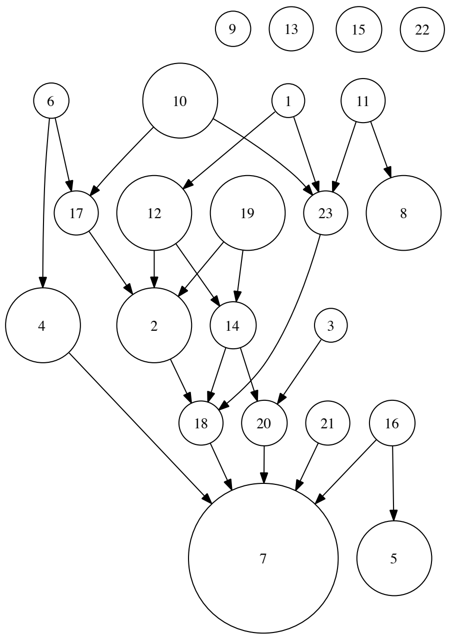


Figure 1. The AD of Koala

Prop. 9 and Prop. 11 provide the basis for our polynomial algorithm for the computation of the AD since it allows us to construct $\mathcal{A}(\mathfrak{F}_\mathcal{O})$ via α-modules only. The whole procedure is described in Alg. 1. A proof for its correctness can be found in [10].

We ran Algorithm 1 on a selection of ontologies, including those used in [6], and indeed managed to compute the AD in all cases, even for ontologies where a complete modularization was previously impossible. Table 1 summarizes ontology data: size, expressivity, number of compact modules (= number of atoms), number of connected components in the AD poset, size of largest compact module and of largest atom. Our tests were obtained on a 2.16 GHz Intel Core 2 Duo Macbook with 2 GB of memory running Mac OS X 10.5.8; each AD was computed within a couple of seconds (resp. 3 minutes for Galen).

We have also generated a graphical representation using GraphViz[5]. Our ADs show atom size as node size, see e.g. Fig. 1. It shows four isolated atoms, e.g., Atom 22, consisting of the axiom `DryEucalyptForest` $\sqsubseteq$ `Forest`. This means that, although other modules may use some (but not all) 22's terms, they do not "need" 22's axioms for any entailment. Hence, removing (the axioms in) isolated atoms from the ontology would not result in the loss of any entailments regarding other modules or terms. Of course, for entailments involving both `DryEucalyptForest` and `Forest` and possibly other terms, axioms in isolated atoms may be needed. A similar structure is observable in all ontologies considered, see the graphs at `http://bit.ly/i4olY0` .

The following results have a deep impact on the way we describe modules in ADs: the poset structure of an AD is a 1-1 representation of compact modules.

Definition 12. *The* principal ideal of an atom $\mathfrak{a}$ *is the set* $(\mathfrak{a}] = \{\alpha \in \mathfrak{b} \mid \mathfrak{b} \preceq \mathfrak{a}\} \subseteq \mathcal{O}$.

Lemma 13. *Principal ideals of atoms are modules.*

Proof. Given an atom $\mathfrak{a} \in \mathcal{A}(\mathfrak{F}_{\mathcal{O}})$, we want to compare its principal ideal $(\mathfrak{a}] = \bigcup_{\mathfrak{b} \preceq \mathfrak{a}} \mathfrak{b}$ with the module $\mathcal{M}_\alpha$. By the definition of atoms, $\mathcal{M}_\alpha \supseteq (\mathfrak{a}]$. We still need to prove that the equality holds. By contraposition, let $\mathcal{M}_\alpha$ be a proper superset of $(\mathfrak{a}]$. Then it contains at least one atom $\mathfrak{b}$ which $\mathfrak{a}$ is not dependent on. Let β be an axiom in $\mathfrak{b}$, and let us consider $\mathcal{M}_\beta$. By Prop. 4, $\mathcal{M}_\beta$ is the smallest module containing $\mathfrak{b}$. Then, $\mathcal{M}_\beta$ is contained in $\mathcal{M}_\alpha$, and since the latter is the smallest module containing $\mathfrak{a}$, this means that $\mathfrak{a}$ is dependent on $\mathfrak{b}$. This last fact contradicts the assumption. □

Prop. 9 implies that two axioms from the same atom generate the same compact module. The converse also holds.

Proposition 14. *Let* α,β *be two axioms such that* $\mathcal{M}_\alpha = \mathcal{M}_\beta$. *Then, an atom* $\mathfrak{a}$ *exists such that* $\alpha,\beta \in \mathfrak{a}$.

Proof. By contraposition, let $\mathfrak{a}$ and $\mathfrak{b}$ be two distinct (hence, disjoint) atoms such that $\alpha \in \mathfrak{a}$ and $\beta \in \mathfrak{b}$. Then, by Prop. 13 the principal ideals $(\mathfrak{a}]$ and $(\mathfrak{b}]$ are also distinct modules, and this contradicts the hypothesis. □

Another interesting property is the existence of a mapping, denoted by $r_{\mathcal{O}}$, between the family $\mathfrak{F}_{\mathcal{O}}$ of modules of the ontology $\mathcal{O}$ into the set of antichains of the poset structure of the AD, such that if $\mathcal{M} = \mathfrak{a}_1 \cup \ldots \cup \mathfrak{a}_n$, then $r_{\mathcal{O}}(\mathcal{M})$ is the minimum set of atoms such that $\mathcal{M} = (\mathfrak{a}_1] \cup \ldots \cup (\mathfrak{a}_\kappa]$. In particular, $\{\mathfrak{a}_1, \ldots, \mathfrak{a}_\kappa\}$ is a set of uncomparable atoms. Unfortunately, this mapping is not 1-1.

Corollary 15. *For each module* $\mathcal{M}$ *of an ontology* $\mathcal{O}$, *there are uncomparable (w.r.t. the dependency relation* $\preceq$*) atoms* $\mathfrak{a}_1, \ldots, \mathfrak{a}_\kappa$ *such that* $\mathcal{M} = \bigcup_{i=1}^{n} (\mathfrak{a}_i]$.

Proof. By the definition of atoms, if $\mathcal{M}$ contains one axiom from an atom $\mathfrak{a}_i$, then it contains all its axioms. By the definition of dependency, if $\mathcal{M}$ contains one atom $\mathfrak{a}_i$, then it contains all the atoms that $\mathfrak{a}_i$ depends on. Finally, we can consider only uncomparable atoms because in case $\mathcal{M}$ contains an $\mathfrak{b}$ such that $\mathfrak{a}_i \succeq \mathfrak{b}$, then $\mathfrak{b}$ is already included in the representation of $\mathcal{M}$ as $(\mathfrak{a}_1] \cup \ldots \cup (\mathfrak{a}_i] \cup \ldots \cup (\mathfrak{a}_\kappa]$. □

[5] `http://www.graphviz.org/About.php`

3.1 Genuine modules

Another notion of module that we want to describe consists of those that do not fall apart into more than one piece, and hence have a strong internal coherence.

Definition 16. *A module is called* fake *if there exist two uncomparable (w.r.t. $\subseteq$) modules $\mathcal{M}_1, \mathcal{M}_2$ with $\mathcal{M}_1 \cup \mathcal{M}_2 = \mathcal{M}$; a module is called* genuine *if it is not fake.*

Lemma 17. *The notions of α- and genuine modules coincide.*

Proof. Both directions are proven by contraposition.

α- $\Rightarrow$ genuine : Let $\mathcal{M}$ be a fake module. Then there are two uncomparable modules $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$. In particular, there exist suitable atoms such that $\mathcal{M}_1 = \mathfrak{a}_1 \cup \ldots \cup \mathfrak{a}_\kappa$ and $\mathcal{M}_2 = \mathfrak{b}_1 \cup \ldots \cup \mathfrak{b}_\ell$. Since the modules are uncomparable, then there is at least one atom $\mathfrak{a}_k$ in $\mathcal{M}_1$ such that $\mathfrak{a}_k \not\in \{ \mathfrak{b}_1, \ldots, \mathfrak{b}_\ell\}$; similarly, there is at least one atom $\mathfrak{b}_l$ in $\mathcal{M}_2$ such that $\mathfrak{b}_k \not\in \{ \mathfrak{a}_1, \ldots, \mathfrak{a}_\kappa\}$. Finally, there is no atom $\mathfrak{c} \in \mathcal{M} = \{\mathfrak{a}_1, \ldots, \mathfrak{a}_\kappa, \mathfrak{b}_1, \ldots, \mathfrak{b}_\ell\}$ dependent both on $\mathfrak{a}_k$ and on $\mathfrak{b}_l$, otherwise these atoms would be both in $\mathcal{M}_1$ and in $\mathcal{M}_2$; hence, $\mathcal{M}$ is not compact.

genuine $\Rightarrow$ α- : Let $\mathcal{M}$ be a non compact module. By Cor. 15, there exist atoms $\mathfrak{a}_1, \ldots, \mathfrak{a}_\kappa$ such that $\mathcal{M} = (\mathfrak{a}_1] \cup \ldots \cup (\mathfrak{a}_\kappa]$, with $\kappa \geq 2$. By Lemma 13 we have that the principal ideal of every atom is a module. Hence $\mathcal{M} = (\mathfrak{a}_1] \cup \ldots \cup (\mathfrak{a}_\kappa]$ is the union of uncomparable modules, and more in specific, fake. □

A straightforward consequence of Cor. 15 and Lemma 17 is the set of genuine modules to be a base for all locality-based modules: more precisely, each module is the union of a combination of genuine modules. However, the converse does not hold: not all combinations are modules, and given an AD is non-trivial to determine which combinations of genuine modules generate a module.

Example 18. Let us consider the ontology $\mathcal{O} = \{\mathtt{A}_i \sqsubseteq \mathtt{A}_{i+1} \,|\, i = 0, \ldots, n-1\}$. Then, the AD of this ontology consists of n atoms pairwise independent. However, for each choice of two terms $\mathtt{A}_\kappa, \mathtt{A}_\ell$ with $\kappa < \ell$, the module for the seed signature $\Sigma = \{\mathtt{A}_\kappa, \mathtt{A}_\ell\}$ is the set $\top\bot^*(\Sigma, \mathcal{O}) = \{\mathtt{A}_\kappa \sqsubseteq \mathtt{A}_{\kappa+1}, \ldots, \mathtt{A}_{\ell-1} \sqsubseteq \mathtt{A}_\ell\}$. In other words, the atoms concerning the terms "between" $\mathtt{A}_\kappa$ and $\mathtt{A}_\ell$ are not really independent, because they are "pulled into" the module for Σ.

The reason for this to happen can be found in the overlapping of minimal seed signatures for genuine modules. We have seen in Sect.2 how modules are extracted, and how the seed signature is "enlarged" to include the signature of all non-local axioms. Hence, if the extended signature overlaps with the minimal seed signature of a different genuine module $\mathcal{M}'$, then $\mathcal{M}'$ is pulled into the module extracted.

4 Towards Applications

4.1 What for?

Fast Module Extraction (FME) : Ontologies are sometimes difficult even to load, so an interesting task to perform would be the off-line extraction of modules by using the

AD of an ontology; in practice, we want to be able to recognize which combinations of atoms generate a module, that is, find the inverse of the representing mapping $r_{\mathcal{O}}$. As briefly introduced in Ex. 18, this operation does not directly follow from the AD: we need more information concerning the minimal seed signatures of genuine modules, because their overlapping can cause other atoms to fall into the module we are extracting. Further in this section we are going to discuss some preliminary issues about FME.

Module Count (MC) : In [6] we tried to compute a full modularization for the ontologies of different size listed in Table 1 in order to test the hypothesis that the number of modules does not grow exponentially with the size of the ontology. Unfortunately, we managed to compute all modules for two ontologies only, namely Koala and Mereology. For the others, we sampled subontologies and extracted all of their modules. The results we obtained made us tend towards rejecting the hypothesis, but they were not strong enough for a clear rejection. From Cor. 15 we derive that one plausible application of ADs is an estimate of the number of modules of an ontology, as a first approximation by counting the number of antichains of the AD poset. However, this approach has been proven unsuccessful: the estimate is still too large, because not all antichains generate a module, as in Ex.18. So the problem remains open, and only preliminary though encouraging results are reported in [8].

Topicality for Ontology Comprehension (TOC) : The AD of an ontology derives from strong logical properties of locality-based modules, so we expect it to preserve, or indeed reveal, these properties. The first observation that we want to point out is that, given an ontology and a notion of module, the structure defined in its AD is uniquely determined. The stability of this structure implies that the issues described in what follows are well-defined. Since modules are defined as set of axioms providing coverage to a given set of terms Σ, it is natural to investigate the relations between terms and modules.

In [9] we have exploited different notions of topicality in ontologies (and, more in general, for logic-based theories) for notion of modules with strong logical properties. Clearly, a notion as AD is too loose to define topicality for ontologies, since the sole structure does not explain what the ontology is about. A refined suitable version of AD would also contain labels to describe the content of an atom. Preliminary results in this sense are reported in [9].

Beside the tasks described so far that we started addressing, we identified at least other two tasks of interest, that we briefly describe in what follows.

Suggesting axioms to Repair First (RF) : One task that ontology engineers perform commonly is maintaining and repairing ontologies. Interesting tools for this task make use of *Justifications* [12]. Justifications are minimal sets of axioms that explain why a specific entailment holds. Ontology engineers often search justifications for classes to be unsatisfiable. Unfortunately, justifications can be large, and numerous. The logical dependency of axioms defined in AD could be used to suggest which axioms of a justification to repair first, and in particular, those that the other axioms depend on. The hope is that mistakes in the modeling phase propagate within the ontology by means of the logical dependency as defined here.

Suggesting Seed Signatures (SigSug) : The users of ontologies are often interested in extracting a (possibly minimal) set of axioms that "know everything" about a specific set of terms. However, locality-based modules are designed to provide coverage for a given seed signature Σ. Even if related to what required, modules are often too small, because users are interested also in the relation that a term has with some of its sub- or super-classes, or sub- or super-roles. However, there is no trivial relation between the seed signature of input and the signature of the module extracted, so these relations are sometimes left out the module. The current solution for this problem is the extraction of a module for an enlarged signature, but the AD could be of interest for refining this approach.

Throughout the description of the various tasks, we mentioned that often the AD is too loose w.r.t. the actual modular structure of the ontology, hence adding information can be of help in real applications. One possible refinement of ADs consists of including information about seed signatures in the AD: the result is called *Labelled Atomic Decomposition* (LAD).

Definition 19. *Given: an ontology $\mathcal{O}$ and its AD $\mathcal{A}(\mathfrak{F}_{\mathcal{O}}) = \{\mathfrak{a}_1, \ldots, \mathfrak{a}_n\}$, a* labelling function $\mathsf{Lab}(.)$ *is a function from $\mathcal{A}(\mathfrak{F}_{\mathcal{O}})$ to the power set of $\widetilde{\mathcal{O}}$, that matches each atom with a suitable set of terms from the ontology.*

The information to be added depends on the task the users want to perform. Throughout the rest of this section we briefly discuss the suitability of a specific labelling function to perform FME directly from the LAD of an ontology.

4.2 LAD for Fast Module Extraction

Let us consider the labelling function $\mathsf{Lab}_{\text{ssig}}$ such that to each atom $\mathfrak{a}$ is assigned the set of minimal seed signatures that generate the module $\mathcal{M} = (\mathfrak{a}]$. This labelling is useful to discover "hidden relations" between an atom and terms that do not occur in it.

Example 20. Let us consider the ontology $\mathcal{O} = \{\mathtt{A} \equiv \mathtt{B}, \mathtt{B} \sqsubseteq \mathtt{C}, \mathtt{B} \sqcap \mathtt{D} \sqsubseteq \mathtt{C} \sqcup \mathtt{E} \sqcup (\mathtt{G} \sqcup \neg\mathtt{G}), \mathtt{D} \sqsubseteq \mathtt{E}, \mathtt{E} \equiv \mathtt{F}\}$. Each axiom identifies an atom, and $\mathcal{O}$ equals the principal ideal of the atom $\mathfrak{a}_3$ consisting of the axiom $\mathtt{B} \sqcap \mathtt{D} \sqsubseteq \mathtt{C} \sqcup \mathtt{E} \sqcup (\mathtt{G} \sqcup \neg\mathtt{G})$. Although the signature of $\mathfrak{a}_3$ contains neither $\mathtt{A}$ nor $\mathtt{F}$, the set $\Sigma = \{\mathtt{A}, \mathtt{F}\}$ is indeed a minimal seed signature of the module $(\mathfrak{a}_3]$. The need of this axiom for the signature Σ is not evident at first sight.

On the other hand, $\mathsf{Lab}_{\text{ssig}}$ does not include "irrelevant" terms: since under any interpretation of $\mathtt{G}$ the concept $\mathtt{G} \sqcup \neg\mathtt{G}$ is always $\top$, then $\mathtt{G}$ does not appear in any of the minimal seed signatures of the atom $\mathfrak{a}_3$. Although this can be seen as a good behaviour of $\mathsf{Lab}_{\text{ssig}}$, we need to consider how $\top\bot^*$-modules are extracted: whenever an axiom α is non local, the seed signature Σ is extended with all terms in $\widetilde{\alpha}$. This means that in our example $\mathtt{G}$ belongs to the extended signature of the module $\mathfrak{a}_3$, and can interfere with other terms of the seed signature of input, even if it is logically irrelevant. We need to keep track of this information too. We define $\mathsf{Lab}_{\text{FEM}}$ to be the refinement of $\mathsf{Lab}_{\text{ssig}}$ by adding to the label of each atom also its irrelevant terms, i.e., all terms in the module that do not occurr in any minimal seed signature. In Fig. 2 we show such LAD for the ontology Koala. The refinement of $\mathsf{Lab}_{\text{ssig}}$ affects only the atom labelled $\{\mathtt{Koala}\}$.

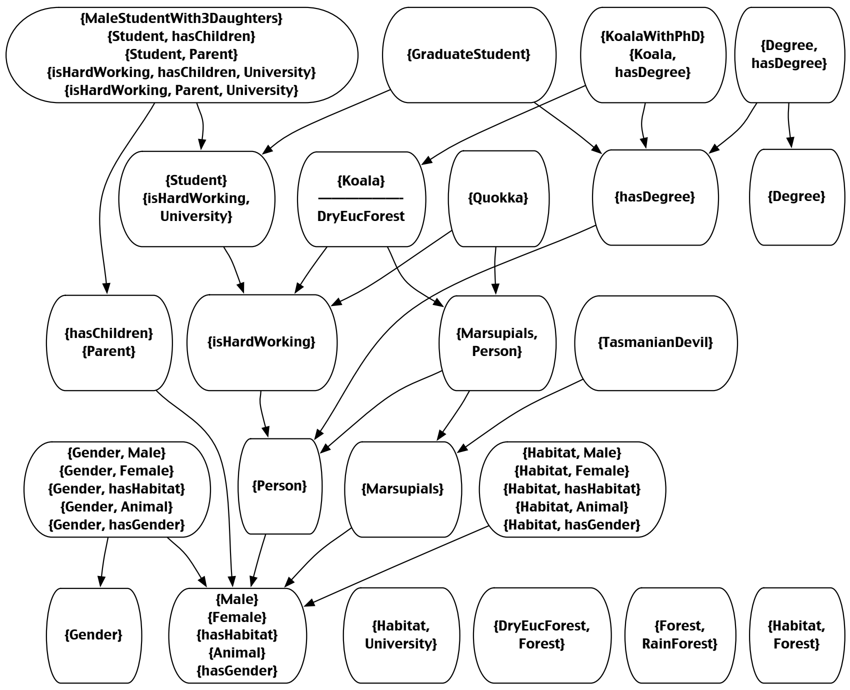


Figure 2. LAD for performing FME of the ontology Koala

A problem that arises with $\mathsf{Lab}_{\mathrm{FEM}}$ consists of the possibility of labels to be of exponential size w.r.t. the ontology size, as in the following example.

Example 21. Let us consider the the family of ontologies $\mathcal{O}_n = \{\mathtt{A}_i \equiv \mathtt{A}_{i-1} \sqcup \mathtt{A}'_{i-1}, \mathtt{B}_i \equiv \mathtt{B}_{i-1} \sqcup \mathtt{B}'_{i-1}, \mathtt{C}_i \equiv \mathtt{C}_{i-1} \sqcup \mathtt{C}'_{i-1}, \mathtt{D}_i \equiv \mathtt{D}_{i-1} \sqcup \mathtt{D}'_{i-1}, \mathtt{A}_i \sqcup \mathtt{B}_i \sqsubseteq \mathtt{C}_i \sqcap \mathtt{D}_i \mid i = 1, \ldots, n\}$. Then, $\#\mathcal{O}_n = 5n$, and each AD consists of $5n$ atoms with an axiom each. Now, some minimal seed signatures for the atom $\mathfrak{a}_i^n = \{\mathtt{A}_i \sqcup \mathtt{B}_i \sqsubseteq \mathtt{C}_i \sqcap \mathtt{D}_i\}$ contain 2 terms, one from $\{\mathtt{A}_i, \mathtt{B}_i\}$ and one from $\{\mathtt{C}_i, \mathtt{D}_i\}$. However, each term can be replaced by the two terms defining it (for example, $\mathtt{A}_i$ can be replaced by $\mathtt{A}_{i-1}, \mathtt{A}'_{i-1}$). Since this procedure can be recursively applied, the atom $\mathfrak{a}_i^n$ results to have at least 4^i minimal seed signatures.

Despite the discussion throughout this section, a procedure to perform FME is still not defined, and the exploitation of this matter is in our future work.

5 Outlook

We presented the *Atomic Decomposition* of an ontology, and showed its definition, its properties, and its tractable generation. The AD reveals the overall modular structure of an ontology, thus we expect to apply such decomposition in various scenario, from the module count, to the support to ontology engineers in the modeling phase. We have also introduced a family of refinements of AD, called *Labelled Atomic Decompositions*, justifying the need for labels in the specific task consisting of extracting a module without loading the ontology.

Future work includes the completion of the preliminary results described here. In particular, we want to explore suitable ADs/LADs for the tasks described in this paper. Then, we are open to investigate other tasks that could be of interest for users of ontologies and where a suitable LAD would be of help.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider, eds. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. G.Antoniou, K. Kehagias. A note on the refinement of ontologies. Int. J. of Intelligent Systems, vol. 15, pp. 623–632 (2000)
3. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. In: Proc. of ISWC/ASWC-07. LNCS, vol. 4825, pp. 183–196 (2007)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In Proc. of IJCAI-07. pp. 298–304 (2007)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artif. Intell. Research, vol. 31, pp. 273–318 (2008)
6. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: an empirical study. In V. Haarslev, D. Toman, and G. Weddell (eds.) Proc. of DL 2010. `ceur-ws.org`, vol. 573. (2010)
7. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The Modular Structure of an Ontology: Atomic Decomposition. Accepted for IJCAI-11 (2011)
8. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The Modular Structure of an Ontology: Atomic Decomposition and Module Count. Accepted for WoMO-11 (2011)
9. Del Vescovo, C., Parsia, B., Sattler, U.: Topicality in Logic-Based Ontologies. Accepted for ICCS-11 (2011)
10. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The Modular Structure of an Ontology: Atomic Decomposition. Tech. rep., The University of Manchester (2011) Available at `http://bit.ly/i4olY0`
11. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. of KR-06. pp. 187–197. AAAI Press (2006)
12. Horridge, M.. Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) Proc. of ISWC-08. LNCS, vol. 5318, pp. 323–338 (2008)
13. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., Berlanga Llavori, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: Proc. of ESWC-08. LNCS, vol. 5021, pp. 185–199 (2008)
14. Jimeno, A., Jiménez-Ruiz, E., Berlanga, R., Rebholz-Schuhmann, D.: Use of shared lexical resources for efficient ontological engineering. In: SWAT4LS-08. `ceur-ws.org` (2008)
15. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularization. In: Modular Ontologies, LNCS, vol. 5445, pp. 25–66. Springer-Verlag (2009)
16. Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., Zakharyaschev, M.: Minimal module extraction from DL-Lite ontologies using QBF solvers. In: Proc. of IJCAI-09. pp. 836–841 (2009)
17. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Cuenca Grau, B., Horrocks, I., Motik, B., Sattler, U. (eds.) DL 2009. `ceur-ws.org`, vol. 477. (2009)

Satisfiability in $\mathcal{EL}$ with sets of Probabilistic ABoxes[⋆]

Marcelo Finger, Renata Wassermann, and Fabio G. Cozman

University of São Paulo, São Paulo, Brazil
mfinger@ime.usp.br,renata@ime.usp.br,fgcozman@usp.br

Abstract. We present $\mathcal{EL}$PA, a probabilistic extension of the lightweight DL $\mathcal{EL}$ with a fixed TBox and a set of probabilistic ABoxes, and study the problem of satisfiability in such context.

1 Introduction

This work studies an extension of Description Logic $\mathcal{EL}$, called $\mathcal{EL}$PA, that allows for probabilistic assessments on ABoxes. We concentrate on the problem of verifying the *satisfiability* of an $\mathcal{EL}$PA-knowledge base, proposing algorithms for this problem based on recent advances on probabilistic satisfiability (PSAT) [FB11]. Consider an example, adapted from [LS10].

Example 1.1 Two symptoms of Lyme disease are fever and fatigue. As these symptoms are common and the disease is rare, the chance that they are indeed caused by Lyme disease is small. Nevertheless, because the disease is of difficult diagnosis, patients get treated if there is a chance that they have it.

The TBox $\mathcal{T}_0$ contains the following axioms:	And the following ABox $\mathcal{A}_0$:
Fatigue $\sqsubseteq$ Symptom	Fever(s_1)
Fever $\sqsubseteq$ Symptom	hasSymptom(john, s_1)
Lyme $\sqsubseteq$ Disease	Fatigue(s_2)
Symptom $\sqsubseteq$ ∃hasCause.Disease	hasSymptom(john, s_2)
Patient $\sqsubseteq$ ∃suspectOf.Disease	Patient(john)
Patient $\sqsubseteq$ ∃hasSymptom.Symptom	
∃hasSymptom.(∃hasCause.Lyme)$\sqsubseteq$ ∃suspectOf.Lyme	

Consider also the following probabilistic statements originating from medical experience on symptoms that are caused by Lyme disease.

$\mathcal{A}_1$ = ∃hasCause.Lyme(s_1), $P(\mathcal{A}_1) \geq 0.1$

$\mathcal{A}_2$ = ∃hasCause.Lyme(s_2), $P(\mathcal{A}_2) \geq 0.2$

Now we want to know whether we can consistently assert an upper bound p_{ub} for the probability of John having Lyme disease:

$\mathcal{A}_3$ = ∃suspectOf.Lyme(john), $P(\mathcal{A}_3) \leq p_{\text{ub}}$ □

The set of statements in the example above is a *probabilistic knowledge base.* It contains four probability assignments; $\mathcal{A}_0$, which is the conjunction of 5 atomic statements, is assigned probability 1; the other three atomic ABoxes $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ get probabilities smaller than 1.

⋆ Work supported by Fapesp Thematic Project 2008/03995-5 (LOGPROB). Authors supported by CNPq grants PQ 302553/2010-0, 304043/2010-9, 305395/2010-6.

Our method is an alternative to existing combinations of DL with probabilities that impose deterministic restrictions on probabilities. For example, [LS10] assigns probabilities to concepts over a fixed interpretation, forcing the probability of ABoxes to be either 0 or 1. No such deterministic "side effects" occur in our method.

Our goal in this work is to formally define the notion of $\mathcal{EL}$PA-knowledge bases and its satisfiability problem and provide algorithms to verify it. Adding probabilities to logic sentences usually adds complexity; e.g. probabilistic 2SAT is NP-complete [GKP88]. We show that $\mathcal{EL}$PA-satisfiability is in NP.

1.1 Related work

There have been several proposals to add probabilities to description logics in recent years [LS08]; most of this work is based on various kinds of probabilistic logic [GT07,Hal03].

Probabilistic description logics differ in several dimensions. Some approaches associate probabilities with elements of TBoxes [Jae94,KLP97,CP09], while others associate probabilities with ABoxes [DS05], and still others combine both kinds of assessments [Luk08]. In this paper we focus on probabilities over ABoxes.

As an alternative classification scheme, some logics assign probabilities over elements of the domain [DPP06,DS05,Hei94,Jae94,KLP97,Luk08], while others assign probabilities over interpretations [CL06,DFL08,LS10], with some logics in between [Seb94]. In this paper we focus on probabilities over interpretations.

Yet another classification is possible, as we have probabilistic description logics that allow for assessments of stochastic independence, often organized through graphs [CL06,DPP06,KLP97,CP09], and logics that do not allow for assessments of stochastic independence [DS05,Hei94,Jae94,Luk08,LS10]. In this paper we do not allow for stochastic independence.

In a sense, our work is a refinement of First Order Probabilistic Logic by Jaumard et al [JFSS06]; however, we use the decidable and tractable logic $\mathcal{EL}$, and we show that our probabilistic version remains in NP. Note that probability assignments remain *external* to the logic $\mathcal{EL}$; this has the advantage of making it capable of dealing with conditional probabilities of ABox statements in a classical manner, as $P(A(a)|B(b)) = P(A(a) \wedge B(b))/P(B(b)$. A complete treatment of conditional probabilities remains outside the scope of this work. Another related problem is *probability inference*; that is, determining the maximal and minimal values for p_{ub} that leave the knowledge base satisfiable; this problem is also outside the scope of this work.

1.2 Organization of the paper

The remainder of the paper is organized as follows. In the next section, we introduce the logic $\mathcal{EL}$PA. We first define the probabilistic assignments that are allowed in the ABox, and then formalise the satisfiability problem for $\mathcal{EL}$PA, showing that it is in NP. We show that the probabilistic knowledge base in

$\mathcal{EL}$PA can be translated into a normal form that is used in Section 3, where an algorithm for testing the satisfiability of $\mathcal{EL}$PA is presented.

2 The Probabilistic DL $\mathcal{EL}$PA

We introduce $\mathcal{EL}$PA, a probabilistic extension of the polynomial-time Description Logic $\mathcal{EL}$ with a fixed TBox and a set of probabilistic ABoxes.

We first establish a regular $\mathcal{EL}$ vocabulary. Fix countably infinite sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of *concept names*, *role names*, and *individual names*, respectively. The set of *$\mathcal{EL}$-concepts* is given by the following syntax rules:

$$C ::= A \mid C \sqcap D \mid \exists r.C$$

where A ranges over $\mathsf{N_C}$, C and D over $\mathcal{EL}$-concepts and r over $\mathsf{N_R}$. No negation or disjunction of concepts is expressible in this language.

A *TBox* is a finite set of *concept inclusions* (CIs) of the form $C \sqsubseteq D$; TBoxes usually represent an ontology. On the other hand, *ABoxes* represent instance data and obey the following syntax rules

$$\mathcal{A} ::= C(a) \mid r(a, b) \mid \mathcal{A} \wedge \mathcal{A}'$$

where C and r are as before, $a, b \in \mathsf{N_I}$ and $\mathcal{A}$, $\mathcal{A}'$ range over ABoxes.

The standard $\mathcal{EL}$ semantics is used for TBoxes and ABoxes, based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps each each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $r \in \mathsf{N_R}$ to a subset $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each $a \in \mathsf{N_I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; see [BBL05]. We extend the interpretation $\mathcal{I}$ for all concepts in the usual way. So $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$. Then concept inclusion $C \sqsubseteq D$ is satisfied by interpretation $\mathcal{I}$, represented by $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Similarly TBox $\mathcal{T}$ is satisfied by interpretation $\mathcal{I}$, $\mathcal{I} \models \mathcal{T}$, iff $\mathcal{I} \models C \sqsubseteq D$ for every $C \sqsubseteq D \in \mathcal{T}$.

For ABoxes, we say that $C(a)$ is satisfied by $\mathcal{I}$, represented by $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$; similarly, $\mathcal{I} \models r(a, b)$ iff($a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$; and $\mathcal{I} \models \mathcal{A} \wedge \mathcal{A}'$ if both $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{A}'$. If both a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ are true under $\mathcal{I}$, we say that the pair $(\mathcal{T}, \mathcal{A})$, called a *(deterministic) knowledge base*, is *satisfied* by $\mathcal{I}$. The problem of deciding if a deterministic knowledge base $(\mathcal{T}, \mathcal{A})$ is satisfiable in $\mathcal{EL}$ can be solved in polynomial time [Bra04].

2.1 Probability Assignments to ABoxes

We now introduce probabilities in ABoxes. We deal with probability distribution π over the set of interpretations endowed with a suitable algebra.

Let $\mathcal{T}$ be a TBox. Given a probability distribution π over the set of all interpretations $\mathbb{I}$, the probability of an ABox $\mathcal{A}$ in the context of $\mathcal{T}$ is given by the probability of all interpretations that satisfy all of $\mathcal{T}$ and $\mathcal{A}$, that is, the probability of $\{\mathcal{I} | \mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}\}$.

A *probability assignment* to an ABox, is an expression of the form

$$P(\mathcal{A}) \bowtie p,$$

where $\mathcal{A}$ is an ABox, $\bowtie \in \{\leq, \geq, =\}$ and $p \in \mathbb{Q}, 0 \leq p \leq 1$. Note that probability assignments are external to the logic $\mathcal{EL}$, and are *not* statements in the logic.

Let $\mathcal{PA}$ be a set of k probability assignments

$$\mathcal{PA} = \{P(\mathcal{A}_i) \bowtie_i p_i | 1 \leq i \leq k\}.$$

Then the pair $\langle \mathcal{T}, \mathcal{PA} \rangle$ is a *probabilistic knowledge base* in the DL $\mathcal{EL}$ with sets of probability assignments, $\mathcal{EL}$PA. Clearly, all probability assignments in $\mathcal{PA}$ are to be evaluated in the context $\mathcal{T}$.

The main problem of this work is, given an $\mathcal{EL}$PA probabilistic knowledge base, determine whether there exists a probability distribution π such that, in the context of the TBox $\mathcal{T}$, π satisfies all the assignments in $\mathcal{PA}$; this is the *satisfiability problem* for an $\mathcal{EL}$PA-knowledge base. Before we formalize this problem, we must "finitize" the set of all interpretations of a TBox $\mathcal{T}$, as the set $\mathbb{I}_{\mathcal{T}}$ of all interpretations of a TBox $\mathcal{T}$ is uncountably infinite.

For that, define an equivalence relation $\simeq_{\mathcal{PA}} \subseteq \mathbb{I}_{\mathcal{T}} \times \mathbb{I}_{\mathcal{T}}$, where $\mathcal{PA} = \{P(\mathcal{A}_i) = p_i | 1 \leq i \leq k\}$ is a set of probabilistic assignments, such that

$$\mathcal{I} \simeq_{\mathcal{PA}} \mathcal{I}' \text{ iff for every } k, 1 \leq i \leq k, \mathcal{I} \models \mathcal{A}_i \text{ if and only if } \mathcal{I}' \models \mathcal{A}_i;$$

that is, $\mathcal{I}$ and $\mathcal{I}'$ satisfy the same ABoxes in $\mathcal{PA}$ in a context of TBox $\mathcal{T}$.

Lemma 2.1 *Let n be the number of atomic elements in $\mathcal{PA}$. The relation $\simeq_{\mathcal{PA}}$ is an equivalence relation on $\mathbb{I}_{\mathcal{T}} \times \mathbb{I}_{\mathcal{T}}$ and the set of equivalence classes $\mathbb{I}_{\mathcal{T}}/\simeq_{\mathcal{PA}}$ has at most 2^n distinct equivalence classes.*

Each equivalence class of $\mathbb{I}/\simeq_{\mathcal{PA}}$ will be represented by any interpretation $\mathcal{I}$ in it. We can now formalise the satisfiability problem for an $\mathcal{EL}$PA-knowledge base $\langle \mathcal{T}, \mathcal{PA} \rangle$. We will write $\mathcal{I}(\mathcal{A}) = 1$ for $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I}(\mathcal{A}) = 0$ for $\mathcal{I} \not\models \mathcal{A}$.

2.2 The Satisfiability of Probabilistic Knowledge Bases

Let n be the number of atomic elements in $\mathcal{PA}, |\mathcal{PA}| = k$. Consider $\mathcal{I}_1, \ldots, \mathcal{I}_{n'}$, $n' \leq 2$ be all the $\simeq_{\mathcal{PA}}$-distinct interpretations that satisfy $\mathcal{T}$. Consider a $k \times n'$ matrix $A = [a_{ij}]$ such that $a_{ij} = \mathcal{I}_j(\mathcal{A}_i)$. The *probabilistic satisfiability problem* for an $\mathcal{EL}$PA-knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{PA} \rangle$ is to decide if there is a probability vector π of dimension n' that obeys the *$\mathcal{EL}$PA-restrictions*:

$$A\pi \bowtie p, \qquad \sum \pi_i = 1, \qquad \pi \geq 0. \tag{1}$$

An $\mathcal{EL}$PA-knowledge base $\mathcal{K}$ is *satisfiable* iff its associated $\mathcal{EL}$PA-restrictions (1) have a solution. If π is a solution to (1) we say that π satisfies $\mathcal{K}$. The last two conditions of (1) force π to be a probability distribution. It is convenient to assume that first two conditions of (1) are joined, A is a $(k+1) \times n'$ matrix with 1's at its first line, $p_1 = 1$ in vector $p_{(k+1)\times 1}$, so $\bowtie_1$-relation is "="; we will keep this convention in the rest of the paper.

Example 2.2 Recall Example 1.1 with $p_{\text{ub}} = 0.3$, where the probability of John having Lyme is at most 30%. We consider only the 3 ABox with probabilistic assignments, and only interpretations of these atoms that are jointly consistent with the fixed TBox and the fixed (probability 1) ABox formulas. Consider the following probability distribution π and the probability it assigns to the ABoxes in Example 1.1.

	π	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
$\mathcal{I}_1$	0.75	0	0	0
$\mathcal{I}_2$	0.10	0	1	1
$\mathcal{I}_3$	0.03	1	0	1
$\mathcal{I}_4$	0.12	1	1	1
	1.00	0.15	0.22	0.25

It is easy to verify that the interpretations $\mathcal{I}_1$–$\mathcal{I}_4$ are all consistent with $\langle \mathcal{T}_0, \mathcal{A}_0 \rangle$, so all probability relations are verified by π, so probabilistic database for $p_{\text{ub}} = 0.3$ is satisfiable. □

Some important questions remain: how to compute a probability distribution when one exists, and whether that probabilistic knowledge base remains satisfiable when $p_{\text{ub}} = 0.05$ or not, and how to verify it. This paper presents algorithms for that.

An important result of [GKP88] guarantees that a satisfiable knowledge base has a "small" witness:

Fact 2.3 *If $\mathcal{ELC}$PA-restrictions (1) for knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{PA} \rangle$ with $\mathcal{PA} = \{P(\mathcal{A}_i) = p_i | 1 \leq i \leq k\}$ have a solution, then there are $k+1$ columns of matrix A such that the system $A_{(k+1)\times(k+1)}\pi = p_{(k+1)\times 1}$has a solution $\pi \geq 0$.*

This result is a consequence of Caratheodory's Theorem [Eck93], which states that if a k-dimensional point is a convex combination of m points, then it is a convex combination of at most $k+1$ points among them. Fact 2.3 gives an NP-certificate for the satisfiability of an $\mathcal{EL}$PA-knowledge basel; hence:

Corollary 2.4 *The $\mathcal{EL}$PA-satisfiability problem is in NP.*

Finding polynomial-sized certificates is the heart of the matter. We will now study algorithms that solve the $\mathcal{EL}$PA-satisfiability problem. We start by defining a normal form for $\mathcal{EL}$PA-knowledge base. Note that Fact 2.3 is stated for equality only, and we also allow inequalities; the normal form will be useful both for the algorithms and for showing that all those cases can be reduced to equality only, with all probabilistic assignments over atoms only.

2.3 A Normal Form for Probabilistic Knowledge Base

For the sake of providing a normal form, we add a few new convenient definitions. Let N_0 be a set of 0-ary atomic propositions. A *propositional rule* is an expression of the form $q \to \mathcal{A}_1$ or $\mathcal{A}_2 \to q$, where $q \in \mathsf{N}_0$ and $\mathcal{A}_1, \mathcal{A}_2$ ABoxes, with the obvious semantic that $\mathcal{I} \models q \to \mathcal{A}_1$ iff $\mathcal{I} \not\models q$ or $\mathcal{I} \models \mathcal{A}_1$; and $\mathcal{I} \models \mathcal{A}_2 \to q$ iff $\mathcal{I} \models q$ or $\mathcal{I} \not\models \mathcal{A}_2$. We extend the notion of ABox such that

$$\mathcal{A} ::= C(a) \mid r(a,b) \mid q \mid \mathcal{A} \wedge \mathcal{A}'$$

such that $q \in \mathsf{N}_0$ and $C(a), r(a,b), \mathcal{A}, \mathcal{A}'$ are as before; we call $q, C(a), r(a,b)$ *atomic* ABoxes.

If $\mathcal{R}$ is a set of propositional rules and $\mathcal{A}$ an ABox, $\mathcal{R} \cup \mathcal{A}$ is a set of Horn clauses, and thus has a polynomial-time computable minimal model; so the $\mathcal{I}$-satisfiability of $\mathcal{R} \cup \mathcal{A}$ reduces to the $\mathcal{I}$-satisfiability of the atomic positive formulas in its minimal model. Thus the satisfiability problem of $\mathcal{EL}$ with TBoxes, ABoxes and sets of propositional rules can be achieved in polynomial time.

We also distinguish deterministic ABoxes, which are assigned probability 1, from probabilistic ABoxes, which are assigned probabilities < 1.

We then extend previous definitions with the notion of a set of propositional rules. For the rest of this paper, an *extended* $\mathcal{EL}$PA-knowledge base is a 4-tuple $\mathcal{K}^e = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}, \mathcal{PA} \rangle$, in which the *probabilistic assignment of ABoxes* $\mathcal{PA}$ is evaluated in an *(deterministic) evaluation context* consisting of a triple $\mathcal{C} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ of TBox, propositional rules and deterministic ABox $\mathcal{A}$; we also represent $\mathcal{K}^e = \langle \mathcal{C}, \mathcal{PA} \rangle$. Clearly, an $\mathcal{EL}$PA-knowledge base $\mathcal{K}$ is a special case of an extended $\mathcal{EL}$PA-knowledge base $\mathcal{K}^e$ the previous view in which $\mathcal{R} = \varnothing$ and $\mathcal{A}$ is part of $\mathcal{PA}$.

Now we define the normal form. A knowledge base $\mathcal{K}^e = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}, \mathcal{PA} \rangle$ is in *(atomic) normal form* if $\mathcal{PA}$ is of the form

$$\mathcal{PA} = \{P(y_i) = p_i \mid y_i \text{ is an atom, } 1 \le i \le k\}, \text{ with } 0 < p_i < 1.$$

In this case, $\mathcal{PA}$ is an *atomic probability assignment* evaluated in context $\mathcal{C} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$. Clearly, $\mathcal{C}$ is a small generalisation of the deterministic knowledge bases of, for instance, [BBL05].

By adding a small number of propositional rules, any knowledge base can be brought into atomic normal form.

Theorem 2.5 (Normal Form) *For every extended $\mathcal{EL}$PA-knowledge base $\mathcal{K}^e$ there exists an atomic normal form knowledge base $\mathcal{K}^e_{nf}$ that is satisfiable iff $\mathcal{K}^e$ is; the former can be obtained from the latter in polynomial time $O(k \times \ell)$, where $k = |\mathcal{PA}|$ and ℓ is the largest number of conjuncts in an ABox in $\mathcal{PA}$.* □

Example 2.6 We transform the knowledge base(s) of Example 1.1 into the normal form. The TBox $\mathcal{T}_0$ and deterministic ABox $\mathcal{A}_0$ remain the same. We introduce atoms q_1, q_2, q_3 for ABoxes $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ respectively, which generates the following set or rules $\mathcal{R}_0$:

$$q_1 \to \exists\text{hasCause.Lyme}(s_1), \quad q_2 \to \exists\text{hasCause.Lyme}(s_2), \quad \exists\text{suspectOf.Lyme(john)} \to q_3$$

$\mathcal{C}_0 = \langle \mathcal{T}_0, \mathcal{R}_0, \mathcal{A}_0 \rangle$ is the evaluation context; the atomic probability assignment is

$$\mathcal{PA}_0 = \{ \quad P(q_1) = 0.1, \qquad P(q_2) = 0.2. \qquad P(q_3) = p_{\text{ub}} \quad \}.$$

The normal form knowledge base is $\mathcal{K}_0^e = \langle \mathcal{C}_0, \mathcal{PA}_0 \rangle$. Note that the probability distribution of Example 2.2 does not satisfy $\mathcal{K}_0^e$ when $p_{\text{ub}} = 0.3$, but by Theorem 2.5 there must exist other interpretations involving q_1, q_2, q_3 and rules $\mathcal{R}_0$ and another probability distribution π that satisfies $\mathcal{K}_0^e$. □

The following result allows us to see a satisfiable normal form knowledge base $\mathcal{K}_{nf}^e$ as an interaction between a solution to assignments $\mathcal{PA}$ constrained by the $\mathcal{EL}$-decisions of context $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$. An interpretation $\mathcal{I}$ is $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$-consistent if $\mathcal{I}$ jointly satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$. Recall that we represent the binary $\mathcal{I}$-evaluation of ABoxes such that $\mathcal{I}(\mathcal{A}) = 1$ iff $\mathcal{I} \models \mathcal{A}$. Lemma 2.7 is the basis for the $\mathcal{EL}$PA-satisfiability solving algorithm that we present in the next section.

Lemma 2.7 *A normal form knowledge base $\mathcal{K}^e = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}, \mathcal{PA} \rangle$ is satisfiable iff there is a binary $(k+1) \times (k+1)$-matrix $A_{\mathcal{K}^e}$, such that all of its $\{0,1\}$-columns represent interpretations that are $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$-consistent and $A_{\mathcal{K}^e} \cdot \pi = p$ has a solution $\pi \geq 0$.*

3 An Algorithm for $\mathcal{EL}$PA Satisfiability

We present a logic-algebraic algorithm to verify the satisfiability of a normal form knowledge base $\mathcal{K}^e = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}, \mathcal{PA} \rangle$ and, if the answer is positive, present a satisfying model in the form of a set of $k+1$ $\mathcal{EL}$-interpretations, where $k = |\mathcal{PA}|$, and a probability distribution over them.

We first establish some terminology. If A is a matrix, A^j is its j-th column and A_i is its i-th line; $A_{(s)}$ is the state of matrix A at step s. If A is a matrix and b a column of compatible dimension, $A[j := b]$ is obtained by replacing A's j-th column with b. A square matrix that has an inverse is *non-singular*. A matrix A that satisfies conditions (2) is a *feasible solution* for $\mathcal{PA}$.

$$\begin{bmatrix} 1 & \cdots & 1 \\ a_{1,1} & \cdots & a_{1,k+1} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,k+1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_{k+1} \end{bmatrix} = \begin{bmatrix} 1 \\ p_1 \\ \vdots \\ p_k \end{bmatrix}, \qquad (2)$$

$$a_{i,j} \in \{0,1\}, \qquad A \text{ is non-singular}, \qquad \pi_j \geq 0$$

We always assume that the lines of A are ordered such that the input probabilities $p_1, \ldots, p_k$ in (2) are in decreasing order. By Lemma 2.7, $\langle \mathcal{C}, \mathcal{PA} \rangle$ has a solution iff there is a partial solution A satisfying (2) such that if $\pi_j > 0$ then $a_{1,j}, \ldots, a_{k,j}$ represent $\mathcal{C}$-consistent interpretations for $1 \leq i \leq k, 1 \leq j \leq k+1$. We usually abuse terminology calling A^j a $\mathcal{C}$-consistent column.

This method is based on PSAT-solving method of [FB11], which is an improvement on the methods of [KP90,HJ00]; it consists of an algebraic optimisation problem in the form of a special linear program of the form

$$\begin{array}{ll} \text{minimize} & \textit{objective function}\langle |J|, f \rangle \\ \text{subject to} & A\pi = p, \pi \geq 0, f = \sum_{j \in J} \pi_j \text{ and} \\ & J = \{ j | A^j \text{ is } \mathcal{C}\text{-inconsistent}, \pi_j > 0 \} \end{array} \qquad (3)$$

which is solved iteratively by the simplex algorithm [BT97]. Matrix A is a $(k+1) \times (k+1)$ {0,1}-matrix, whose columns represents an $\mathcal{EL}$-interpretations and whose lines represent the atoms occurring in $\mathcal{PA}$. An iterative step s receives a matrix $A_{(s)}$ and employs a *column generation* method that solves an *auxiliary problem*; the latter is a logic-based satisfiability problem that employs $\mathcal{EL}$-decision procedure, generates a column that replaces some column in $A_{(s)}$, obtains $A_{(s+1)}$ and decreases the objective function $\langle |J|, f\rangle$, where $\langle |J_1|, f_1\rangle > \langle |J_2|, f_2\rangle$ iff$0 \leq |J_1| < |J_2|$ or $|J_1| = |J_2|$ and $f_1 < f_2$, until its minimum is reached. The objective function is discussed in Section 3.1.

In the iterative method, some columns are not $\mathcal{C}$-consistent and the process is done such that the number of $\mathcal{C}$-consistent columns A^j associated to $\pi_j > 0$ never decreases.

We now define $A_{(0)}$, the starting feasible solution. For that, consider an empty context $\mathcal{C} = \varnothing$, that is a knowledge base $\langle \varnothing, \mathcal{PA}\rangle$. As the elements of p are in decreasing order, consider the $\{0,1\}$-matrix $I^* = [a_{i,j}]_{1 \leq i,j \leq k+1}$ where $a_{i,j} = 1$ iff $i \leq j$, that is, I^* is all 1's in and above the diagonal, 0's elsewhere. As p is in decreasing order, I^* satisfies $\langle \varnothing, \mathcal{PA}\rangle$ and is called a *relaxed solution* for $\langle \mathcal{C}, \mathcal{PA}\rangle$. Clearly, I^* is a feasible for $\mathcal{PA}$. Make $A_{(0)} = I^*$.

Example 3.1 Consider the form of knowledge base in Example 2.6 with $p_{\text{ub}} = 0.3$ (left) and $p_{\text{ub}} = 0.05$ (right). An initial feasible solution for it is $A_{(0)} \cdot \pi^{(0)} = p$, with atoms ordered in decreasing probability, namely

$$\begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\0&0&0&1\end{bmatrix} \cdot \begin{bmatrix}0.7\\0.1\\0.1\\0.1\end{bmatrix} = \begin{bmatrix}1\\0.3\\0.2\\0.1\end{bmatrix}\begin{array}{l}\\ q_3\\ q_2\\ q_1\end{array} \quad\Bigg|\quad \begin{bmatrix}1&1&1&1\\0&1&1&1\\0&0&1&1\\0&0&0&1\end{bmatrix} \cdot \begin{bmatrix}0.8\\0.1\\0.05\\0.05\end{bmatrix} = \begin{bmatrix}1\\0.2\\0.1\\0.05\end{bmatrix}\begin{array}{l}\\ q_2\\ q_1\\ q_3\end{array}$$

On the left, all columns are $\mathcal{C}$-consistent, so the problem is satisfiable with solution $A_{(0)}$ and π as above. On the right, the second and third columns of $A_{(0)}$ are $\mathcal{C}$-inconsistent, so the decision is not yet made.

The relaxed solution is the initial feasible solution of our method. Further feasible solutions are obtained by generating new $\{0,1\}$-columns and substituting them into a feasible solution, as shown by the following.

It is well known from the pivoting in the simplex algorithm that given any $\{0,1\}$-column of the form $b = [1 \; b_1 \; \cdots \; b_k]'$, $A[j := b]$ is a feasible solution. So we simply assume there is a function $merge(A, b)$ that computes it. Our method moves through feasible solutions, at each step generating a column b that decreases the value of the objective function.

3.1 The Objective Function

In a feasible solution A such that $A\pi = p$ and $\pi \geq 0$, some columns may not be $\mathcal{C}$-consistent. Let $J = \{j | A^j$ is $\mathcal{C}$-inconsistent and $\pi_j > 0\}$; J is the set of column indexes in A corresponding to $\mathcal{C}$-inconsistent columns with non-null associated probability; clearly $|J| \leq k+1$. If $J = \varnothing$, we call A a *solution.* As $|J| = 0$ when a solution is found, it is one component of the objective function. However, it is

Algorithm 3.1 ELPA-satisfiability solver

Input: A normal form ELPA knowledge base $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}, \mathcal{PA} \rangle$.

Output: Solution A; or "No", if unsatisfiable.

```
1: $p := sortDecrescent(\{1\} \cup \{p_i | P(y_i) = p_i \in \mathcal{PA}\}$;
2: $A_{(0)} := I^*$; $s := 0$; compute $\langle |J_{(s)}|, f_{(s)} \rangle$;
3: while $\langle |J_{(s)}|, f_{(s)} \rangle \neq \langle 0, 0 \rangle$ do
4:     $b^{(s)} = GenerateColumn(A_{(s)}, p, \mathcal{C})$;
5:     return "No" if $b_1^{(s)} < 0$; /* instance is unsat */
6:     $A_{(s+1)} = merge(A_{(s)}, b^{(s)})$;
7:     increment $s$; compute $\langle |J_{(s)}|, f_{(s)} \rangle$;
8: end while
9: return $A_{(s)}$; /* PSAT instance is satisfiable */
```

not guaranteed that, if a solution exists, we can find a sequence of iterations in which $|J|$ decreases at every step s.

The second component of the objective function is the sum of probabilities of $\mathcal{C}$-inconsistent columns, $f = \sum_{j \in J} \pi_j$. Note that f and $|J|$ become 0 at the same time, which occurs iff a positive decision is reached. The simplex algorithm with appropriate column generation ensures that, if there is a solution, it can be obtained with finitely many steps of non-increasing f-values. We thus use a combined objective function $\langle |J|, f \rangle$ ordered lexicographically.

We first try to minimise the number of $\mathcal{C}$-inconsistent columns; if this is not possible, then minimise f, keeping J constant. So a knowledge base instance $\langle \mathcal{C}, \mathcal{PA} \rangle$ associated to program (3) is satisfiable iff the objective function is minimal at $\langle 0, 0 \rangle$.

Assume there is a function $GenerateColumn(A, p, \mathcal{C})$, presented at Section 3.2, that generates a $\mathcal{C}$-consistent column that decreases the objective function, if one exists; otherwise it returns an illegal column of the form $[-1 \ \cdots]$. Algorithm 3.1 presents a method that decides a PSAT instance by solving problem (3).

Algorithm 3.1 starts with a relaxed solution for $\langle \mathcal{C}, \mathcal{PA} \rangle$ (line 2), and via column generation (line 4) generates another feasible solution (line 6), decreasing the objective function, until either the search fails (line 5) or a solution is found; the latter only occurs with the termination of the loop in lines 3–8, when the objective function reaches $\langle 0, 0 \rangle$.

3.2 Column Generation for ELPA

Algorithm 3.1 is, unsurprisingly, almost the same algorithm for PSAT solving in [FB11]; the only difference between the two rests in the column generation method $GenerateColumn(A, p, \mathcal{C})$.

It has been shown in [FB11] that to eliminate a $\mathcal{C}$-inconsistent column A^j associated to $\pi_j > 0$, a new $\mathcal{C}$-consistent column $b = [1 \ y_1 \dots y_k]'$ to substitute A^j must satisfy the set of linear inequalities:

$$(LR_{ij}) \quad (A_j^{-1}\pi_i - A_i^{-1}\pi_j)[1 \ y_1 \dots y_k]' \geq 0, \quad 1 \leq i \leq k+1 \qquad (4)$$

Such a column is here obtained by a combination of a SAT solver, which guarantees that (4) is verified, with an $\mathcal{EL}$-solver to guarantee that b is $\mathcal{C}$-consistent. This combination can be done in several ways.

(a) By coding the polynomial time $\mathcal{EL}$-decision in a SAT solver.
(b) By using $\mathcal{EL}$-theories as an SMT (SAT Modulo Theories) engine.
(c) By coupling an $\mathcal{EL}$-solver at the end of the SAT solver, rejecting $\mathcal{C}$-inconsistent answers, and proceeding with the SAT solver after the rejection.

The latter option is perhaps the most straightforward and is the one we employ here.

Example 3.2 Recall the matrix $A_{(0)}$ in Example 3.1 on the right, whose second and third columns were $\mathcal{C}$ inconsistent. Applying Algorithm 3.1 with column generation as above, all 3 columns generated by the SAT solver were rejected by the $\mathcal{EL}$-solver, so no column could be generated that minimised the objective function in $\langle 0, 0\rangle$. Therefore the corresponding $\mathcal{EL}$PA-knowledge base is unsatisfiable. □

Theorem 3.3 *Algorithm 3.1 with column generation as above is correct and always terminates.*

4 Conclusions and Further Work

We have introduced the notion of $\mathcal{EL}$PA-knowledge bases and its satisfiability problem, and we have shown that the problem has a finite version that can be tacked by algorithms that resemble PSAT solvers. We have also provided complexity upper bounds for these algorithms.

Algorithm 3.1 has the theoretical possibility of generating an exponential number of steps. It remains an open problem to find an example in which such an exponential number of steps occur. It also remains an open problem whether a polynomial time algorithm exists for ELPA-satisfiability. Our plan for future work is to investigate the practical behavior of our algorithms, and to explore logics that allow for probability over TBoxes and for stochastic independence.

References

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI05, 19th International Joint Conference on Artificial Intelligence*, pages 364–369, 2005.

[Bra04] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, gci axioms, and - what else? In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, pages 298–302, 2004.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, 1997.

[CP09] FG Cozman and RB Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[Eck93] J. Eckhoff. Helly, Radon, and Caratheodory type theorems. In P. M. Gruber and J. M. Wills, editors, *Handbook of Convex Geometry*, pages 389–448. Elsevier Science Publishers, 1993.

[FB11] Marcelo Finger and Glauber De Bona. Probabilistic satisfiability: Logic-based algorithms and phase transition. In *To appear on IJCAI2011*, 2011.

[GKP88] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *J. of Complexity*, 4(1):1–11, 1988.

[HJ00] P. Hansen and B. Jaumard. Probabilistic satisfiability. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol.5*. Springer, 2000.

[JFSS06] B. Jaumard, A. Fortin, I. Shahriar, and R Sultana. First order probabilistic logic. In *NAFIPS 2006*, pages 341–346, 2006.

[KP90] D. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *AMAI*, 1:189–205, 1990.

[LS10] Carsten Lutz and Lutz Schröder. Probabilistic description logics for subjective uncertainty. In *KR 2010, 12th International Conference of Knowledge Representation and Reasoning*. AAAI Press, 2010.

[Luk08] T Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

[GM10] O. Gries, and R. Möller. Gibbs sampling in probabilistic description logics with deterministic dependencies. *First International Workshop on Uncertainty in Description Logics (Uni-DL)*, 2010.

[CL06] P. C. G. Costa and K. B. Laskey. PR-OWL: A framework for probabilistic ontologies. *Conf. on Formal Ontology in Information Systems*, 2006.

[DFL08] C. D'Amato, N. Fanizzi, and T. Lukasiewicz. Tractable reasoning with Bayesian description logics. *Int. Conf. on Scalable Uncertainty Management* (LNAI 5291), pages 146–159, 2008.

[DPP06] Z. Ding, Y. Peng, and R. Pan. BayesOWL: Uncertainty modeling in semantic web ontologies. *Soft Computing in Ontologies and Semantic Web*, pages 3–29. Springer, Berlin/Heidelberg, 2006.

[DS05] M. Dürig and T. Studer. Probabilistic ABox reasoning: preliminary results. *Description Logics*,pages104–111,2005.

[GT07] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning.* MIT Press, 2007.

[Hal03] J. Y. Halpern. *Reasoning about Uncertainty.* MIT Press, Cambridge, Massachusetts, 2003.

[Hei94] J. Heinsohn. Probabilistic description logics. *Conf. Uncertainty in AI*, page 311-318, 1994.

[Jae94] M. Jaeger. Probabilistic reasoning in terminological logics. *Principles of Knowledge Representation*, pages 461–472, 1994.

[KLP97] D. Koller, A. Y. Levy, and A. Pfeffer. P-CLASSIC: A tractable probablistic description logic. *AAAI Conf. on AI*, pages 390–397, 1997.

[LS08] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, November 2008.

[Seb94] F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. *17th Conf. on Research and Development in Information Retrieval*, pages 122–130, Dublin, Ireland, 1994. Springer-Verlag.

Quelo: an ontology-driven query interface

Enrico Franconi, Paolo Guagliardo, Sergio Tessaris, and Marco Trevisan

franconi@inf.unibz.it, paolo.guagliardo@stud-inf.unibz.it, tessaris@inf.unibz.it, evenjn@gmail.com

KRDB Research Centre, Free University of Bozen-Bolzano, Italy

Abstract. In this paper we present a formal framework and tool supporting the user in the task of formulating a precise query – which best captures their information needs – even in the case of complete ignorance of the vocabulary of the underlying information system holding the data. Our intelligent interface is driven by means of appropriate automated reasoning techniques over an ontology describing the domain of the data in the information system.
We will define what a query is and how it is internally represented, which operations are available to the user in order to modify the query and how contextual feedback is provided about it presenting only relevant pieces of information. We will then describe the elements that constitute the query interface available to the user, providing visual access to the underlying reasoning services and operations for query manipulation. Lastly, we will define a suitable representation in "linear form", starting from which the query can be more easily expressed in natural language.

1 Introduction

Recent research showed that adopting formal ontologies as a means for accessing heterogeneous data sources has many benefits, in that not only does it provide a uniform and flexible approach to integrating and describing such sources, but it can also support the final user in querying them, thus improving the usability of the integrated system.

We introduce a framework that enables access to heterogeneous data sources by means of a conceptual schema and supports the users in the task of formulating a precise query over it. In describing a specific domain, the ontology defines a vocabulary which is often richer than the logical schema of the underlying data and usually closer to the user's own vocabulary. The ontology can thus be effectively exploited by the user in order to formulate a query that best captures their information need. The user is constantly guided and assisted in this task by an intuitive visual interface, whose intelligence is dynamically driven by reasoning over the ontology. The inferences drawn on the conceptual schema help the user in choosing what is more appropriate with respect to their information need, restricting the possible choices to only those parts of the ontology which are relevant and meaningful in a given context.

The most powerful and innovative feature of our framework lies in the fact that not only do not users need to be aware of the underlying organisation of the data, but they are also not required to have any specific knowledge of the vocabulary used in the ontology. In fact, such knowledge can be gradually acquired by using the tool itself, gaining confidence with both the vocabulary and the ontology. Users may also decide to just explore the ontology without actually querying the information system, with the aim of discovering general information about the modelled domain.

Another important aspect is that only queries that are logically consistent with the context and the constraints imposed by the ontology can be formulated, since contradictory or redundant pieces of information are not presented to the user at all. This makes user's choices clearer and simpler, by ruling out irrelevant information that might be distracting and even generate confusion. Furthermore, it also eliminates the often frustrating and time-consuming process of finding the right combination of parts that together constitute a meaningful query. For this reason, the user is free to explore the ontology without the worry of making a “wrong” choice at some point and can thus concentrate on expressing their information need at best.

Queries can be specified through a refinement process consisting in the iteration of few basic operations: the user first specifies an initial request starting with generic terms, then refines or deletes some of the previously added terms or introduces new ones, and iterates the process until the resulting query satisfies their information need. The available operations on the current query include addition, substitution and deletion of pieces of information, and all of them are supported by the reasoning services running over the ontology.

In this paper we present a complete and coherent view of the **Quelo** tool, whose basic ideas have been already sketched in the past ([4; 1; 2; 3; 6]). Quelo relies on a web-based client-server architecture consisting of three components:

1. the tool logic, responsible of “reasoning” over the ontology in order to provide only relevant information w.r.t. the current query;
2. the natural language generation (NLG) engine, that given a query and a lexicalisation map for the ontology produces an English sentence; the lexicon is automatically generated from the ontology;
3. the user interface (GUI), that provides visual access to the query and editing facilities for it, allowing to interact with the reasoning sub-system while benefiting from the services of the NLG engine.

2 The Abstract Functionality

In this section we describe the behaviour of the tool using a generic representation based on an abstract user interface. Consider a scenario in which we have a conceptual schema, say an OWL ontology, we know nothing about. In such situation, the tool reveals to be particularly useful in that it allows to discover information about the ontology and the modelled domain, even when its vocabulary is completely ignored. What we call *intensional navigation* of the ontology is the process of building a query, starting from a very general request which is then refined by adding or deleting constraints according to the user’s information need. In our abstract representation, the default initial query generically asks for some “thing”. Four operations are available for manipulating the query: *add* for the addition of new terms and relations; *substitute* for replacing a portion of the query with a more general, equivalent or more specific term; *delete* for discarding parts of the query; and *weaken* for making a portion of the query as general as possible.

The first step in the refinement of our query consists in being more specific about what we are looking for. This can be achieved by selecting something within the query and asking for a substitution. In our example, we tick the check-box associated with the term **Thing** and then press the **Substitute** button. As shown in Figure 1, we are presented with a three-part menu listing all the possible substitutions available for the selected portion of the query: terms that appear at the top are more general than the selection, the ones in the middle are equivalent, while those at the bottom are more specific. Moreover, these terms are organised in sub-menus according to the taxonomic information defined in the ontology. Thus, we

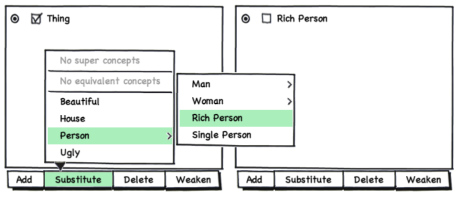


Fig. 1: Example of specialisation

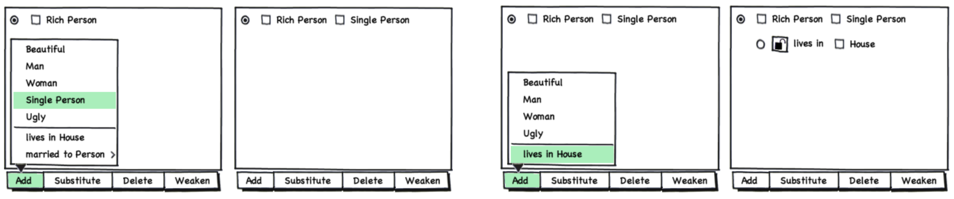


Fig. 2: Addition of (a) a new term and (b) a new relation

can easily navigate the different options choosing the desired level of detail for the substitution. In our case, instead of **Person** we choose the further specific term **Rich Person** as a replacement for the selection, resulting in the query visible on the right side of Figure 1.

Another way of modifying a query is to add constraints in the form of new terms or relations. As shown in Figure 2, upon clicking on the add button a two-part menu is displayed, containing suitable terms and relations that can be safely added to the query. Terms are shown in the upper part of the menu, while relations in the lower one. The chosen item is inserted in a specific place of the query, that can be selected by means of the radio button present in each line. Figure 2a shows how the new term **Single Person** is added to the first (and only) line of the query, while Figure 2b shows how adding a relation results in the creation of a new line, indented w.r.t. the first one and consisting of the name of the relation **lives in** followed by the label **House** associated with its range. Observe that the menu of Figure 2b, compared to that of Figure 2a, does not include the term **Single Person** and the relation **married to Person** as possible options. In fact, the former is already present in the query, thus it would be redundant to propose it again; the latter became incompatible with the query due to the addition of the previous term, and this means that in our ontology a person who is rich and single (or perhaps just single) cannot be married to anyone.

A query can be made more general or "weaker" in a variety of ways, one of which is the substitution with a more general term. Other possibilities are given by deletion and weakening, both of which remove selected elements from the query but with distinct approaches and outcomes. The difference between them is shown in Figure 3: while in Figure 3a deleting the selected portion causes the second row of the query to disappear, in Figure 3b weakening the same portion preserves the row, although the term **House** is replaced with the generic term **Thing**.

Suppose that our ontology states that a rich man who is married to a beautiful woman and lives in a beautiful house is indeed a lucky person. As a result of the substitution shown in Figure 4, where the whole query is replaced with the (more general) term **Lucky Person**, the second line disappears. In some situations this "side-effect" is undesired, because we would perhaps like to operate on that part of the query later on. The closed padlock icon visible in the third line of the query indicates that that line is protected against such an "accidental" deletion and would not inadvertently disappear as the result of the substitution. However, note that locked portions of the query are still fully affected by explicit deletion.

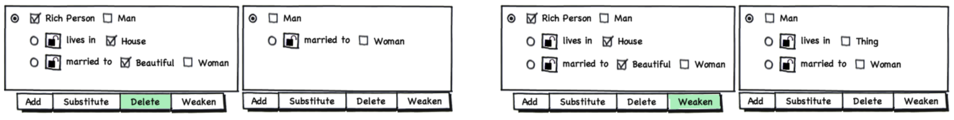


Fig. 3: Example of (a) deletion and (b) weakening

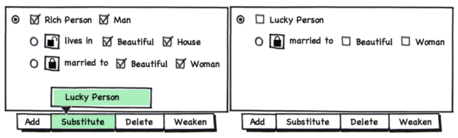


Fig. 4: Preventing side-effects of substitution

3 The Reasoning module

The framework and its functional API are defined in a formal way; here, we concisely summarise the main definitions introduced in [7] and formally prove the most important of tool's properties, namely that it generates only "meaningful" queries.

3.1 Formal framework

From the point of view of the reasoning sub-system, a query is a labelled tree where each node corresponds to a variable and is associated with a set of concept names from the ontology, while each edge is labelled with a role name.

Let **N** be a countably infinite set of node names, **C** be a finite set of concept names and **R** a finite set of role names, and let **N**, **C** and **R** be pairwise disjoint. A *query* Q is a quintuple $\langle V, E, o, \mathcal{V}, \mathcal{E}\rangle$ in which (V, E) is a directed tree rooted in $o \in V$, with set of nodes $V \subseteq \mathbf{N}$ and with set of edges $E \subseteq V \times V$; $\mathcal{V}$ is a total function, called *node-labelling function*, associating each node with either a non-empty set of concept names or with the singleton $\{\top\}$; and $\mathcal{E}$ is called the *edge-labelling function* that associates each edge with a role name. A query consisting of exactly one node, whose set of labels is a singleton, is called *atomic*. For an edge $e = \langle x, y\rangle$, we indicate its initial node x with $\mathsf{init}(e)$ and its terminal node with $\mathsf{ter}(e)$. Given queries S and Q, we say that S is a *subquery* of Q, and write $S \subseteq Q$, iff $V(S) \subseteq V(Q)$, $E(S) \subseteq E(Q)$, each node $n \in V(S)$ is s.t. $\mathcal{V}_S(n) \subseteq \mathcal{V}_Q(n)$ and every edge $e \in E(S)$ is such that $\mathcal{E}_S(e) = \mathcal{E}_Q(e)$. We say that S is a *complete subquery* of Q (in symbols $S \subseteqq Q$) if it also holds that, for every $n \in V(S)$, $\mathcal{V}_S(n) \supseteq \mathcal{V}_Q(n)$ and every descendant of o_S in Q is a node in S. A *selection* within a query Q is a subquery S of Q, which is called *simple* if $S \subseteqq Q$ or S consists of exactly one node, namely its root o_S, such that $\mathcal{V}_S(o_S)$ is a singleton or is equal to $\mathcal{V}_Q(o_S)$. Every selection S within a query Q partitions the nodes of Q into *selected*, which belong to $V(S)$, and *unselected*, belonging to $V(Q) \setminus V(S)$. The selected nodes can be further partitioned into *totally selected*, having all of their labels selected, and *partially selected*, which have some, but not all, of their labels selected. An example of query as represented is shown in Figure 5, which also shows the compact graphical notation we use for representing a selection within a query: selected nodes are drawn using a double circle and selected labels within each of them are underlined.

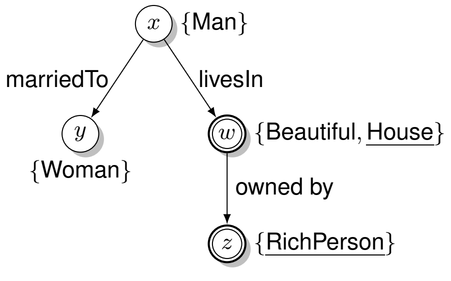


Fig. 5: Example of query and a selection within it

The *weakening* of a query Q w.r.t. a selection S within Q is the query $Q \ominus S$ obtained from Q by replacing its node-labelling function $\mathcal{V}_Q$ with a function that associates each totally selected node with $\{\top\}$, each partially selected node n with $\mathcal{V}_Q(n) \setminus \mathcal{V}_S(n)$ and each unselected node m with $\mathcal{V}_Q(m)$.

The last notion we introduce is that of *sticky edges*, which are edges that can only be deleted explicitly (that is, when performing a deletion), but never implicitly (e.g., as the consequence of a substitution). Sticky edges are closed w.r.t. the tree structure of the query, that is, when an edge e is sticky, then all the edges in the path from the root of the query to $\mathsf{ter}(e)$ are such. The meaning and importance of sticky edges will become more clear in the next section, where we introduce and describe the two operations delete and substitute. For the moment, sticky edges can be simply understood as immutable (to some extent) pieces of information within a query, which are not modified as a "side effect" of an operation not directly intended to do so.

3.2 Functional API

To draw the inferences that are at the basis of the query formulation tasks, we express a query as a concept of some description logic (DL) language, for which the containment test of two conjunctive queries is decidable and available as a reasoning service. In what follows, we assume the existence of an underlying knowledge base $\mathcal{K}$ in such a DL language $\mathcal{L}$ over $\mathbf{C}$ and $\mathbf{R}$. We say that C is a sub-concept of (or *subsumes*) D in $\mathcal{K}$, and write $C \sqsubseteq_{\mathcal{K}} D$, iff $\mathcal{K} \models C \sqsubseteq D$, in which case we also say that D is a super-concept of (or *is subsumed by*) C. Two concepts C and D are *equivalent* in $\mathcal{K}$, written $C \equiv_{\mathcal{K}} D$, iff one subsumes the other and vice versa. For two concept names c_1 and c_2 we say that c_1 is a *direct sub-concept* of c_2 (and that c_2 is a *direct super-concept* of c_1) iff c_1 subsumes c_2 and there is no $c \in \mathbf{C}$ equivalent to neither c_1 nor c_2 and such that $c_1 \sqsubseteq_{\mathcal{K}} c \sqsubseteq_{\mathcal{K}} c_2$.

Before introducing the functional API, let us first give some preliminary definitions. Given a query Q and $n \in V(Q)$, the operation $\mathsf{roll\text{-}up}(Q, n)$ translates Q into an $\mathcal{L}$-concept w.r.t. n and it is defined as $\mathsf{enc\text{-}rollup}(Q, n, n)$, where enc-rollup is the recursive procedure described in Algorithm 1. We use $\mathsf{roll\text{-}up}(Q)$ as an abbreviation for $\mathsf{roll\text{-}up}(Q, o)$, where o is the root of Q. The concept $\mathsf{roll\text{-}up}(Q, n)$ is called the *context* of Q w.r.t. n, expressing the informative content of Q from the point of view of a specific node, which we call the *focus*. Queries Q_1 and Q_2 are *equivalent*, in symbols $Q_1 \equiv Q_2$, iff $\mathsf{roll\text{-}up}(Q_1) \equiv_{\mathcal{K}} \mathsf{roll\text{-}up}(Q_2)$. We say that a query Q over a consistent knowledge base $\mathcal{K}$ is *satisfiable* iff its roll-up is such in $\mathcal{K}$ (that is, $\mathcal{K} \not\models \mathsf{roll\text{-}up}(Q) \sqsubseteq \bot$).

The functional API of the tool is structured in two main parts:

- the underlying reasoning services, consisting of the operations getComp, getRel, getSupers, getEquiv, getSubs;
- the operations for query manipulation, including addRel, addComp, weaken, substitute and delete.

Given a query Q and a node n, we say that a concept name c is *compatible* with Q focused in n iff $c \sqcap \mathsf{roll\text{-}up}(Q, n) \not\sqsubseteq \bot$, while a role name r is such iff $\exists r^-. \mathsf{roll\text{-}up}(Q, n) \not\sqsubseteq \bot$. The operation $\mathsf{getComp}(Q, n)$ returns a directed acyclic graph (DAG) G, whose nodes are all the concept names that are compatible with Q focused in n and that are neither sub- nor super-concepts of $\mathsf{roll\text{-}up}(Q, n)$, and whose edges are all the pairs of concept names $c_1, c_2 \in V(G)$ such that c_1 is a direct sub-concept of c_2. In other words, the output of getComp is a taxonomy of concept names which are compatible with the query and not in hierarchy with the context. The operation $\mathsf{getRel}(Q, n)$ returns a DAG G, whose nodes are all the pairs $\langle r, c\rangle$ of role names and concept names such that r is compatible with Q focused in n and c is a sub- or a super-concept of $\exists r^-. \mathsf{roll\text{-}up}(Q, n)$, and whose edges are the pairs $\langle\langle r, c_1\rangle, \langle r, c_2\rangle\rangle \in V(G) \times V(G)$ such that c_1 is a direct super-concept of c_2.

Let S be a selection within a query Q. Then, the operations getSupers, getEquiv and getSubs return the concept names that are more general than, equivalent to and more specific

Algorithm 1 Calculate enc-rollup(Q, n, m)

```
Input: a query $Q$ and two nodes $n, m \in V(Q)$
Output: a concept $C$ expressing $Q$ in the description logics language $\mathcal{L}$
 1: $C \leftarrow c$, for some $c \in \mathcal{V}(n)$
 2: for all $x \in \mathcal{V}(n)$ such that $x \neq c$ do
 3:     $C \leftarrow C \sqcap x$
 4: end for
 5: for all children $x$ of $n$ in $Q$ such that $x \neq m$ do
 6:     $R \leftarrow \mathcal{E}(\langle n, x\rangle)$
 7:     $C \leftarrow C \sqcap \exists R$ . enc-rollup$(Q, x, n)$
 8: end for
 9: if $n \neq o$ then
10:     Let $p$ be the parent node of $n$ in $Q$
11:     if $p \neq m$ then
12:         $R \leftarrow \mathcal{E}(\langle p, n\rangle)$
13:         $C \leftarrow C \sqcap \exists R^{-}$ . enc-rollup$(Q, p, n)$
14:     end if
15: end if
16: return $C$
```

than roll-up(S), respectively. Moreover, the concept names in the output of getSubs(Q, S) are additionally required to be compatible with Q focused in the root of S.

Let Q be a query and n a focus node. For a concept name c in the output of getComp(Q, n), the operation addComp adds c to $\mathcal{V}(n)$. More precisely, the result of addComp(Q, n, c) is the query Q' obtained from Q by replacing its node-labelling function $\mathcal{V}$ with $\mathcal{V}' := \mathcal{V}\,[n \mapsto \mathcal{V}\,(n) \cup \{c\}]$. For a pair $\langle r, c\rangle$ in the output of getRel(Q, n), the operation addRel creates a new node n' such that $\mathcal{V}(m) = \{c\}$ and an edge $e = \langle n, m\rangle$ with $\mathcal{E}(e) = r$.

Let Q and R be queries and $\widetilde{E}$ be a set of sticky edges. Then, the operation prune deletes from Q the maximal number of non-root nodes, having no incoming sticky edge (if any) and associated with the same concept names both in R and Q, such that the result is still a query.

Let S be a selection within a query Q and let $\widetilde{E}$ be a set of sticky edges. We define weaken(Q, S) as $Q \ominus S$ and

$$\mathsf{delete}(Q, S, \widetilde{E}) := \mathsf{prune}\big(\mathsf{weaken}(Q, S), R, \widetilde{E}\big) \ ,$$

where R is the query obtained from S by replacing $\mathcal{V}_S$ with the function on $V(S)$ associating each node n that is both in Q and S with $\mathcal{V}_Q(n) \cap \mathcal{V}_S(S)$ if such intersection is non-empty and with $\{\top\}$ otherwise, and each other node m of S with $\mathcal{V}_S(m)$. The last operation we introduce is "substitution" which, for a concept name c in the output of getSupers (*generalisation*) or getEquiv or getSubs (*specialisation*), is defined as follows:

$$\mathsf{substitute}(Q, S, \widetilde{E}, c) := \mathsf{delete}(Q', S, \widetilde{E}) \ ,$$

where Q' is the query obtained from Q by adding c to the set of concept names associated with the root of S.

3.3 Properties of the framework

We will now formally state that, starting from an atomic query that is satisfiable, the query obtained by means of the operations in the tool's functional API is satisfiable. In order to do that, we first prove that the operations for query manipulation preserve satisfiability, i.e., the application of each of them to a satisfiable query results in a query that is satisfiable.

Lemma 1. *Each of the operations* addComp, substitute, addRel, weaken *and* delete *preserves query satisfiability.*

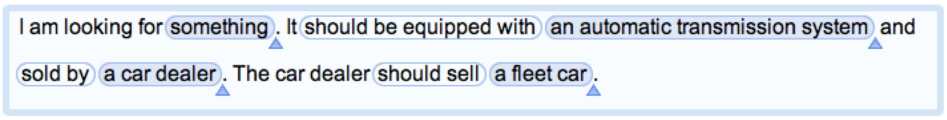
The fundamental property of the tool is then proved by means of a simple induction.

Theorem 1. *The query obtained from an initial satisfiable atomic query through a finite sequence of applications of the operations* addComp, addRel, substitute, weaken *and* delete *is satisfiable.*

4 The user interface

In this section we describe the basic elements of the concrete UI Quelo, based on natural language generation. In the UI, the query is represented as a continuous string of natural language text, composed of a sequence of coherent text constituents called *spans*. Each of the tags occurring in the query is associated with a span by means of an injective mapping. As for each edge there is one and only one corresponding edge tag, if a span is associated with the tag of an edge we simply say that the span is associated with that edge.

The English sentence representing the query in the UI is generated by the NLG subsystem, which will be described in the next section. Here is an example of the textual rendering of the query in natural language as displayed by the UI:

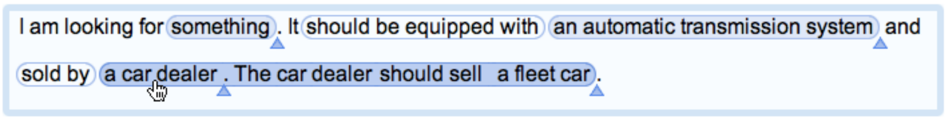

I am looking for something. It should be equipped with an automatic transmission system and sold by a car dealer. The car dealer should sell a fleet car.

4.1 Hovering

In graphical user interfaces terminology, the user *hovers* on a graphic element whenever the mouse cursor moves from some point outside the element to some point inside the element. In normal conditions, as the user hovers on the query, the system gives visual hints about its structure:

– hovering on the span associated with a tag of some node n causes the span to become *lightly highlighted*, along with all the spans associated with the tags occurring in the complete subquery rooted in n;
– hovering on the span associated with the tag of some edge e causes that span and all the spans associated with the elements of $\mathsf{tags}(\mathsf{ter}(e))$ to become lightly highlighted.

The highlighting is such that spans associated with different tags are visualised as distinct, even when adjacent. One way of obtaining this kind of effect is, for instance, by rounding the corners of the highlighted rectangular area around each span. The only case in which highlighting on hovering does not trigger is when a menu is being displayed.


I am looking for something. It should be equipped with an automatic transmission system and sold by a car dealer. The car dealer should sell a fleet car.

Associated with each node of the query is a button, called the *add-button*, which is located below the text baseline immediately after the rightmost span associated with a tag of that node. Hovering on the add-button of some node lightly highlights all the spans associated with tags of that node.

4.2 Selection

The UI provides facilities to easily select portions of the query. A simple selection can be directly specified by clicking on the span associated with a tag of some node n in one of the following ways:

– a single click results in an atomic selection, highlighting only the span on which the click occurred;
– a double click results in a node selection, highlighting all the spans associated with the elements of tags(n);
– a triple click results in a complete selection, highlighting all the spans in the complete subquery rooted in n.

A selection can be *cleared* by clicking on an area of the UI where clicking does not have any other effect (e.g., on the white space between the lines of text representing the query, or on a span that is not associated with any tag). Clearing a selection results in an *empty selection*.

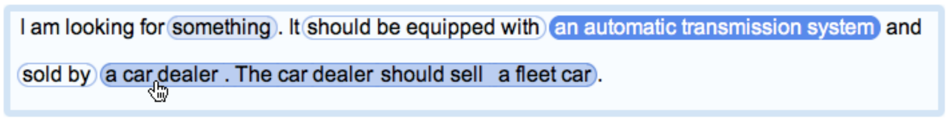
Observe that when a node has only one node label, an atomic selection on that node happens to be also a node selection, and when a node has no children, a node selection is also a complete selection. Thus, an atomic selection on a node having only one label and no children is also a node selection as well as a complete selection.

We consider atomic selections to have the lowest priority and complete selections the highest, and when a simple selection belongs to more than one class, it is considered to be only of the type with higher priority. Furthermore, whenever a double click would result in the same kind of selection a single click would, it yields a complete selection instead.

A *complex selection* (non-simple) is obtained from an empty or simple selection by control-clicking on additional spans associated with node tags, which are consequently included in the existing selection. Note that a complex selection can be *disconnected*, in the sense that it is not a well-formed subquery from the formal point of view, because there might be two selected nodes that are not connected by an edge.

From the graphic point of view, spans associated with tags in a selection are highlighted in a stronger way (e.g., a darker color) than they are when highlighted because of hovering and, unlike the highlighting effect triggered by hovering, it is not possible to distinguish between adjacent selected spans associated with the same node. When the selection includes one or more paths between nodes (that is, all of the nodes in a path within the query are selected), spans associated with edge tags are also highlighted.

The visual appearance of the spans associated with tags of selected nodes or edges does not change as the result of an hovering event, as shown here:


I am looking for something . It should be equipped with an automatic transmission system and sold by a car dealer . The car dealer should sell a fleet car .

Moreover, as the reader might already have noticed, when a non-empty selection is present, the add-buttons become invisible without changing the layout of the text.

4.3 Addition

The query logic sub-system provides two operations, namely addComp and addRel, for refining a query through the addition of compatible terms and relations to a focus node. The UI makes these operations available to the user by means of a pop-up menu, activated by clicking on the add-button of a node which is set as the focus.

The menu contains a list of suitable arguments for the invocation of either addComp or addRel. The menu entries are concept names and pairs consisting of a role name and a concept name, which are obtained from the output of the Quelo operations getComp and getRel w.r.t. the current query and focus. In particular, for a query Q focused in n, the menu is populated with the nodes in the graph resulting from disjoint union of the output graphs of getComp(Q, n) and getRel(Q, n), arranged in the following way:

– nodes with no incoming edge populate a menu of level 0, which is the *topmost* menu, where entries corresponding to concept names are listed before entries associated with pairs of concept/role names.

– for each node n in a menu at level k, all the nodes that are reachable from n in one step populate a sub-menu at level $k + 1$ associated with entry n.

The actual items shown to the user in the above menu structure are natural language descriptions of the node-entries (either concept names or atomic concept/role pairs) generated by the NLG sub-system.

Some of the items come with an icon on their left: an upward-pointed (resp., downward-pointed) triangle is displayed for concept names (resp., role/concept pairs) indicating that the option is associated with a nested sub-menu containing more specific (resp., more generic) options of the same type. Hovering on any of these options opens the pop-up menu associated with that item and displayed next to it.

Clicking on any of the elements in the menu triggers the invocation of either addComp or addRel, according to whether the clicked item is associated with a concept name or a concept/role pair, respectively. Upon clicking, the menus disappear and the UI updates its representation of the query after the necessary changes are performed by the Quelo sub-system.

4.4 Weakening and Deletion

The user can *weaken* (respectively, *delete*) a selected portion of the query by pressing the backspace (resp., delete) key on the keyboard, which invokes the Quelo operation weaken (resp., delete) with the current query and selection as input arguments. Upon weakening (resp., deletion), the selection is cleared and the UI updates its representation to reflect the changes in the query.

Note that the operations weaken and delete, as defined in our functional API, cannot directly handle a disconnected complex selection. However, such a selection can be decomposed by the UI in a series of connected selections that are then suitable for the actual invocation of the two operations.

Observe that in some cases deletion produces the same result as weakening (e.g., for a node selection rooted in a non-leaf node).

4.5 Substitution

The Quelo sub-system provides the operation substitute in order to allow the substitution of a selection within the query with a more generic, equivalent or more specific term. The UI makes this operation available to the user: upon long-clicking on a selected portion (i.e., strongly highlighted) of the query a pop-up menu is displayed, listing all the possible terms with which the selection can be replaced.

Such a menu is populated with concept names that are more general than, equivalent to and more specific than the selection and that are retrieved from the Quelo sub-system by means of the operations getSupers, getEquiv and getSubs, respectively. More general terms are shown at the top of the list, equivalent terms in the middle and more specific terms at the bottom. At the left of each item an icon is shown: an upward-pointing triangle for more general terms, a square for equivalent terms and a downward-pointing triangle for more specific terms.

The substitution menu has a similar hierarchical structure as the menu for addition, reflecting the taxonomic information in the output graphs of the operations getSupers, getEquiv and getSubs. In particular, some (possibly none) of the more general terms might be further generalised, in which case hovering on one such item triggers a sub-menu containing its direct super-concepts; similarly, if some of the more specific terms can be further specialised, then hovering on one such item triggers a sub-menu containing its direct sub-concepts that are compatible with the query. The same rules for further generalisation/specialisation apply to the items in the sub-menus, while equivalent terms (if any) cannot be further generalised nor specialised.

As in the case of addition, the actual items shown to the user in the substitution menu are natural language descriptions generated by the NLG sub-system, rather than bare concept names. Clicking on any of given options triggers the invocation of substitute with the current selection and the concept name associated with the clicked item as input arguments. The selection is then cleared and the UI updates the representation of the query after the selected elements have been replaced with the chosen term.

Observe that the operation substitute, as defined in our functional API, cannot deal with disconnected complex selection and, unlike the case of weakening and deletion, the problem cannot be overcome by converting such selection in a series of connected ones. This is due to the fact that substitution relies on the roll-up of the input selection itself, which is thus required to be tree-shaped (i.e., connected). For this reason, in the presence of a disconnected selection, the substitution operation is disabled.

5 Natural language rendering

The natural language interface of the tool masks the composition of a precise query as the composition of English text describing the equivalent information needs. Interfaces following this paradigm are known as “menu-based natural language interfaces to databases” or“conceptual authoring” (see,most notably, [8]). As we have seen before, the users of such systems edit a query by composing fragments of generated natural language provided by the system through *contextual* menus. In [6] we describe how the natural language rendering of a query is achieved.

We start by defining a particular linear form of the query that satisfies certain constraints, necessary to represent the elements of the query using a linear medium, that is, text. The constraints are enforced at the API level to ensure that different graphical user interfaces represent the query in a homologous way. Moreover, a consistent ordering of the query elements needs to be preserved during the operations for query manipulation to avoid confusing the end user. The linearised version of the query is then used as a guide for the language generation performed by the tool’s NLG engine.

The natural language interface (NLI) of the tool relies on a natural language generation (NLG) system to produce the textual representation of the query, following an idea first presented in [11] and lately refined in [8].

For the tool’s NLI to work with a specific knowledge base (KB) a lexicon and a template map must be provided for it. Devising these resources requires an understanding of both the domain of interest and basic linguistic notions such as verb tenses, noun genders and countability. To ease the burden of developing these resources from scratch, we let the system generate them automatically. This technique follows an approach to domain independent generation proposed in [10], after the learning of a rich corpus of relations. The functionality we implemented allows to produce all the resources necessary to configure our NLI for use with a new KB, using as a source of data the ontology itself. It has to be noted that the process is not completely reliable, therefore system engineers must review the result and make the necessary corrections.

6 Future Work

The framework and functional API presented in this paper consider only the addition of new relations to the query, but they could be extended to deal with attributes (i.e., properties relating a concept to a datatype) as well. The only difference with the current framework would be that a node associated with a datatype (i.e., the “range” of the attribute) cannot be the focus of a query for operations other than deletion. This basically means that such a node is always a leaf of the query tree and the only operation allowed on it is deletion. Then, since a node of this kind cannot be refined by adding a compatible term or attaching a new property, the query is never rolled-up with respect to it, thus avoiding the nonsensical eventuality that an edge associated with an attribute has to be inverted (going from the datatype to the subject). Though apparently simple, allowing for attributes poses some interesting questions, that we are currently investigating, in order to deal with concrete values from the point of view of reasoning.

Another direction we plan to pursue is that of continuing the series of experiments already carried on (see [1; 2]), in order to evaluate the usability of the tool and its complexity of use from the user's point of view. In particular, we are interested in determining how difficult it is for the user to formulate queries using the tool and to understand the results.

An online fully functional demonstrator of Quelo is freely accessible at:

`http://krdbapp.inf.unibz.it:8080/quelo/`

References

1. T. Catarci, P. Dongilli, T. Di Mascio, E. Franconi, G. Santucci, and S. Tessaris. An ontology based visual tool for query formulation support. In *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, 2004.
2. T. Catarci, P. Dongilli, T. Di Mascio, E. Franconi, G. Santucci, and S. Tessaris. Usability evaluation tests in the SeWAsIE (SEmantic Webs and AgentS in Integrated Economies) project. In *Proceedings of the 11th International Conference on Human-Computer Interaction (HCII 2005)*, 2005.
3. P. Dongilli and E. Franconi. An Intelligent Query Interface with Natural Language Support. In *Proc. of the 19th Int. Florida Artificial Intelligence Research Society Conference (FLAIRS 2006)*, Melbourne Beach, Florida, USA, May 2006.
4. P. Dongilli, E. Franconi, and S. Tessaris. Semantics driven support for query formulation. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, 2004.
5. N. Drummond, M. Horridge, R. Stevens, C. Wroe, and S. Sampaio. Pizza ontology. The University of Manchester. `http://www.co-ode.org/ontologies/pizza/`.
6. E. Franconi, P. Guagliardo, and M. Trevisan. An intelligent query interface based on ontology navigation. In *Proc. of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010)*, Feb. 2010.
7. P. Guagliardo. Theoretical foundations of an ontology-based visual tool for query formulation support. Technical Report KRDB09-5, KRDB Research Centre, Free University of Bozen-Bolzano. `http://www.inf.unibz.it/krdb/pub/TR/KRDB09-05.pdf`, October 2009.
8. C. Hallett, D. Scott, and R. Power. Composing questions through conceptual authoring. *Computational Linguistics*, 33(1):105–133, 2007.
9. Ordnance Survey - Great Britain's national mapping agency. `http://www.ordnancesurvey.co.uk/oswebsite/ontology/`.
10. X. Sun and C. Mellish. Domain independent sentence generation from RDF representations for the Semantic Web. In *Proc. ECAI'06 Combined Workshop on Language-Enhanced Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems*, 2006.
11. H. R. Tennant, K. M. Ross, R. M. Saenz, C. W. Thompson, and J. R. Miller. Menu-based natural language understanding. In *Proc. 21st Annual Meeting of the Association for Computational Linguistics*, pages 151–158. Association for Computational Linguistics, 1983.
12. M. Trevisan. A portable menu-guided natural language interface to knowledge bases. Master's thesis, University of Groningen, 2009.
13. D. Tufis and O. Mason. Tagging Romanian texts: a case study for QTAG, a language independent probabilistic tagger. *Proc. 1st Int. Conf. on Language Resources and Evaluation (LREC'98)*, pages 589–596, 1998.

A Connection Method for the Description Logic $\mathcal{ALC}$

Fred Freitas

Informatics Center, Federal University of Pernambuco (CIn - UFPE), Brazil
fred@cin.ufpe.br

1 Introduction

The problem of reasoning over ontologies written in Description Logic (DL) [1] has received strong interest, particularly after the Semantic Web inception. The necessity of creating reasoners to deal with OWL (Ontology Web Language) [2], in its various versions, posed interesting research questions for inference systems, making the problem earn the reputation of being complex, whose users see solutions as black boxes. Consequently, not many inference systems became well known to the public; indeed, tableaux methods [1] took over the field. Apart from the reasoning algorithms, many other practical issues have been brought about, stemming from the natural features of the Web, that ask for certain requirements for inferencing. Among them, the use of memory is certainly an important asset for a good reasoning performance. Therefore, methods that do not require a huge amount of memory may be promising.

This paper tries to tackle this issue by adapting a reasoning method that makes a parsimonious usage of memory. The inference system proposed here is based on the connection calculus [3], which is a clear and effective inference method applied successfully over first order logic (FOL). Its main features meet with the demands: it keeps only one copy of each logical sentence in memory and it does not derive new sentences from the stored ones. The formal system, the $\mathcal{ALC}$ Connection Method ($\mathcal{ALC}$ CM), displays some desirable features for a DL reasoner,: it is able to perform all DL inference services, namely, subsumption, unsatisfiability (or inconsistency), equivalence and disjointness. As the most widespread DL tableaux systems, $\mathcal{ALC}$ CM reduces them to subsumption or validity - the dual of unsatisfiability, used in most DL systems (interested readers should check that information at [7]).

The rest of the article is organized as follows. Section 2 defines one of the most basic description logic languages, $\mathcal{ALC}$ (Attributive Concept Language with Complements) [1], together with the disjunctive normal formal used by the method. $\mathcal{ALC}$ was chosen as the starting of the work, because it constitutes the foundation of many other Description Logics. Section 3 brings a proposed $\mathcal{ALC}$ ontologies' matrix normal form that fits to the connection method. Section 4 describes the $\mathcal{ALC}$ adapted connection method and states the key logical properties for inference systems (soundness, completeness and termination). Section 5 brings discussion in the light of representation and reasoning, and section 6 presents future work and conclusions.

2 The Description Logic $\mathcal{ALC}$

Description Logics (DLs for short) are a family of knowledge representation formalisms that have been gaining growing interest in the last two decades, particularly after OWL (Ontology Web Language) [2], was approved as the W3C standard for representing the most expressive layer of the Semantic Web.

An ontology or knowledge base in $\mathcal{ALC}$ is a set of axioms a_i defined over the triple (N_C, N_R, N_O) [1], where N_C is the set of *concept names* or *atomic concepts* (unary predicate symbols), N_R is the set of *role or property names* (binary predicate symbols), and N_O the set of *individual names* (constants), instances of N_C and N_R.

N_C contains concepts (like Bird, Animal, etc) as well as other concept definitions as follows. If r *is* a *role* (r $\in N_R$) and C and D are *concepts* (C, D $\in N_C$) then the following definitions belong to the *set of $\mathcal{ALC}$ concepts*: (i) C $\sqcap$ D (the intersection of two concepts); (ii) C $\sqcup$ D (the union of two concepts); (iii) $\neg$C (the complement of a concept); (iv) $\forall$r.C (the universal restriction of a concept by a role); and (v) $\exists$r.C (the existential restriction of a concept by a role). Note that, in the definitions above, C and D can be inductively replaced by other complex concept expressions.

There are two axiom types allowed in $\mathcal{ALC}$: (i) *Assertional axioms*, which are concept assertions C(a), or role assertions R(a,b), where C $\in N_C$, r $\in N_R$, a,b $\in N_O$ and (ii) *Terminological axioms,* composed of any finite set of GCIs (*general concept inclusion*) in one of the forms C $\sqsubseteq$ D or C $\equiv$ D, the latter meaning C $\sqsubseteq$ D and D $\sqsubseteq$ C, C and D being concepts. An ontology or knowledge base (*KB*) is referred to as a pair ($\mathcal{T}$,$\mathcal{A}$), where $\mathcal{T}$ is the *terminological box* (or *Tbox*) which stores terminological axioms, and $\mathcal{A}$ is the *assertional box (ABox)* which stores assertional axioms. $\mathcal{T}$ may contain *cycles*, in case at least an axiom of the form C $\sqsubseteq$ D, D can be expanded to an expression that contains C.

There are two major ways of defining $\mathcal{ALC}$ semantics. The most frequently used one relies on the definitions of interpretation, model, fixpoints, etc, over a domain Δ [1]. Another way is mapping $\mathcal{ALC}$ constructs to first order logic (FOL) (like in [6]) and exploiting the semantics defined for FOL. FOL semantics is available, for instance in [3]. The latter was chosen, with the purpose of taking advantage of the already formalized completeness and soundness theorems for the work.

Next, the $\mathcal{ALC}$ disjunctive normal form used by the inference system is presented.

2.1. An $\mathcal{ALC}$ Positive Matrix Form

Definition 1 ($\mathcal{ALC}$ disjunctive normal form ($\mathcal{ALC}$ DNF), clause). An *$\mathcal{ALC}$ DNF formula* is a disjunction of conjunctions, in the form $C_1 \vee \ldots \vee C_n$, where each C_i is a *clause.* Clauses are conjunctions of $\mathcal{ALC}$ predicates ($\mathcal{ALC}$ concepts or relations, instantiated or not), like $L_1 \wedge \ldots \wedge L_m$, also denoted as $\{L_1, \ldots, L_m\}$. $\mathcal{ALC}$ formulae can be also expressed in *$\mathcal{ALC}$ disjunctive clausal form* or *$\mathcal{ALC}$ positive matrix form,* as $\{C_1, \ldots, C_n\}$,. A formula stated this way is called an *$\mathcal{ALC}$ matrix*. In an $\mathcal{ALC}$ matrix, each clause occupies a column.

To reach this normal form, the first two actions to be made over the axioms are: (i) splitting equivalence axioms of the form $\mathrm{C} \equiv \mathrm{D}$ into two axioms $\mathrm{C} \sqsubseteq \mathrm{D}$ and $\mathrm{D} \sqsubseteq \mathrm{C}$, and (ii) converting all the axioms into a *Negated Normal Form* (NNF), in which negations occurs only on literals [1]. Next, some necessary definitions are introduced.

Definition 2 ($\mathcal{ALC}$ disjunction, $\mathcal{ALC}$ conjunction). An $\mathcal{ALC}$ disjunction is either a literal, a disjunction $\mathrm{E}_0 \sqcup \mathrm{E}_1$ or an universal restriction $\forall \mathrm{r}.\mathrm{E}_0$. An $\mathcal{ALC}$ conjunction is either a literal, a conjunction $E_0 \sqcap E_1$ or an existential restriction $\exists \mathrm{r}.\mathrm{E}_0$. E_0 and E_1 are arbitrary concept expressions.

Definition 3 ($\mathcal{ALC}$ pure disjunction). The set S_D of $\mathcal{ALC}$ pure disjunctions is the smallest set where: (i) $\mathrm{D}_0 \in S_D$ for every literal D_0; (ii) $\mathrm{D}_0 \sqcup \mathrm{D}_1 \in S_D$, in case $\mathrm{D}_0, \mathrm{D}_1 \in S_D$; (iii) if $\mathrm{D}_0 \in S_D$ then $\forall \mathrm{r}.\mathrm{D}_0 \in S_D$. An element $\breve{\mathrm{D}} \in S_D$ is an $\mathcal{ALC}$ pure disjunction. An $\mathcal{ALC}$ *non-pure disjunction* is an $\mathcal{ALC}$ disjunction that is not pure.

Definition 4 ($\mathcal{ALC}$ *pure* conjunction). The set S_C of $\mathcal{ALC}$ pure conjunctions is the smallest set where: (i) $\mathrm{C}_0 \in S_C$ for every literal C_0; (ii) $\mathrm{C}_0 \sqcap \mathrm{C}_1\ S_C$ iff $\mathrm{C}_0, \mathrm{C}_1 \in S_C$; and (iii) if $\mathrm{C}_0 \in S_C$ then $\exists \mathrm{r}.\mathrm{C}_0 \in S_C$. An element $\hat{\mathrm{C}} \in S_C$ is an $\mathcal{ALC}$ pure conjunction. An $\mathcal{ALC}$ *non-pure conjunction* is an $\mathcal{ALC}$ conjunction that is not pure.

Definition 5 (Impurity on a non-pure expression). Impurities on non-pure $\mathcal{ALC}$ DL expressions are either conjunctive expressions in a non-pure disjunction or disjunctive expressions in a non-pure conjunction. The set of impurities is called $\mathcal{ALC}$ *impurity set*, and is denoted by S_I.

Example 1 (Impurities on non-pure expressions).
The expression $(\forall \mathrm{r}.(\mathrm{D}_0 \sqcup \ldots \sqcup \mathrm{D}_\mathrm{n} \sqcup (\mathrm{C}_0 \sqcap \ldots \sqcap \mathrm{C}_\mathrm{m}) \sqcup (\mathrm{A}_0 \sqcap \ldots \sqcap \mathrm{A}_\mathrm{p}))$, a non-pure disjunction, contains two impurities: $(\mathrm{C}_0 \sqcap \ldots \sqcap \mathrm{C}_\mathrm{m})$ and $(\mathrm{A}_0 \sqcap \ldots \sqcap \mathrm{A}_\mathrm{p})$.

Definition 6 (Positive normal form). An $\mathcal{ALC}$ axiom is in positive normal form iff it is in one of the following forms: (i) $\hat{\mathrm{C}} \sqsubseteq \breve{\mathrm{D}}$; (ii) $\mathrm{C} \sqsubseteq \exists \mathrm{r}.\hat{\mathrm{C}}$; and (iii) $\forall \mathrm{r}.\breve{\mathrm{D}} \sqsubseteq \mathrm{C}$; where C is a concept name, $\hat{\mathrm{C}}$ a pure conjunction and $\breve{\mathrm{D}}$ a pure disjunction.

In [7], algorithms for transforming $\mathcal{ALC}$ axioms to this normal form are available.

2.2 Translation Rules for the normalization

With all axioms in the normal form defined just above, it is easy to map them to matrix form, by applying the rules given in Table 1. Table 2 brings the mapping treatment of recursive sub-cases of existential and universal restrictions, when they occur inside any of the the normal form $\hat{\mathrm{C}} \sqsubseteq \breve{\mathrm{D}}$,. An improvement of the approach is, as the usual DL notation, variables are not needed, since all relations are binary and dash lines make for disambiguation, as soon will be seen.

Some important remarks must be stated at this point for the sake of clarity.

The first is that the whole knowledge base *KB* is negated during this transformation. This is due to the fact that in order to prove the validity of $KB \vDash \alpha$ (α being a subsumption axiom) using a direct method, $\neg KB \vee \alpha$ must be proven a tautology (coming from $KB \longrightarrow \alpha$, according to the Deduction Theorem [3]).

Table 1. Translation rules to map $\mathcal{ALC}$ into FOL positive NNF and matrices.

Axiom type	**FOL Positive NNF mapping**	**Matrix**
$\mathrm{C} \sqsubseteq \exists \mathrm{r}.\hat{\mathrm{C}}$, where $\hat{\mathrm{C}} = \bigwedge_{i=1}^{n} \mathrm{A_i}$ with $\mathrm{A_i} \in S_C$ (pure conjunction)	$(C(x) \wedge \neg r(x,f(x))) \vee$ $(C(x) \wedge \neg A_1(f(x))) \vee$ $\ldots \vee$ $(C(x) \wedge \neg A_n(f(x)))$	$\begin{bmatrix} C & C & \cdots & C \\ \neg r & \neg A_1 & \cdots & \neg A_n \end{bmatrix}$
$\forall \mathrm{r}.\breve{\mathrm{D}} \sqsubseteq \mathrm{C}$, where $\breve{\mathrm{D}} = \bigvee_{j=1}^{m} \mathrm{A'_j}$ with $A'_j \in S_D$ (pure disjunction)	$(\neg r(x,f(x)) \wedge \neg C(x)) \vee$ $(\neg A'_1((f(x)) \wedge \neg C(x)) \vee$ $\ldots \vee$ $(\neg A'_m((f(x)) \wedge \neg C(x))$	$\begin{bmatrix} \neg r & A'_1 & \cdots & A'_m \\ \neg C & \neg C & \cdots & \neg C \end{bmatrix}$
$\hat{\mathrm{C}} \sqsubseteq \breve{\mathrm{D}}$, where $\hat{\mathrm{C}} = \bigwedge_{i=1}^{n} \mathrm{A_i}$, $\breve{\mathrm{D}} = \bigvee_{j=1}^{m} \mathrm{A'_j}$ with $\mathrm{A_i} \in S_C$ (pure conjunction) and $\mathrm{A'_j} \in S_D$ (pure disjunction)	$A_1(x) \wedge \ldots \wedge A_n(x)$ $\wedge$ $\neg A'_1(x) \wedge \ldots \wedge \neg A'_m(x)$	$\begin{bmatrix} A_1 \\ \vdots \\ A_n \\ \neg A'_1 \\ \vdots \\ \neg A'_n \end{bmatrix}$

Table 2. Recursive sub-cases of restrictions for axioms of type $\hat{\mathrm{C}} \sqsubseteq \breve{\mathrm{D}}$,.

Axiom type	**FOL Positive NNF mapping**	**Matrix**
A is an existential restriction: … ⊓ ∃r.A ⊓ … , with A ∈ S_C (pure conjunction)	… ∧ $r(x,y) \wedge$ $A(y) \wedge$ …	$\begin{bmatrix} \vdots \\ r \\ A \\ \vdots \end{bmatrix}$
A' is an universal restriction: …⊔ ∀r.A' ⊔ …, with A' ∈ S_D (pure disjunction)	…∧ $r(x,y) \wedge$ $\neg A'(y) \wedge$ …	$\begin{bmatrix} \vdots \\ r \\ \neg A' \\ \vdots \end{bmatrix}$

The first consequence of that is that subsumption axioms of the form $\mathrm{C} \sqsubseteq \mathrm{D}$, which are usually logically translated as $\mathrm{C} \rightarrow \mathrm{D}$, because negated ($\neg(\mathrm{C} \rightarrow \mathrm{D})$, indeed), are now translated to $\mathrm{C} \wedge \neg\mathrm{D}$, instead of $\neg\mathrm{C} \vee \mathrm{D}$. Moreover, to establish a uniform set of rules applicable over $\mathcal{ALC}$ formulae, $\neg\alpha$ is dealt with, instead of α, so as to consider $\mathcal{ALC}$ axioms as $\neg A_1 \vee \ldots \vee \neg A_n \vee\ \alpha$ (coming from $\neg A_1 \vee \ldots \vee \neg A_n \vee \neg\neg\alpha$), where $A_i \in \mathcal{T}$ (axioms in the TBox). The translation rules are then applied over $\neg\alpha$ and all A_i.

Another remark regards *implicit skolemization***.** In the original CM, once the whole KB should be negated, universal quantifiers ($\forall$) are skolemized instead of existential, and all variables in the resulting DNF are then (implicitly) existentially quantified. Aiming at keeping a close correspondence with the original CM, only the cases in which $\mathcal{ALC}$ axioms, when translated to FOL, contain Skolem functions need to solved. For instance, the axiom $\mathrm{C} \sqsubseteq \exists \mathrm{r.B}$, negated, translates to FOL Skolem DNF as $\exists x(A(x) \wedge \neg r(x,f(x))) \vee (A(x) \wedge \neg B(f(x)))$ (coming from the skolemization of $\forall y$ in $\exists x\ (A(x) \wedge (\forall y\ (\neg r(x,y) \wedge \neg B(y))))$. In such cases, vertical and horizontal dash lines are employed to stand for the relations in the matrix. Vertical dash lines represent existential restrictions ($\exists$r.C), which are not skolemized, while horizontal dash lines represent universal restrictions ($\forall$r.C), in which the qualifier concept corresponds to a skolemized concept. An example should makes things clearer.

Example 2 (Positive normal form, skolemization, clause, matrix). The query

$$\left.\begin{array}{l} \mathrm{Animal} \sqcap \exists\mathrm{hasPart.Bone} \sqsubseteq \mathrm{Vertebrate} \\ \mathrm{Bird} \sqsubseteq \mathrm{Animal} \sqcap \exists\mathrm{hasPart.Bone} \sqcap \exists\mathrm{hasPart.Feather} \end{array}\right\} \vDash \mathrm{Bird} \sqsubseteq \mathrm{Vertebrate}$$

reads in FOL as $(\forall w(Animal(w) \wedge (\exists z\ (hasPart(w,z) \wedge Bone(z)) \rightarrow Vertebrate(w))) \wedge (\forall x(Bird(x) \rightarrow Animal(x) \wedge \exists y(hasPart(x,y) \wedge Bone(y)) \wedge \exists v(hasPart(x,v) \wedge Feather(v))) \rightarrow \forall t(Bird(t) \rightarrow Vertebrate(t)))$

where the variables y and t were respectively skolemized by the function $f(x)$ and the constant c. In positive matrix clausal form, the formula is represented by

$\{\{Bird(x), \neg Animal(x)\}, \{Bird(x), \neg hasPart(x,f(x))\}, \{Bird(x), \neg Bone(f(x))\}, \{Bird(x), \neg hasPart(x,g(x))\}, \{Bird(x), \neg Feather(g(x))\}, \{Animal(w), hasPart(w,z), Bone(z), \neg Vertebrate(w)\}, \{\neg Bird(c)\}, \{Vertebrate(c)\}\}$. Figure 1 shows it as a matrix.

$$\left[\begin{array}{cccccccc} Bird & Bird & Bird & Bird & Bird & Animal & \neg Bird(c) & Vertebrate(c) \\ \neg Animal & \underline{\neg hasPart} & \underline{\neg Bone} & \underline{\neg hasPart} & \underline{\neg Feather} & hasPart\ \vdots & & \\ & & & & & Bone\ \vdots & & \\ & & & & & \neg Vertebrate & & \end{array}\right]$$

Figure 1. An $\mathcal{ALC}$ formula and query in disjunctive clausal form represented as a matrix.

The vertical line connecting the symbols hasPart and Bone in the antepenultimate column has the function of denoting that the concept Bone has implicitly in FOL as argument the variable that is the 2nd argument of the relation hasPart (coming from the FOL translation $\exists x \exists y\ hasPart(x,y) \wedge Bone(y)$ that arose from the axiom $\exists$hasPart.Bone $\sqsubseteq$ Vertebrate. On its turn, the horizontal dash line connecting the symbols $\neg$hasPart and $\neg$Feather represents the $\mathcal{ALC}$ expression $\exists$hasPart.Feather in the original axiom Bird $\sqsubseteq$ $\exists$hasPart.Feather. This expression, negated, corresponds in FOL Skolem NF to $\exists x\ \neg hasPart(x,f(x)) \vee \neg Feather(f(x))$. Thus, during reasoning, skolemized predicates such as the last ones can only be unified with variables, and not with concrete individuals or other with distinct Skolem functions.

Dash lines can superpose other dash lines, for instance to represent the axiom $\mathrm{A} \sqsubseteq \exists \mathrm{r}_1.(\exists \mathrm{r}_2.\mathrm{B})$. In this normal form they cannot cross, although this would not consist in a problem for the $\mathcal{ALC}$ method.

A last remark regards memory usage. By using this normal form, the number of newly introduced symbols (and also the number of axioms) grows linearly with the number of impurities. It is in most cases a gain compared with other normalizations (e.g., $\mathcal{EL}$ [4] and resolution [6]), whose number of new axioms grows linearly to the axioms' length, although in the worst case they are equal. See [7] for more examples and further discussion regarding this topic.

3 An $\mathcal{ALC}$ Connection Method

The connection method (CM) [3] was created by W. Bibel in the 70s, and earned good reputation in the field of automated theorem proving in the 80s and 90s. Apart from needing less memory than other first-order logic inference systems, it offers the same advantages, according to its author, since any reductions, compressions, optimizations and improvements that fit to resolution or tableaux in can also be properly applied to it (see [3], 4.4). As a result, it is probably feasible to finding ways to implement typical DL tableaux optimizations to the $\mathcal{ALC}$ connection method under development, although such issues are out of the scope of the current paper.

Such distinctive features naturally led to the idea of proposing a connection method especially tailored to infer over description logic Semantic Web ontologies. The core of the paper, the $\mathcal{ALC}$ connection calculus, is explained next.

3.1 An $\mathcal{ALC}$ Connection Sequent and Matrix Calculus

Definition 7 (Path, connection, unifier, substitution). A *path* is a set of literals from a matrix in which every clause (or column) contributes with one literal. A *connection* is a pair of complementary literals from different clauses, like $\{L_1^\sigma, \neg L_2^\sigma\}$, where $\sigma(L_1)$ (or $\sigma(\overline{L_2})$) is the most general unifier (mgu) between predicates L_1 and $\neg L_2$. σ is the set of *substitutions,* which are mappings from variables to terms. All occurrences of the variable contained in a substitution would be replaced by the term indicated in σ.

Example 4 (Path, connection, unifier, substitution). Some paths of the matrix of Figure 1 are*: {Bird(x), ¬Bone(f(x)), Animal(w) ,¬Bird(c), Vertebrate(c)}* and *{¬Animal(w), Bird(x), ¬Bone(f(x)), Animal(w) ,¬Bird(c), Vertebrate(c)}*. Some connections from the matrix are *{Bird(x), ¬Bird(c)}*, *{¬hasPart(x,f(x)), hasPart(w,z)}*, and *σ={x/c, w/c, z/f(c)}* is the whole matrix' mgu.

Lemma 1 (Validity, active path, set of concepts). An $\mathcal{ALC}$ formula represented as a matrix M is *valid* when every path contains a connection $\{L_1, \neg L_2\}$, provided that $\sigma(L_1) = \sigma(\neg\overline{L_2})$. This is due to the fact that a connection represents the tautology $L_1^\sigma \vee \neg L_2^\sigma$ in DNF. As a result, the connection method aims at finding a connection in each path, together with a unifier for the whole matrix. During the proof, the current path is called *active path* and denoted by $\mathcal{B}$. The *set of concepts* τ of a variable or instance x during a proof is defined by $\tau(x) \overset{\text{def}}{=} \{C | C(x) \in \mathcal{B}\}$ [9].

Definition 8 ($\mathcal{ALC}$ connection calculus in sequent style). Figure 2 brings the $\mathcal{ALC}$ connection calculus rules in sequent style of the, adapted from [10]. The basic structure is the tuple *{C,M,Path}*, where the clause C is the open sub-goal, M is the matrix corresponding to the query $KB \vDash \alpha$, and *Path* is the active path.

$$Axiom\ (Ax)\ \frac{}{\{\,\}, M, Path}$$

$$Start\ Rule\ (St)\ \frac{C_2, M, \{\}}{\varepsilon, M, \varepsilon}$$
$$where\ M\ is\ the\ matrix\ KB \vDash \alpha,\ C_2\ is\ a\ copy\ of\ C_1 \in \alpha$$

$$Reduction\ Rule\ (Red)\ \frac{C^{\sigma}, M, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}$$
$$with\ \sigma(L_1) = \sigma(\overline{L_2})$$

$$Extension\ Rule\ (Ext)\ \frac{C_2^{\sigma} \backslash \{L_2^{\sigma}\}, M, Path \cup \{L_1\} \qquad C^{\sigma}, M, Path}{C \cup \{L_1\}, M, Path}$$
$$with\ \ C_2\ a\ copy\ of\ C_1 \in M, L_2 \in C_2, \sigma(L_1) = \sigma(\overline{L_2}),$$

$$Copy\ Rule\ (Cop)\ \frac{C \cup \{L_1\}, M \cup \{C_2^{\mu}\}, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}$$
$$with\ L_2 \in C_2, \mu \leftarrow \mu + 1, and$$
$$\left(x_{\mu}^{\sigma} \notin N_O\ or\ \tau(x_{\mu}^{\sigma}) \nsubseteq \tau\left(x_{\mu-1}^{\sigma}\right)\right)\ with\ \sigma(L_1) = \sigma(\overline{L_2})\ (blocking\ conditions)$$

Figure 2. The $\mathcal{ALC}$ connection sequent calculus rules (adapted from [10]).

Rules are applied bottom-up. To start, the matrix *M* representing $KB \vDash \alpha$ is put in the bottom of the *St* rule, and a clause $C_1 \in \alpha$ is chosen as the first clause. Then, the three last rules are applied. The key rule for the calculus is the *Extension rule*, once it finds connections of the form $\{C(x), \neg C(y)\}$, ($\{\mathrm{C}, \neg\mathrm{C}\}$ in the notation used here) or $\{r(x,y), \neg r(z,w)\}$and the proper unifier σ for it (In the first case $\sigma=\{x/y\}$ and in the second $\sigma=\{x/z, y/w\}$). Besides this slight change in the *Start rule* ($C_1 \in \alpha$), a blocking mechanism was included as a new rule, the *Copy rule*. If the current literal can only connect a clause already in the active path, this clause is copied to the matrix, and the indexing function $\mu: M \to \mathbb{N}$, that assigns the number of copies of each clause, is incremented. The use of this function avoids ordering of individuals, as done in the original CM. An important remark about the rule is that the copy is virtual, in the sense that only an index μ is created. CM requires only one copy of the matrix in memory. This is one of its main advantages. Its use of memory may be better than tableaux in cases the proof demands many derivations, like with cyclic ontologies.

Blocking didn't occur in the original CM due to FOL semi-decidability, but it consists in a common practice in DL to guarantee termination. The first action to be accomplished is incrementing μ. Then, before making a copy of clause C_2 (this copy will be named C_2^μ), it is necessary to check whether the set of concepts τ associated to the variable x_μ^σ (i.e., if the new x_μ was unified) of the new literal L_2^μ from C_2^μ being created by the *Cop* rule is not contained in the set of concepts of the original variable x ($\tau(x^\sigma)$), from $L_2(x)$ of C_2 [9].

Examples of the $\mathcal{ALC}$ CM calculus' application can be found at [7].

Theorem 1 (Soundness and completeness) An $\mathcal{ALC}$ formula in positive matrix form is valid iff there is a connection proof for "$\varepsilon, M, \varepsilon$", i.e. the application of the rules makes the *Axiom rule (Ax)* applicable to all leaves of the generated tree.

Proof The introduced changes in rules do not lose the soundness of the original CM (whose proof is available at [3], III.3.9 and 3.10), because such changes made the new rules only more restrictive, so as to making them fire in less cases than in the original CM. They also do not affect completeness, *viz*: (i) The original *Start rule* allows any clause to start the method. By restricting it to begin just with a literal from the consequent to be proven (α), proofs departing from possibly inconsistent matrices are avoided, from which any axiom can be summoned. In other words, it ensures the participation of α in the proof. If the axiom α holds, then a proof starting by one of its literals is found by the original CM as well. (ii) The blocking conditions hold only when: (a) a new clause is being created from a variable (thus, it is not an instance, as stated by the first blocking condition $x_\mu^\sigma \notin N_O$), and, (b) the variable being copied is already a copy of another variable attached to the same concepts (as stated by the conditions $\tau(x_\mu^\sigma) \nsubseteq \tau(x^\sigma)$). Therefore it prevents infinite applications of the *Copy rule* that would not help proving $KB \vDash \alpha$, and thus, such condition does not interfere in the original CM's completeness. ■

Theorem 2 (Termination). $\mathcal{ALC}$ CM always terminates.

Proof Having the mindset that the *Copy rule* is the only source of non-termination, the proof has three subcases: (i) *When KB contains no cycle,* the calculus finishes in a finite number of steps because the number of choices arising from the finite number of clauses and possible connections is also finite. Since the *Copy rule* is never fired, and there are no loops, $\mathcal{ALC}$ CM always terminates. (ii) *When KB contains cycles and* $KB \vDash \alpha$*,* the system always terminates, because $\mathcal{ALC}$ CM behaves exactly equal the original CM, whose termination proof can be found at [3]), except for the blocking conditions. However, blocking only plays the role of a termination condition, since it prevents the *Copy rule* to be fired, thus assuring termination; (iii) The case *when KB contains cycles and* $KB \nvDash \alpha$ is the only aspect not covered by the original CM, given FOL's semi-decidability. Nevertheless, $\mathcal{ALC}$ CM always terminates for this case, due to the fact that the *Copy rule* obliges the set of concepts of the newly created instance of literal L_2 not to be a subset of any of the sets of concepts of other introduced instances of the literal. Once every set of concepts is a subset of N_C and N_C is finite, the number of distinct sets is limited to $2^{|N_C|}$. So, the number of copies for literals is finite. Thus $\mathcal{ALC}$ CM always terminates in all three cases. ■

The system can also be expressed in an easier way using matrices to build proofs. Figure 5 portraits an example using matrices. An $\mathcal{ALC}$ CM calculus' algorithm of the system was adapted [7] from the one described in [3], III.7.2.

Example 5 ($\mathcal{ALC}$ connection calculus). Figure 3 deploys the proof of the query stated in example 1. In the figure, literals of the active path are in boxes and arcs denote connections. For building a proof, firstly a clause from the consequent (*Start rule*) is chosen, say, the clause $\{\neg\mathrm{Bird(c)}\}$ and a literal from it ($\neg\mathrm{Bird(c)}$).

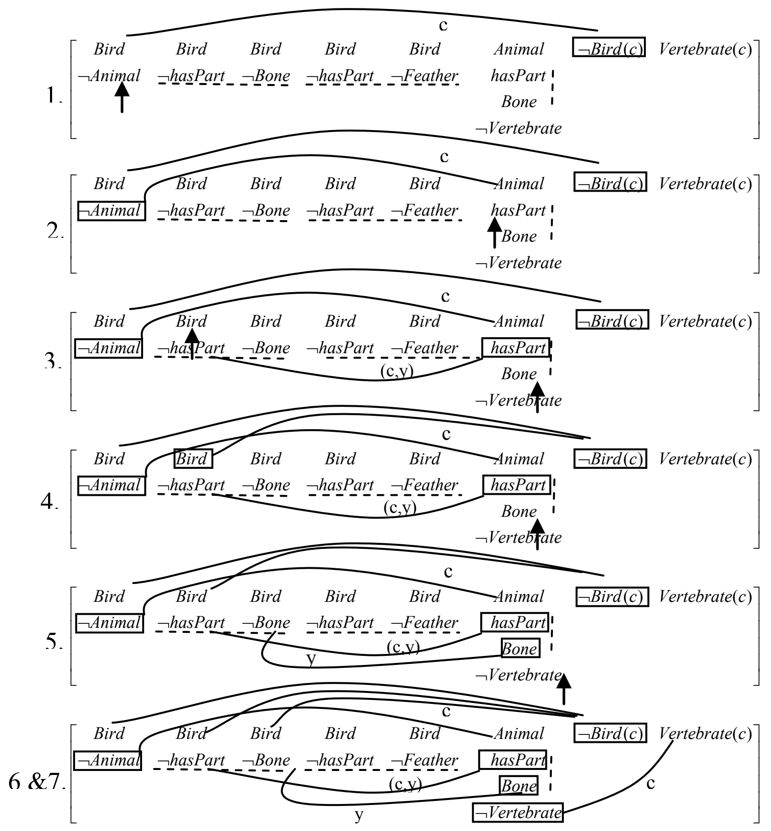


Figure 3. The connection proof example in matrix form.

Step 1 connects this clause with the first matrix clause. An instance or variable - representing a fictitious individual that is being predicating about -, appears in each arc; in this connection, the instance c. The arrow points to literals still to be checked in the clause (only the literal ¬Animal in Step 1), that should be checked afterwards. After step 2, the connection {¬Animal, Animal} is not enough to prove all paths stemming from the other clause (the one with literal ¬Animal*)*. In order to assure that, the remaining literals from that clause, *viz* hasPart, Bone and ¬Vertebrate, have still to be connected. Then, in step 3, when the predicate hasPart is connected, instance c being dealt with any more, but a relation between it and another variable or fictitious individual, say y (indicated by (c,y)).

Until that moment, only the *Extension rule* was applied. However, in step 4, the *Reduction rule* is used, triggered by its two enabling conditions: (i) there is a connection for the current literal already in the proof; and (ii) unification can take place. Unification would not be possible between different individuals and/or skolemized functions (in $\mathcal{ALC}$, equality among individuals is not necessary).

Next, a connection for the literal Bone was needed. That time, the variable (or fictitious individual) y is being referred, for the reason that it stands for a part of the individual c (hasPart(c,y)).

In case the system is able to summon the query, the processing finishes when all paths are exhausted and have their connections found. In case a proof cannot be entailed, the system would have tried all available options of connections, unifiers and clause copies, having backtracked to the available options in case of failure.

In [7], the interested reader could also find how the system deals with cycles and blocking, the DL services it is able to perform (such as a new way for inconsistency checking) and the reductions to subsumption and unsatisfiability.

4 Discussion

While planning the $\mathcal{ALC}$ CM, i.e. adapting the original CM to $\mathcal{ALC}$, the notation was designed in order to avoid any ambiguities to arise from the lack of variables in the representation – that was the rationale of dashlines replacing the quantifiers, so as to eliminate possible ambiguities with skolemization for the connections. This notation can be changed even further, since assertions are better stored outside the matrices, bringing two benefits: matrix size can be dramatically shrunk and powerful indexing mechanisms for retrieving assertions, such as relational databases, can be employed. However, even sticking only to TBoxes, matrices can be sparse, a memory waste. A simple solution for this consists in storing matrices as arrays of arrays.

Concerning reasoning, the first aspect to be taken into account resides in the strong similarity between CM and tableaux [10]. For that reason, reductions and optimizations designed over the latter are very likely to be incorporated to the former. Additionally, connection calculi are more goal-oriented than tableau calculi; therefore, they tend to be more efficient. As a support from this belief, this fact could be empirically observed in a comparison between a connection prover (leanCoP) and a tableaux connection prover (leanTAP) [8], although proving this requires more formal work and experiments Departing from such premises, in case the DL tableaux optimizations fit to $\mathcal{ALC}$ CM, a performance at least comparable to other leading DL tableaux approaches is expected to an $\mathcal{ALC}$ leanCoP , which is under development, given that CM itself usually displays a quite competitive performance (it already won a CASC competition once [11]). Of course, a deeper investigation on the cases in which could be advantageous to rely on tableaux or $\mathcal{ALC}$ CM is on the research agenda of this work.

5 Conclusions and Future Work

A connection method to take on the DL $\mathcal{ALC}$ was formalized in this work, by adapting the CM calculus formalized in sequent style from [10] and including a new rule. Notational improvements were also introduced, the key one being the representation without variables. Of course, this work is planned to continue in many research directions, such as implementations, other DLs, Semantic Web, etc.

First, the work presented here shall be extended to more complex description logic languages in a near future. Particularly, formalizations and implementations for the DLs $\mathcal{EL}$++, $\mathcal{SHIQ}$ and $\mathcal{SROIQ}$ will be practically useful for applications related to the Semantic Web provided that good solutions for dealing with equality are available, like eq-connections [3].

Another potential reasoning advantage to be pursued could be the memory required for $\mathcal{ALC}$ CM. Despite the fact that, given a connection proof, the size of an equivalent tableau proof is not significantly bigger (perhaps a quadratic increase in size); the proof search might indeed take up much more space, as the search is essentially saturation-based and many parts of a problem maybe not be required for the proof. However, these statements demand additional research to be proven true.

Last but not least, lean implementations written in Prolog, in the flavor of leanCop [8], that demand small memory space, can serve applications that are constrained in memory, such as stream reasoning in mobile applications, for instance.

References

1. Baader, F., Calvanese, D. McGuinness, D,, Nardi, D., Patel-Schneider, P. (Eds.): The Description Logic Handbook. Cambridge University Press, 2003.
2. Patel-Schneider, P., Hayes, P., Horrocks, I.: OWL - Web Ontology Language Semantics and Abstract Syntax W3C Recommendation. www.w3.org/tr/2004/rec-owl-semantics-20040210/ , 2004. [Accessed 10 Jan 2010].
3. Bibel, W. Automated theorem proving. Vieweg Verlag, Wiesbaden, 1987.
4. Baader, F., Brandt, S., Lutz, C. Pushing the EL Envelope. LTCS-Report 05-01, Chair for Automata Theory, Inst. for Theoretical Computer Science, TU Dresden, Germany, 2005.
5. Ghilardi, S., Lutz, C., Wolter, F. Did I damage my ontology: A Case of Conservative Extensions of Description Logics. Proceedings of the Tenth International Conference of Principles of Knowledge Representation and Reasoning 2006 (KR'06), AAAI Press, 2006
6. Schlicht, A. Stuckenschmidt, H. Peer-to-peer Reasoning for Interlinked Ontologies. Int. Journal of Semantic Computing, Special Issue on Web Scale Reasoning, 2010
7. Freitas, F., Schlicht, A., Stuckenschmidt, H. A Connection Method for Reasoning with the Description Logic $\mathcal{ALC}$. Technical report. University of Mannheim. 2010. www.cin.ufpe.br/~fred/CM-ALCTechRep.doc [Accessed 10 Dec 2010].
8. Otten, J., Bibel, W. leanCoP: Lean Connection-Based Theorem Proving. Journal of Symbolic Computation, Volume 36, pages 139-161. Elsevier Science, 2003.
9. Schmidt, R., Tishkovsky, D. Analysis of Blocking Mechanisms for Description Logics. In Proceedings of the Workshop on Automated Reasoning, 2007.
10. Otten, J. Restricting backtracking in connection calculi. AI Comm., 23(2-3): 159-182 2010.
11. Moser, M., Ibens, O., Letz, R., Steinbach, J., Goller, C., Schumann, J., Mayr, K. SETHEO and e-SETHEO - the CADE-13 systems. J. of Automated Reasoning, 18(2):237{246, 1997.

Modeling Structure in Description Logic

Henson Graves, Yvonne Bijan

Algos Associates,
2829 West Cantey Street,
Fort Worth, TX 76109 United States
Henson.graves@hotmail.com

Abstract. An overall goal of the INCOSE MBSE initiative is to provide SysML with a formal semantics and to integrate reasoning services as part of system engineering. UML class diagrams have been encoded as Knowledge Bases (KB) within the Description Logic (DL), ALCQI. The encoding provides a formal semantics for class diagrams which accords with the informal semantics. The encoding applies to SysML which is a profile of UML. The SysML block definition and internal block diagrams are not covered by the class diagram encoding. These diagrams are essential for representing composite structure such as manufactured products and molecular structures. The class diagram encoding is extended to composite structure diagrams in the DL $ALCQIb_{id}$. A composite structure diagram describes structures in terms of part decompositions and connections between objects. A SysML composite structure diagram can be encoded in the language of OWL2, but is not an OWL2 axiom set, as the diagrams contain property equations which violate the regularity ordering constraints for complex property inclusions. Conditions are given for an $ALCQIb_{id}$ KB which are sufficient to encode a SysML composite structure diagram. Further conditions are given for a KB, called a template, which ensure that within an interpretation all realizations of the composite structure have the same graph structure.

Keywords: Description Logic, Ontology, OWL, SysML, UML, structural modeling, molecular chemistry, human anatomy.

1 Introduction

Many engineering tasks involve reasoning on a description (a model in engineering terminology) to determine consistency and to derive implicit knowledge. An overall goal of the INCOSE MBSE initiative [4] is to provide the system engineering modeling language SysML [10], a dialect of UML [11] with a formal semantics and to integrate reasoning services as part of system engineering. SysML lacks a formal logic-based semantics, but has a well-developed informal semantics. For reasoning to give correct results, the formal semantics must be in accord with the informal semantics of SysML. To provide a formal semantics, one may axiomatize SysML directly [7] or encode SysML in a language which has a formal semantics such as OWL2 [12]. The OWL2 semantics is based on the Description Logic (DL)[1], SHOIQ [8].

UML class diagrams have been encoded as a Knowledge Base (KB) within the DL, ALCQI [2], a sublogic of SHOIQ. In this encoding, UML classes are encoded as concepts and UML associations are encoded as roles; to encode the additional information contained in class diagrams, other KB assertions are needed. The result is that an encoding of a UML class diagram is as a KB. The encoding provides a formal semantics for class diagrams which conforms to their informal semantics; the encoding is further validated by comparison of first order logic (FOL) axiomatizations of the UML constructions with the FOL representation of description logic. A consequence of the encoding for integration with reasoning is that a class diagram (class model) corresponds to a knowledge base within a DL. The results of DL consistency checking and derived classes and class inclusions can be reinterpreted within UML.

The DL encoding for UML and its results carry over to SysML. In SysML, blocks are a stereotype of class and SysML uses associations as does UML. SysML is well suited for representing descriptions of composite structure [5]. The SysML diagrams (models) used to represent composite structure are not fully covered by the UML encoding. A composite structure consists of objects and part objects linked by connectors. A structural description is a collection of classes and properties which describe a structure. A structure which satisfies the description is called a realization of the structure. A composite structure is represented in SysML with a Block Definition Diagram (BDD) and an Internal Block Diagram (IBD). A variety of specializations of associations are used to represent part properties and structural connections. Both part properties and connections are binary properties. A BDD describes a part decomposition structure. An IBD is a BDD with connection properties and property equations. An IBD can be used for representing structures which have, for example, multiple objects of the same class, which play different roles in the description. For example, an automobile description may specify four wheels with two front wheels which are driven by the engine and two rear wheels which are not driven. A SysML IBD model of the human heart accords well with the informal semantics and elucidates the distinction between the different kinds of properties (parts and connections) used to describe a heart.

Finding an appropriate Description Logic to represent the class of composite structure diagrams which is sufficiently expressive and for which reasoning is decidable and computationally tractable is challenging. A SysML IBD can be encoded in the language of SROIQ, but in the direct encoding it is not an OWL2 axiom set, as the connection property equations of an IBD violate the regularity ordering constraints on SROIQ axioms [8]. While it is possible to represent these diagrams within SROIQ with a description graph extension [9], there are questions regarding whether the description graphs correctly capture the intended semantics. For a DG extension of OWL2, the FOL suggested semantics for the human heart example in [9] is different from the DL semantics of a SysML IBD which appears to capture the informal semantics faithfully.

The role equations needed to represent the constructions in a SysML IBD are very restricted, even though they do not satisfy the regularity conditions of OWL2. The roles which represent part properties are all atomic with specified domain and range classes which are also atomic. The conditions satisfied by the part properties of a SysML IBD ensure that the part role paths [3] are finite and are unique. For each part role path, a new atom can be introduced to represent the path. Using these atomic path roles, simple role hierarchy assertions are sufficient to represent the equalities found in an IBD.

The UML class diagram encoding and its extension for conceptual modeling for data integration [3] is used to encode SysML Block Definition and Internal Block Diagrams using the description logic $\mathrm{ALCQIb}_{\mathrm{id}}$. A part structure for a KB is defined which encodes the essential features of a BDD. Conditions are given on a KB to encode the property equation features of an IBD. Additional meta conditions are given for an IBD KB, called a template, which ensure that within an interpretation, all realizations of the KB have the same graph structure. The conditions for a template are easily checked. A template is illustrated with a SysML model of the water molecule. For a template, results computed from the structure of the KB are valid in any realization. For example the weight of a structure can be computed from the IBD as all realizations will have the same number of parts.

2 The Description Logic $\mathrm{ALCQIb}_{\mathrm{id}}$

The specific description logic used to encode SysML Internal Block Diagrams is $\mathrm{ALCQIb}_{\mathrm{id}}$ [3]. $\mathrm{ALCQIb}_{\mathrm{id}}$ is an expressive DL that extends the basic DL language AL (attributive language) with negation of arbitrary concepts (indicated by the letter C), qualified number restrictions (indicated by the letter Q), inverse of roles (indicated by the letter I), boolean combinations of roles (indicated by the letter b), and identification assertions (indicated by the subscript id). Concepts and roles in $\mathrm{ALCQIb}_{\mathrm{id}}$ are formed according to the following syntactic rules [3]

$$C, C' \rightarrow A \mid \neg C \mid C \cap C' \mid C \cup C' \mid \forall R.C \mid \exists R.C \mid \geq n\ Q.C \mid \leq n\ Q.C \tag{1}$$

$$R, R' \rightarrow P \mid P- \mid R \cap R' \mid R \cup R' \mid R \setminus R' \tag{2}$$

where *A* denotes an *atomic concept*, *P* an *atomic role*, *P−* the *inverse* of an atomic role, *C* an arbitrary *concept*, and R, R' arbitrary roles. Furthermore, $\neg C$, $C \cap C'$, $C \cup C'$, $\forall R.C$, and $\exists R.C$ denote negation of concepts, concept intersection, concept union, value restriction, and qualified existential quantification on roles, respectively. We then use Q to denote *basic roles*, which are those roles that may occur in expressions of the form $\geq n\ Q.C$ and $\leq n\ Q.C$. A *basic role* can be an atomic role or its inverse, or a role obtained combining basic roles through set theoretic operators, i.e., intersection ("∩"), union ("∪"), and difference("\"). W.l.o.g., we assume difference applied only to atomic roles and their inverses.

Abbreviations are introduced for terms and assertions. *Thing* denotes the top concept which can be defined as $C \cup \neg C$ for a concept C, and *Nothing* the bottom concept. The concept *kQ.C* is an abbreviation of $\leq kQ.C \cap \geq kQ.C$. The *empty* denotes the role p\p for a role p. The notation *(funct p)* is used for the assertion that p is *functional*, i.e., as an abbreviation for $Thing \leq 1\ p.Thing$. The notation *p:(A,B)* is an abbreviation for the assertions

$$A \leq \forall p.B \text{ and } B \leq \forall p\text{-}.A \tag{3}$$

which capture the property that A is the domain and B is the range of p. For a functional role with $p{:}(A,B)$, we have $A \leq 1p.B$. For a role $p{:}(A,B)$ that is functional, we use the notation $p{:}(A,B)[1]$. A typed role p with $p{:}(A,A)$ is *irreflexive* if

$$Dom(p) \cap range(p) = Nothing \tag{4}$$

Two roles p and q are *disjoint*

$$p \perp q \text{ } IFF \text{ } p \cap q = empty \tag{5}$$

A *path* is given by the production rule

$$\pi 1,\ \pi 2 \rightarrow S \mid D? \mid \pi 1.\ \pi 2 \tag{6}$$

where S denotes an atomic role or the inverse of an atomic role, D denotes a concept, and $\pi 1.\pi 2$ denotes the composition of paths $\pi 1$ and $\pi 2$. The expression $D?$ is called a test role, it denotes the identity relation on instances of the concept D. If π is a path, the length of π, denoted length(π), is 0 if π has the form C?, is 1 if π has the form S, and is length(π1) + length(π2) if π has the form $\pi 1.\pi 2$.

An ALCQIb$_{id}$ knowledge base (KB) is a pair $\langle T, A \rangle$, where T is a TBox and A is an ABox. A TBox is a finite set of assertions of the form $C \leq C'$ with C and C' arbitrary concepts, or of the form $R \leq R'$ with arbitrary roles R and R', or an identification constraint. An *identification constraint* is an assertion of the form $(id\ C\ \pi 1, \ldots, \pi n)$ where C is a concept, $n \geq 1$, and $\pi 1, \ldots, \pi n$ are paths (called the components of the identifier) such that $length(\pi i) \geq 1$ for all $i \in \{1, \ldots, n\}$ and $length(\pi i) = 1$ for at least one $i \in \{1, \ldots, n\}$. Intuitively, an identification constraint asserts that for any two different instances o, o' of C, there is at least one π_i such that o and o' differ in the set of their πi-fillers. An ABox is a finite set of membership assertions of the form $A(a)$, $P(a, b)$, and $a \neq b$, with A and P respectively an atomic concept and an atomic role occurring in T, and a, b constants. The condition for role inclusions is weaker than the standard condition for a KB in ALCQIb$_{id}$.

The semantics of ALCQIb$_{id}$ concepts and roles is given in terms of interpretations, where an interpretation is defined as a correspondence of the KB concepts and roles [3] with classes and properties in a domain for which all of the KB assertions are satisfied. The semantics of a path π is defined in terms of the reverse of the composition of the roles occurring in the path. This device allows us to express well-formed path equations as role assertions within an ALCQIb$_{id}$ KB.

The semantics of an ALCQIb$_{id}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the set of models of $\mathcal{K}$, i.e., the set of interpretations satisfying all assertions in T and A. As noted in [3], checking whether an assertion holds in every model of a KB, is decidable in deterministic exponential time.

While ALCQIb$_{id}$ works for encoding SysML block diagrams it seems likely that some new variant could be devised which could be tailored more precisely for representing composite structure models.

3 Encoding SysML Block diagrams in a DL

We review the principles of encoding class diagrams established in [2] as applied to SysML block diagrams. We use a simple illustration of a water molecule model to show how role equations naturally occur in a composite structure diagram. For the water example we show that all realizations have the same structure. The next section will generalize the concept of a KB which abstracts the properties of a composite structure and show that a kind of KB called a template enjoyes the properties that all of its realizations are isomorphic.

The SysML language uses blocks which are classes and associations with several predefined kinds of specializations. The molecular unit of SysML is called a model. A SysML model is a collection of declarations which introduce constants of a signature and specify typing relations. A SysML model may contain subclasses and limited kinds of role assertions and may be composed and presented using multiple visual diagrams. However, all of the diagrams that constitute a model use the same block and property symbols.

A class diagram in UML is a restricted kind of SysML model. The encoding of UML class diagrams [2] carries over to SysML which is a UML profile developed for systems engineering. The Description Logic ALCQI is used to provide the encoding. This encoding accords with the informal semantics of UML. Classes (SysML Blocks) and associations are translated into DL concepts and roles [2]. The translation of a class diagram is as a role assertion. However, SysML models are not covered by the encoding in [2]. In particular, a SysML Block Definition Diagram and an Internal Block Diagram are not covered. An undirected association p is identified with a role p. The diagram of boxes labeled A and B connected by a line becomes the assertion *p:(A,B)*. An *aggregation* property p from A to B with cardinality restriction 1 is represented in DL this becomes $A \leq (\exists p)$. A property *p* with *p:(A,B)* is *mandatory* if $A \leq k\, p.\, B$. for an atomic functional role p with domain A and range B we use the abbreviation p:(A,B)[1].

This encoding of a SysML model as a KB is illustrated with a SysML water molecule model. The water molecule is represented as a SysML model using two kinds of diagrams, a Block Definition Diagram (BDD) to represent the decomposition structure and an Internal Block Diagram (IBD) to represent relationships among the parts within the structure. The language elements in both diagrams are part of the same SysML model. In Figure 1, the top half shows the decomposition structure for water. The BDD shows that water has three part properties whose range classes are oxygen and hydrogen. The shared Association (open diamond headed arrow) is a part property in SysML. There are two kinds of part properties in SysML. The shared Association property is used because the atoms can be a part of any molecule. The two arrows pointing at hydrogen mean that there are two parts of type hydrogen within water. The diamond arrow pointing to oxygen shows that there is one part of type oxygen within a water molecule. The numbers on the arrows in this diagram represent the cardinality restriction on the number of parts that a water molecule can have. In this case, the numbers are all 1, which says that an individual water molecule has exactly one oxygen and two hydrogen atoms as parts.

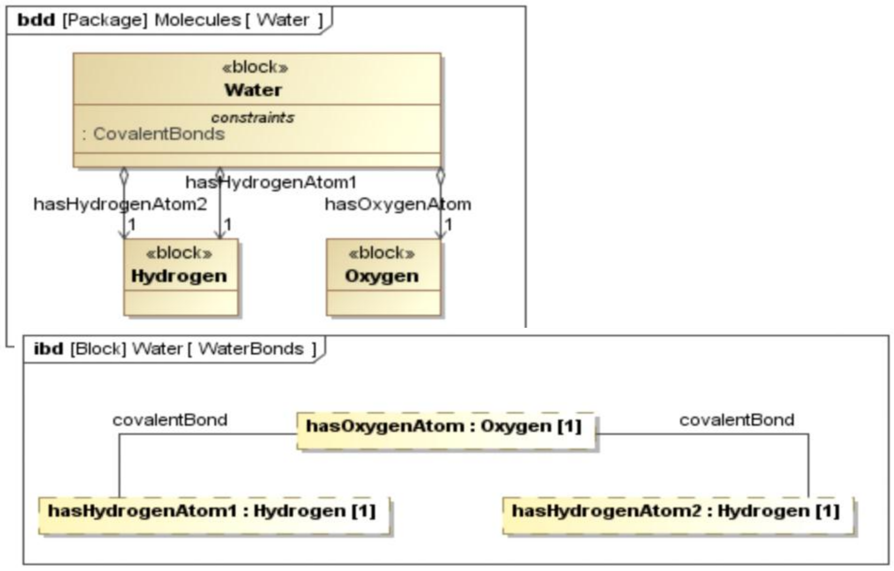

bdd [Package] Molecules [Water]
«block»
Water
constraints
: CovalentBonds
hasHydrogenAtom1
hasHydrogenAtom2
hasOxygenAtom
1
1
1
«block»
Hydrogen
«block»
Oxygen
ibd [Block] Water [WaterBonds]
covalentBond
hasOxygenAtom : Oxygen [1]
covalentBond
hasHydrogenAtom1 : Hydrogen [1]
hasHydrogenAtom2 : Hydrogen [1]

Fig. 1. SysML model for water molecule

The bottom diagram, called an Internal Block Diagram, shows the bonding relationships between the part properties. The arrows from the BDD are represented as rectangles with dashed lines. In this diagram, the rectangles are not blocks; they represent part properties of Water. The rectangle is labeled with the name of the part property and the range type of the property, as well as the cardinality restriction of the properties. The diagram shows the oxygen part has a covalent bond with each of the hydrogen parts. The diagram title box signifies that the IBD is within the scope of water.

The informal semantics of the water model is that any realization of a water molecule has exactly three atoms: two hydrogen atoms and one oxygen atom. Further, we expect that the covalent bonds from oxygen atom are connected to distinct hydrogen atoms. This fact will follow from the declaring the *covalentBond* property to be functional and declaring that the *hasHydrogenAtom1* and *hasHydrogenAtom2* are disjoint. Informally, a realization of a water molecule in this theory is a tree with a root *w1* corresponding to the water molecule and three other nodes, an oxygen atom *o1*, and two hydrogen atoms *h1, h2*. The tree has edges *{ <w1,o1>, <w1,h1>, <w1,h2>, <o1,h1>,<o1,h2> }*. The first three edges correspond to the part properties and the last two correspond to the bond properties.

For a KB used to encode a SysML composite diagram a *realization* of a KB is a collection of ABox assertions of the form objects $C_j(a_i)$ where the C_j are atomic and for any atomic concept *C* in the KB there is at least one ai with $C(a_i)$. Also, for any atomic role p in the KB there is a pair $<a_i,a_j>$ with $p(a_i,a_j)$. There may be multiple a_i with $C(a_i)$. A realization is an internal model of the KB. In general, a realization may not be finite. With the restrictions that will be used on a KB to encode an IBD, one can construct realizations of the KB by adding individuals. A template KB has finite realizations. Also, a model may contain multiple realizations.

In the first order logic representation of a DL we replace an existential assertion *p:(A,B)[1]* with a Skolem function and use the symbol *p* for the Skolem function as well as the role. We use the notation *a.p* for the value of the Skolem function *p*. This notation allows us to write *p(a, p(a))* as *p(a,a.p).* More generally for *p1* and *p2* functional atomic roles then from the composition semantics for roles one has *a.(p1.p2) = (a.p1).p2.*

The KB encoding the SysML water molecule model has as atomic roles, Water, Oxygen, and Hydrogen and as atomic roles, *hasOxygen*, *hasHydrogen1, hasHydrogen2, covalentBond1, covalentBond2.* Using the abbreviations the KB contains the assertions:

$$hasOxygen:(Water,\ Oxygen)[1] \tag{4}$$

$$hasHydrogen1:(Water,\ Hydrogen)[1] \tag{5}$$

$$hasHydrogen2:(Water,\ Hydrogen)[1] \tag{6}$$

$$covalentBond1:(Oxygen,\ Hydrogen)[1] \tag{7}$$

$$covalentBond2:(Oxygen,\ Hydrogen)[1] \tag{8}$$

the equational role equations

$$hasOxygen.covalentBond1\ =\ hasHydrogen1 \tag{9}$$

$$hasOxygen.covalentBond2\ =\ hasHydrogen2 \tag{10}$$

and the disjointness assertions

$$Oxygen \perp Hydrogen \tag{11}$$

$$hasHydrogen1 \perp hasHydrogen2 \tag{12}$$

$$covalentBond1 \perp covalentBond2 \tag{13}$$

It is easy to show that any realization of the water KB has the same structure. For any *ABox w with w.Water*, we iterate the constructions to obtain the set

{w, w.hasOxygen, w.hasHydrogen1, w.hasHydrogen,
w. hasOxygen.covalentBond1, w.hasOxygen.covalentBond2 }.

However,

$$w.\ hasHydrogenAtom1 \neq w.\ hasHydrogenAtom2 \tag{14}$$

by axiom (12). By axiom (9)

$$w.\ hasOxygen.covalentBond2 = w.\ hasOxygen.covalentBond2 \tag{15}$$

The structure only contains the root and part instances. We can verify that the role instance relations hold. This KB implies that any realization of Water has the expected component parts with the expected connections between them. The next task is to identify the properties of part roles which enable these arguments to be generalized.

4 Block Definition Diagrams and Internal Block Diagrams

The KBs which are used to encode SysML BDDs and IBDs are described below. An abstract Block Definition Diagram (ABDD) is a KB with a subset of the KB signature $p_i : i \in \{1,\dots n\}$ called part roles. Part roles satisfy the constraints that each p_i is declared with a domain and range type with a numeric multiplicity, the domain and range concepts of the part roles are in the KB signature and are atomic. There may be multiple part properties with the same domain and range types. For the following discussion, we restrict part properties to be functional. We use the abbreviation *p:Part(A,B)* for a part role *p* with *p(A,B).* The concepts that occur as the domain or range of a part role are called *Part* concepts. A part concept which is not the range concept of any part property is a root. We assume that the part class has a root and that it is unique.

(P0) *Root(A)*, for some atomic class *A* and the class is unique.

(P1) *If p:Part(A,B) then Irreflexive(p)*

(P2) *If p:Part(A,B) and p2:Part(B,C) then Irreflexive (p1.p2)*

(P3) *If p:Part(A,B) and p2:Part(C,B) then* $p1 \perp p2$

(P4) *PartClass(A) IFF Root(A)* or *p:Part(B,A)* and *PartClass(B)*

A *part path* is a well-formed composition of part properties *p1. p2 ….pn* were the *range(pi) = domain (pi+1)* for all i. (P0) identifies a concept as the root. (P1) and (P2) ensure that part paths do not contain multiple occurrences of a part properly and so have finite length. (P3) states that any two part roles with the same range are disjoint. The (P4) implies that all part concepts are connected the root by a part path. The meta-properties (P0) through (P4) are easily in a KB.

For an ABDD, the directed graph whose nodes are the root together with the expressions *p:A*, for a part property *p* with *A = Range(p)* and whose edges are the part roles is a tree. As there may be multiple parts with the same range, class labeling the class with the part role using the expressions of the form *p:A* makes the nodes distinct. Each part concept is reachable by a part path *p1…pn* where *pn = p* from the root by (P4). For any two property paths *p1…pn* and *q1…qk* which terminate at the node *p:A*, then the domain of *pn* and *qk* are equal. By (P3), the part roles *pn* and *qk* are disjoint and so the two paths cannot be equal. Thus, the part property path is unique. The ABBD is used to encode a SysML BDD. The conditions used to define an ABDD are enforced in a BDD.

An Abstract IBD (AIBD) is an ABDD together with a finite set of function role (connections) whose domain and range are part concepts, and a set of path equations of the form

$$p1 \ldots pn = q1 \ldots qk.c \tag{16}$$

where the *pi* and *qj* are part paths and *c* is in *{c1,..,ck}.* The connection roles encode the arrows between the boxes of an IBD. Conversely an AIBD can be displayed as a graph. A new atomic role p.c is introduced for any part path followed by a connection role. Note that p1…pn:(A,B) where domain(p1) = A and range(pn) = B. We extend this notation to p1….pn.c for a connection role c.

A path π of atomic roles $p_1,\ldots p_n$ is *well-formed* if range(p_i) = domain(p_{i+1}) for all $i \in \{1, \ldots, n\}$. For each well-formed path, π, of atomic roles, we introduce a new role atom π. For any path of length 1, the new role is identified with the atomic role that formed the path. As there is a finite number of part paths, we define a new atomic role for each part path *p1….pn*. Recall that when the atoms in a path are functional, we write *a.p* for the unique individual *b* with *p(a,b).* This notation simplifies the application of a path *p* applied to *a*. We also use the notation *p:Path(A,B)* for a path with domain *A* and range *B*.

While the DL ALCQIb$_{id}$ does not permit composition directly in the role inclusion assertions, we simulate composition with the atomic path roles. For any connection equation p1…pn = q1…qk.c in the IBD, we replace it with role inclusion assertion $q1 \ldots qk.c \leq p1 \ldots pn$. However, for any a,b , q.c(a,b) implies *p(a,b)*. However, if *a.A* then as *p* is functional, *a.p = a.c.b*. Thus, *q.c = p*. So the inequalities in an AIBD are actually equalities. A *template* is a AIBD where

(P5) *p1:Part(A,B)* and *p2:Part(A,C)* then $p \perp q$ or p = q

The template axiom says that no parts can be reused in a part decomposition. This statement can be made precise in the first order logic theory of the KB.

To prove properties about the models of an AIBD KB we use the full first order representation of the DL. In the theory generated by the KB existential assertions of the form p:(A,B)[1] are replaced by a first order Skolem function. Properties that hold in this theory will hold in any model of the KB. Note that the part path roles become functions. Thus, the notation *a.p1..pn* is meaningful and we have the associativity law a.(p.q) = (a.p).q.

Definition. For a template KB with root *A*, *a:A, and t:B* for a part class *B,* let

$$Partof(a,t) \text{ IFF } t = a \text{ or } a.p1 \ldots .pn \tag{17}$$

for a part path.

Lemma. For a template with root *A*, and *a:A:* If *t:B* for a part class *B* and *t* is a part of a, then the part decomposition is unique.

If *t* is a part of *a,* then *t* has a decomposition of the form *t = a.p1….pk* for some *p1,…,pk*. If *t = a.p* and *t = a.q*, for two part properties, then by the template property *p* disjoint q or they are equal, and so *a.p = a.q*. The argument is repeated for the successive individuals *(a.p1).p2…pn*. With the first order logic of the KB extended with an abstraction construction which allows terms of the form

$$\{ t : P(t)\} \qquad \textbf{(18)}$$

constructed from a predicate where the predicate is restricted to equalities with Boolean connectives, then we can define the notion of a realization of a template within the extended theory of the axiom set. The axioms for the abstraction construction include

$$P(a) \text{ IFF } a{:}\{ b : P(b)\} \qquad (19)$$

with usual rules for variables. Using the extended logic, a realization of a template is an abstraction type *G = {t : Partof(a,t) }* for *a:Root*. From the axioms for the part property declarations, a realization of a root instance has a unique part decomposition. A tree structure can be defined with the individuals in *G* as the nodes and *<t,t.p>* for a part property. Connection edges can be added similarly. A graph isomorphism can be inductively defined between any two realizations.

Theorem. For a template, the instances of the type *G = {t : Partof(a,t)}* for *a:Root* are the nodes of a tree with root *a*. The edges *<t, t.p>* where *range(t) = domain(p)*. The correspondence defined by mapping *a* to the root and a node of the form *t.p* to *p:Range(p)* corresponds the nodes of the parts structure with the nodes of the BDD together with the mapping of an edge *<t,t.p>* to the edge *p* in the BDD defines an isomorphism of the parts structure with the BDD. Any parts tree has the same number of parts.

The axioms given do not prohibit a structure from sharing individuals with another structure. This property can be added with the axiom:

(P6)   *p:Part(A,B)* implies *p.p*=id(A)*

An interpretation of an axiomatic SysML theory is a mapping of the individuals, pairs, classes, and properties, and other types which preserves the sort structure, the logical axioms, and the declarations. In particular, classes are mapped to subclasses of the mapping of Thing and individuals are instances of Thing. In any valid interpretation of the theory of a model, the unique decomposition will hold.

5 Conclusion

The encoding of a UML class diagram as an ALCQI KB gives an encoding of Class diagrams into OWL2. This encoding is extended to an encoding of a SysML Internal Block Diagram as a KB within the OWL2 language, but not as an OWL2 KB. Each atomic property in the Block Diagram is an atomic role and the well-formed property paths are finite. The encoding correctly captures the part decomposition which ensures that the models are tree like. SysML model development tools enforce the axioms that define a part structure. Conversely, the correspondence between the block diagrams and the DL language constructions provides a graphical syntax for DL. SysML dos not have individuals, i.e., ABoxes. The Encoding makes clear how individuals can be added to SysML. With the encoding all derivations of inconsistency and concept inclusions can be exported back into SysML.

It is easily to check whether the additional axioms for a template are present. These axioms correspond to manufacturing assumptions that ensure implementations of a design have the same parts and connection structure. For example, water is a subclass of molecules which have an oxygen part. However, Oxygen is not a subclass of the things bonded to hydrogen, only the oxygen molecules which are parts of water have this property. The use of a template KB enables the development of SysML models for which all realizations are isomorphic. This is very useful as computations of the model hold for all of its realizations. It seems likely that graph defined for an abstract IBD, is a Description Graph in the sense of [9].

The axioms given for the water model provide only structural information and are incomplete in terms of constraints on the bonds needed to determine a 3D visualization of a water molecule and do not address the dynamic behavior of water such as how it changes when it freezes. Much more complete axiomatic models of water can be given which address these properties. These SysML models require further extensions to DL to be addressed.

References

1. Baader, F., D Calvanese, DL McGuinness, D Nardi.: The description logic handbook, Cambridge University Press (2007)
2. Berardi, D., Calvanese, D., and De Giacomoa, G., Reasoning on UML class diagrams, Artificial Intelligence Volume 168, Issues 1-2, (2005)
3. Calvanese, D., De Giacomo, G., Lembo, D., Conceptual modeling for data integration, (2009)
4. Friedenthal, S., Greigo, R., and Sampson, M., INCOSE MBSE Roadmap, in "INCOSE Model Based Systems Engineering (MBSE) Workshop Outbrief", INCOSE International Workshop, Albuquerque, NM, (2008)
5. Graves, H.: Representing Product Designs Using a Description Graph Extension to OWL 2. OWLED (2008)
6. Graves, H.: Integrating SysML and OWL, OWLED (2009).
7. Graves, H.: Ontological Foundations for SysML, IC – MBSE 3rd International Conference on Model-Based Systems (2011)
8. Horrocks, I., Kutz, O., and Sattler, U.: "The Even More Irresistible SROIQ," in Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, pp. 57-67, American Association of Artificial Intelligence Press, (2006)
9. Motik, B., Cuenca Grau B., Sattler, U.: Structured objects in OWL: Representation and reasoning, Proceeding of the 17th international conference on World Wide Web (2008)
10. OMG Systems Modeling Language (OMG SysML™), V1.2 (2010)
11. OMG Systems Modeling Language (OMG UML) V.2 (2010)
12. OWL 2 Web Ontology Language W3C, September (2009)

Integrity Constraints for Linked Data

Alan Jeffrey and Peter F. Patel–Schneider

Alcatel–Lucent Bell Labs

Abstract. Linked Data makes one central addition to the Semantic Web principles: all entity URIs should be dereferenceable to provide an authoritative RDF representation. URIs in a linked dataset can be partitioned into the exported URIs for which the dataset is authoritative versus the imported URIs the dataset is linking against. This partitioning has an impact on integrity constraints, as a Closed World Assumption applies to the exported URIs, while a Open World Assumption applies to the imported URIs. We provide a definition of integrity constraint satisfaction in the presence of partitioning, and show that it leads to a formal interpretation of dependency graphs which describe the hyperlinking relations between datasets. We prove that datasets with integrity constraints form a symmetric monoidal category, from which the soundness of acyclic dependency graphs follows.

1 Introduction

Motivation. In the Semantic Web, entities are named by URIs, and are described by RDF documents. Linked Data [5] adds the constraint that entity URIs should be *dereferenceable* (HTTP URIs which accept `GET` requests), and dereferencing an entity URI returns an RDF *representation* of that entity. The W3C web architecture [8] calls such representations *authoritative*.

The RDF triples contained in an entity representation will generally refer to entities for which the representation is not authoritative. Such hyperlinks between datasets are often visualized as a *dependency graph*, such as the popular Linking Open Data cloud diagram [6] shown in Figure 1.

Linked Data puts a new spin on the open world stance of the Semantic Web: from the point of view of a given URI owner, the world is partitioned into local entities, for which the owner is authoritative, and imported entities, for which the owner is not authoritative. In this paper, we provide a formal model of this partitioning which includes:

– a partitioning of entities into *imported* and *exported* nodes, in addition to the familiar *blank* nodes,
– a definition of what it means for a dataset to satisfy its integrity constraints, based on the minimal models of Motik *et al.* [13], but adapted to partitioning,
– a model of acyclic dependency graphs, which can be built compositionally, and where integrity constraint satisfaction can be performed locally, and
– a proof that graphical reasoning for datasets is sound, by showing that datasets form a symmetric monoidal category.

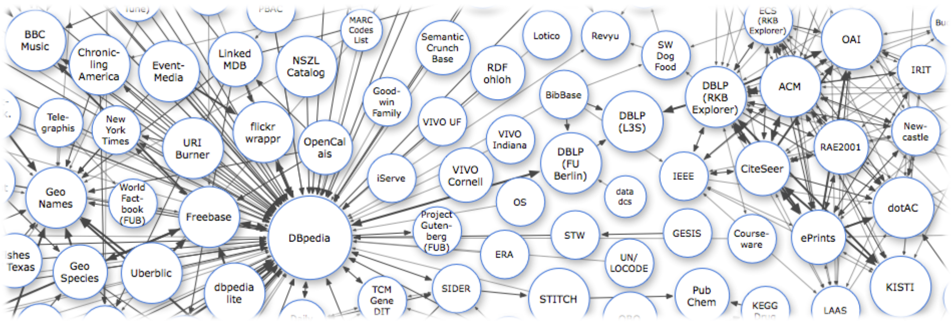


Fig. 1. Linking Open Data cloud diagram (detail)

This paper gives the first formal treatment of authoritative resource, integrity constraints for linked data, dependency graphs, and the categorical structure of semantic data. In this introduction, we provide informal examples to motivate our model, which are made precise in Sections 4 and 5.

Authoritative representations, imports and exports. The W3C web architecture [8] recommends as good practice that a URI owner "should provide authoritative representations of the resource it identifies". Typically, these are HTTP URIs, which respond to `GET` requests. Linked Data [5] applies this practice to the Semantic Web: URI owners provide authoritative representations of their URIs in RDF (for datasets) or OWL (for ontologies).

Semantic reasoners can make deductions from Linked Data. For example, consider a URI `bob:` (in examples, we will use URI prefixes such as `alice:` and `bob:`) which dereferences to the Turtle [4] representation:

```
bob: foaf:primaryTopic bob:me .
bob:me foaf:knows [ foaf:homepage alice: ] .
```

Now, if `alice:` dereferences to:

```
alice: foaf:primaryTopic alice:me .
```

then a reasoner can deduce (using the FOAF [2] specification's definitions):

```
bob:me foaf:knows alice:me .
```

In Linked Data, the entities in a dataset can be partitioned into:

– *exported nodes* (*enodes*): local entities, which the representation is authoritative for, with a publicly defined name that other datasets may link against,
– *blank nodes* (*bnodes*): local entities, which the representation is authoritative for, but without a publicly defined name, and
– *imported nodes* (*inodes*): all other entities.

For example, the RDF representation of `bob:` given above contains inode `alice:`, enodes `bob:` and `bob:me`, and an anonymous bnode (called `_:anon` below).

Ontologies and integrity constraints. A consumer of Linked Data may wish to assume a notion of correctness of the data it is consuming. Rather than considering integrity constraints to be given in a separate formalism such as a rules engine [14] or epistemic logic [15], we will use ontologies to express both deductive reasoning (the *standard* ontology), and the correctness criteria (the *constraint* ontology). A similar approach was taken by Motik *et al.* [13] and Tao *et al.* [17]. For example, consider the standard ontology:

$$\begin{aligned} \mathsf{homepage} &\doteq \mathsf{primaryTopic}^{-} \\ \mathsf{PersonalHomePage} &\sqsubseteq \mathsf{Document} \\ \mathsf{Document} &\sqsubseteq {\leq}1\,\mathsf{primaryTopic} \end{aligned}$$

and the constraint ontology:

$$\begin{aligned} \mathsf{PersonalHomePage} &\sqsubseteq \exists \mathsf{primaryTopic}\,.\,\mathsf{Person} \\ \mathsf{Person} &\sqsubseteq \forall \mathsf{knows}\,.\,\mathsf{Person} \end{aligned}$$

The above example is correct with respect to the *exported interface*:

$$\mathsf{Person}(\texttt{bob:me}) \quad \mathsf{PersonalHomePage}(\texttt{bob:})$$

under the assumption of the *imported interface*:

$$\mathsf{PersonalHomePage}(\texttt{alice:})$$

We reason informally as follows (this will be made formal in later sections).

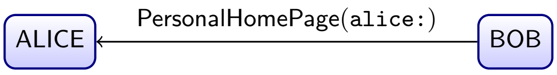
- The constraint $\mathsf{PersonalHomePage} \sqsubseteq \exists \mathsf{primaryTopic}\,.\,\mathsf{Person}$ is satisfied because the only new PersonalHomePage entity is `bob:`, and we have a witness `bob:me` for the role primaryTopic.
- The constraint $\mathsf{Person} \sqsubseteq \forall \mathsf{knows}\,.\,\mathsf{Person}$ is satisfied because the only new Person entity is `bob:me`, and the only entity which `bob:me` knows is `_:anon`. Now, in any world where PersonalHomePage(`alice:`), there must be some individual i such that primaryTopic(`alice:`, i) and Person(i). We can then reason using the standard ontology that $i =$ `_:anon` and so Person(`_:anon`).

This example, shows the use of two different styles of reasoning.

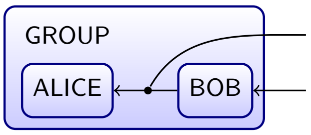
- When reasoning about exported or blank nodes, we can assume that the only properties are ones which can be deduced from information we have asserted, using the standard ontology. For example, this form of reasoning is used in "because the only new PersonalHomePage entity is `bob:`" and "the only entity in a knows role with `bob:me` is `_:anon`."
- We reason differently about imported nodes. All we know about the imported world is that it satisfies the imported interface, the standard ontology and the constraint ontology. For example, this form of reasoning is used in "in any world where PersonalHomePage(`alice:`), there must be some individual i such that primaryTopic(`alice:`, i) and Person(i)."

More succinctly, we use a Closed World Assumption for blank and exported nodes, and an Open World Assumption for imported nodes.

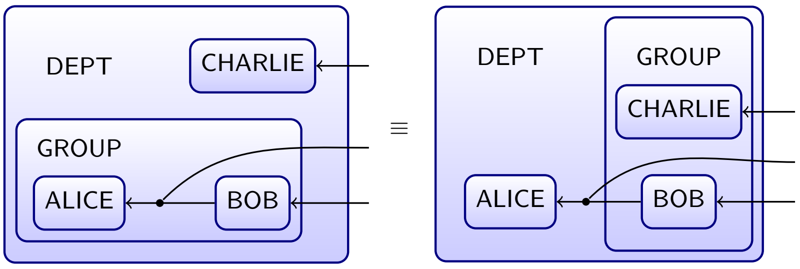
Dependency graphs. Dependency graphs such as Figure 1 are a common way of visualizing linked data, but have, until now, remained informal. We propose a formalization of such graphs as directed graphs where nodes are datasets such as ALICE (the authoritative representation of `alice:`) and BOB (the authoritative representation of `bob:`), and edges indicate the existence of hyperlinks between datasets. These edges are labeled by interfaces to make the contract between datasets explicit, for example:



Dependency graphs can be regarded as datasets, given by taking the union of all their constituent datasets (with a bit of bookkeeping to rename nodes to ensure no name clashes). Since dependency graphs form datasets, they can be nested, for example a GROUP which includes ALICE and BOB might be built:



Ensuring correctness should be compositional, for example knowing that ALICE and BOB are correct should ensure correctness of GROUP. Moreover, nested graphs should respect equivalence of datasets: if ALICE is replaced by an equivalent ALICE′, then GROUP should be equivalent to GROUP′. Finally, isomorphic graphs should be equivalent, irrespective of how they are composed, for example:



Symmetric monoidal categories. Our goals for dependency graphs are:

– Nodes describe datasets, edges describe hyperlink relationships.
– Graphs can be built compositionally, with local checking of correctness.
– Graph construction respects equivalence of datasets.
– Isomorphism of dependency graphs implies equivalence of datasets.

Proving these properties directly would be difficult, but fortunately there is an existing structure which guarantees these properties: a *symmetric monoidal category*. Category theory forms a foundational framework for mathematics, but our need of it is quite pragmatic: the equational theory of a symmetric monoidal category is precisely that of direct acyclic graphs (shown by Joyal and Street [10], see, for example, Selinger [16]). Figure 2 sketches how directed acyclic graphs form a symmetric monoidal category:

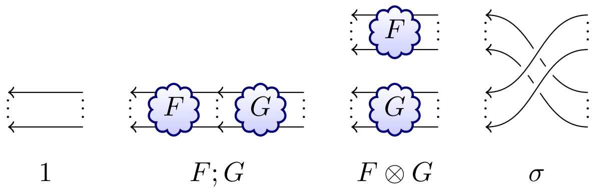


Fig. 2. Directed acyclic graphs form a (strict) symmetric monoidal category.

– The identity graph 1 just connects its source and target edges.
– The composition $F; G$ of graphs takes the disjoint union of F and G, and unifies the target edges of F with the source edges of G.
– The tensor $F \otimes G$ of graphs takes the disjoint union of F and G.
– The symmetry graph σ just permutes its source and target edges.

Since the equational theory of a symmetric monoidal category is precisely that of directed acyclic graphs, we can replace our goals for dependency graphs by the goal of showing that datasets form a symmetric monoidal category. This is a matter of proving a handful of equations, which is a easier than proving directly that graph isomorphism implies dataset equivalence.

Summary. The remainder of this paper will make this motivational section precise. We will define a notion of integrity constraint suitable for partitioning, and show that datasets with integrity constraints form a symmetric monoidal category, and hence can be formalized by dependency graphs. This is the first such investigation of integrity constraints for Linked Data. All results presented in this paper have been mechanically verified, using the Agda [1] mechanical proof assistant; all proofs are publicly available [9].

2 Preliminaries

In this paper, we consider a Description Logic $\mathcal{SHIN}^+_1$, which includes role hierarchies, role inverses, disjoint, reflexive, irreflexive and transitive roles, and singleton cardinality restrictions. We expect the results to apply to other description logics. Spelling this out, *roles* and *concepts* are defined by the grammars (where r and c are drawn from sets of atomic role names and concept names):

$$R ::= r \mid r^-$$
$$C ::= c \mid \neg c \mid \bot \mid \top \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall R \,.\, C \mid \exists R \,.\, C \mid {\leq}1\, R \mid {>}1\, R$$

A *TBox* is a finite set of *axioms* of the form:

$$C_1 \sqsubseteq C_2 \quad \text{or} \quad R_1 \sqsubseteq R_2 \quad \text{or} \quad \mathsf{Dis}(R_1, R_2) \quad \text{or} \quad \mathsf{Ref}(R) \quad \text{or} \quad \mathsf{Irr}(R) \quad \text{or} \quad \mathsf{Tra}(R)$$

Following Motik *et al.* [13], we assume ambient TBoxes S (the *standard* TBox) and T (the *constraint* TBox). For any finite set X, an *ABox* over X is a finite set of *assertions* of the form:

$$c(x) \quad \text{or} \quad r(x, y) \quad \text{or} \quad x \approx y \quad \text{where } x, y \in X$$

Note that ABoxes are restricted to contain only positive statements, and so have a monotone semantics. In many cases, this does not impact expressivity, as S can give names for arbitrary concepts, and T can introduce an irreflexive role $\mathsf{differentFrom}$ used in place of $\not\approx$ assertions. In practice, RDF is limited to positive atomic statements.

An *interpretation* $\mathcal{I}$ over X consists of a set $\Delta^{\mathcal{I}}$, together with $c^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each concept name c, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role name r, and $x^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $x \in X$. The *satisfaction* relations $\mathcal{I} \vDash A$ (for an ABox A over X) and $\mathcal{I} \vDash T$ (for a TBox T) are standard. Note that if $\mathcal{I}$ is an interpretation over $X \supseteq Y$, then $\mathcal{I}$ can be regarded as an interpretation over Y.

In the following, we will write $X \uplus Y$ for the disjoint union of X and Y: for simplicity, we will assume that X and Y are disjoint, and so $X \subseteq X \uplus Y \supseteq Y$. In the mechanized proofs [9], we use explicit tagging to ensure disjointness.

3 Initial interpretations

Consider ABoxes A over X, B over Y, and F over $(X \uplus V \uplus Y)$. We can think of A as the imported interface (where X is the set of inodes), B as the exported interface (where Y is the set of enodes) and F as the dataset (where V is the set of bnodes). Now, what does it mean for F to import A and export B, in the presence of ambient TBoxes S and T?

F can be thought of as a recipe for adding new assertions to an existing interpretation. Given any interpretation $\mathcal{I}$ over X which satisfies (S, T, A), we require there to be a canonical interpretation $\mathcal{J}$ over $(X \uplus V \uplus Y)$ which extends $\mathcal{I}$ with (S, F), and we require $\mathcal{J}$ to satisfy (T, B).

Motik *et al.* [13] use a similar notion of constraint satisfaction, although they consider all minimal $\mathcal{J}$, rather than a canonical $\mathcal{J}$, with respect to subset order on Herbrand models of Skolemized formulae. As they note, Skolemization has an impact on the notion of equivalence of TBoxes, for example $(c \sqsubseteq \exists r \,.\, d)$ is not equivalent to $(c \sqsubseteq \exists r \,.\, d, c \sqsubseteq \exists r \,.\, d)$ because they Skolemize differently (each existential quantifier introduces a new Skolem function, which may be interpreted differently). We avoid Skolemization by considering *initial* interpretations (relative to homomorphisms between interpretations) rather than *minimal* interpretations (relative to subset order).

Tao *et al.* [17] also consider minimal models, with respect to a partial order $\prec_=$ which preserves concept membership, role membership and equality of named individuals. They avoid Skolemization by an alternate semantics, where quantification only ranges over named individuals.

A *homomorphism* between interpretations $\mathcal{I}$ and $\mathcal{J}$ over X is a function $h : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that, for all x, i and j:

$$h(x^{\mathcal{I}}) = x^{\mathcal{J}} \qquad (i \in c^{\mathcal{I}}) \Rightarrow (h(i) \in c^{\mathcal{J}}) \qquad ((i,j) \in r^{\mathcal{I}}) \Rightarrow ((h(i), h(j)) \in r^{\mathcal{J}})$$

We will write $I \lesssim \mathcal{J}$ whenever there is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$. Consider an interpretation $\mathcal{I}$, and a family of interpretations $\mathcal{J}_i$ with a chosen family of homomorphisms $h_i : \mathcal{I} \to \mathcal{J}_i$. An *initial* $\mathcal{J}_i$ is one with a unique family of homomorphisms: $g_j : \mathcal{J}_i \to \mathcal{J}_j$ such that $g_j \circ h_i = h_j$. Note that initial interpretations do not always exist, but that when they do they are unique up to isomorphism.

Definition 1. *For any interpretation $\mathcal{I}$ over X and ABox F over $Z \supseteq X$, let $\mathcal{I} \oplus (S, F)$ be the initial interpretation $\mathcal{J}$ over Z such that $\mathcal{I} \lesssim \mathcal{J}$ and $\mathcal{J} \vDash S, F$.*

Note that $\mathcal{I} \oplus (S, F)$ does not always exist, as S may contain existentials or disjunctions which do not have canonical witnesses. For example there is no initial extension of $\emptyset$ by:

$$\mathsf{Bool} \sqsubseteq \mathsf{True} \sqcup \mathsf{False} \qquad \mathsf{True} \sqcup \mathsf{False} \sqsubseteq \mathsf{Bool} \qquad \mathsf{Bool}(x)$$

since there are two incomparable extensions, one with $\mathsf{True}(x)$ and one with $\mathsf{False}(x)$. However, there is a syntactic restriction which guarantees the existence of initial interpretations. Let S be *minimizable* whenever any axiom $C \sqsubseteq D$ has C built from atoms, $\bot$, $\top$, $\sqcup$, $\sqcap$ and $\exists$, and D built from atoms, $\top$, $\sqcap$, $\forall$ and $\leq$.

Proposition 1. *If S is minimizable, then $\mathcal{I} \oplus (S, F)$ exists.*

4 Integrity constraints

Having defined initiality, we can now define constraint satisfaction. This is a variant of Motik *et al.*'s definition: rather than considering all minimal interpretations, we require a canonical initial interpretation to exist, and for it to satisfy the integrity constraints.

Definition 2. *For ABoxes A over X, B over Y and F over $(X \uplus V \uplus Y)$, define $F : A \Rightarrow B$ whenever, for any interpretation $\mathcal{I}$ over X such that $\mathcal{I} \vDash S, T, A$, we have $I \oplus (S, F) \vDash T, B$.*

For example, in the example from Section 1 we have that in any $\mathcal{I}$ which satisfies the ambient TBoxes and $\mathsf{PersonalHomePage}(\texttt{alice:})$, there must be some i such that $(\texttt{alice:}^{\mathcal{I}}, i) \in \mathsf{primaryTopic}^{\mathcal{I}}$, so we can pick fresh j and k and define $\mathcal{J}$ as the smallest extension of $\mathcal{I}$ where:

$$\begin{array}{ccc} \texttt{bob:}^{\mathcal{J}} = j & \texttt{bob:me}^{\mathcal{J}} = k & \texttt{\_:anon}^{\mathcal{J}} = \texttt{alice:}^{\mathcal{I}} \\ j \in \mathsf{PersonalHomePage}^{\mathcal{J}} & j \in \mathsf{Document}^{\mathcal{J}} & k \in \mathsf{Person}^{\mathcal{J}} \\ (j,k) \in \mathsf{primaryTopic}^{\mathcal{J}} & (k,j) \in \mathsf{homepage}^{\mathcal{J}} & (k,i) \in \mathsf{knows}^{\mathcal{J}} \end{array}$$

and so we have:

$$\begin{pmatrix} \mathsf{PersonalHomePage}(\texttt{bob:}), \\ \mathsf{Person}(\texttt{bob:me}), \\ \mathsf{primaryTopic}(\texttt{bob:}, \texttt{bob:me}), \\ \mathsf{knows}(\texttt{bob:me}, \texttt{\_:anon}), \\ \mathsf{homepage}(\texttt{\_:anon}, \texttt{alice:}) \end{pmatrix} : (\mathsf{PersonalHomePage}(\texttt{alice:})) \Rightarrow \begin{pmatrix} \mathsf{PersonalHomePage}(\texttt{bob:}), \\ \mathsf{Person}(\texttt{bob:me}) \end{pmatrix}$$

5 Symmetric monoidal category

Having defined our notion of integrity constraints for Linked Data, we give our main result, which is that ABoxes with integrity constraints form a symmetric monoidal category, and hence (as shown by Joyal and Street [10] and surveyed, for example, by Selinger [16]) can be modeled formally by directed acyclic graphs.

A symmetric monoidal category $\mathbf{C}$ consists of:

- A collection $\mathbf{Obj}(\mathbf{C})$ of *objects*, including:
  - a chosen object I, and
  - for each pair of objects A and B, an object $A \otimes B$.
- For each pair of objects, A and B, a collection of *morphisms* $\mathbf{C}[A, B]$, including (where we write $f : A \to B$ whenever f is in $\mathbf{C}[A, B]$):
  - for each $f : A \to B$ and $g : B \to C$, a morphism $(f; g) : A \to C$,
  - for each $f : A \to C$ and $g : B \to D$, a morphism $(f \otimes g) : (A \otimes B) \to (C \otimes D)$, and
  - chosen families of morphisms:

$$\begin{aligned} 1_A &: A \to A & \sigma_{AB} &: (A \otimes B) \to (B \otimes A) \\ \alpha_{ABC} &: ((A \otimes B) \otimes C) \to (A \otimes (B \otimes C)) & \lambda_A &: (A \otimes I) \to A \\ \alpha^{-1}_{ABC} &: (A \otimes (B \otimes C)) \to ((A \otimes B) \otimes C) & \lambda^{-1}_A &: A \to (A \otimes I) \end{aligned}$$

  satisfying certain equations (see, for example Mac Lane [12] for details).

The objects of our symmetric monoidal category $\mathbf{ABox}$ will be ABoxes, which we will think of as interfaces.

- $\mathbf{Obj}(\mathbf{ABox})$ is the collection of all ABoxes.
- The chosen object I is the empty ABox.
- Given two ABoxes A over X and B over Y, the object $(A \otimes B)$ is the ABox (A, B) over $(X \uplus Y)$.

The morphisms of the category $\mathbf{ABox}$ will also be ABoxes, this time thought of as datasets satisfying integrity constraints.

- $\mathbf{ABox}[A, B]$ is the collection of all ABoxes F such that $F : A \Rightarrow B$.
- Given two ABoxes F over $(X \uplus V \uplus Y)$ and G over $(Y \uplus W \uplus Z)$, the morphism $(F; G)$ is the ABox (F, G) over $(X \uplus (V \uplus Y \uplus W) \uplus Z)$.
- Given two ABoxes F_1 over $(X_1 \uplus V_1 \uplus Y_1)$ and F_2 over $(X_2 \uplus V_2 \uplus Y_2)$, the morphism $(F_1 \otimes F_2)$ is the ABox (F_1, F_2) over $((X_1 \uplus X_2) \uplus (V_1 \uplus V_2) \uplus (Y_1 \uplus Y_2))$.

To verify that this definition is well-formed, we have to verify that checking integrity constraints is compositional, that is we only have to check integrity locally, and know it is preserved by composition and tensor.

Proposition 2.

1. *If $F : A \Rightarrow B$ and $G : B \Rightarrow C$, then $(F;G) : A \Rightarrow C$.*
2. *If $F_1 : A_1 \Rightarrow B_1$ and $F_2 : A_2 \Rightarrow B_2$, then $(F_1 \otimes F_2) : (A_1 \otimes A_2) \Rightarrow (B_1 \otimes B_2)$.*

Note that the composition $(F;G)$ may introduce bnodes, since the intermediate names which are exported by F and imported by G become bnodes (indeed, this is why bnodes are present in this model). For example:

$$\begin{aligned}&(\mathsf{knows}(\texttt{alice:me}, \texttt{bob:me})); (\mathsf{knows}(\texttt{bob:me}; \texttt{charlie:me}))\\ &\quad\equiv (\mathsf{knows}(\texttt{alice:me}, \texttt{\_:anon}), \mathsf{knows}(\texttt{\_:anon}, \texttt{charlie:me}))\end{aligned}$$

As well as composition of ABoxes, we have to provide the "wiring" combinators for identity, symmetry, unit and associativity. These are all constructed in the same way: given any function $f : Y \to X$ on finite sets, we define the ABox $\mathsf{wiring}(f)$ over $(X \uplus Y)$ as containing $f(y) \approx y$ for each $y \in Y$. We can then show that $\mathsf{wiring}(f)$ respects renaming of ABoxes. Given any ABox A over Y, let $f[A]$ be the ABox over X given by replacing any individual y in A by $f(y)$.

Proposition 3. *If $f : Y \to X$ and $B \subseteq f[A]$, then $\mathsf{wiring}(f) : A \Rightarrow B$.*

This suffices to define the combinators of a symmetric monoidal category, for example $1_A : A \Rightarrow A$ is given by wiring the identity function.

Finally, we have to prove the equations of a symmetric monoidal category. These equations are *not* true up to syntactic equality of ABoxes, due to introduction of bnodes, for example a counter-example to $1; F = F$ is:

$$\begin{aligned}&(\texttt{alice:me} \approx \texttt{alice:me}'); (\mathsf{knows}(\texttt{alice:me}', \texttt{bob:me}))\\ &\quad\equiv (\texttt{alice:me} \approx \texttt{\_:anon}, \mathsf{knows}(\texttt{\_:anon}, \texttt{bob:me}))\\ &\quad\neq (\mathsf{knows}(\texttt{alice:me}, \texttt{bob:me}))\end{aligned}$$

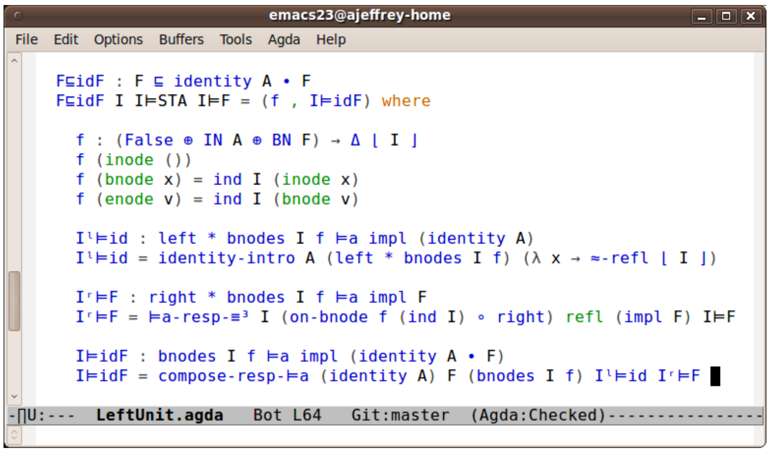
The equations *are* true when we consider ABoxes up to equivalence (in the presence of S, T and A), that is:

$$F \equiv G : A \Rightarrow B \text{ whenever } S, T, A, F \vDash G \text{ and } S, T, A, G \vDash F$$

We therefore consider the morphisms of **ABox** up to equivalence, which requires us to show that composition and tensor respect equivalence:

Proposition 4.

1. *If $F \equiv F' : A \Rightarrow B$ and $G \equiv G' : B \Rightarrow C$ then $(F;G) \equiv (F';G') : A \Rightarrow C$.*
2. *If $F_1 \equiv F_1' : A_1 \Rightarrow B_1$ and $F_2 \equiv F_2' : A_2 \Rightarrow B_2$
then $(F_1 \otimes F_2) \equiv (F_1' \otimes F_2') : (A_1 \otimes A_2) \Rightarrow (B_1 \otimes B_2)$.*



```
FᴇidF : F ⊑ identity A • F
F⊑idF I I⊨STA I⊨F = (f , I⊨idF) where

  f : (False ⊕ IN A ⊕ BN F) → Δ ⌊ I ⌋
  f (inode ())
  f (bnode x) = ind I (inode x)
  f (enode v) = ind I (bnode v)

  Iˡ⊨id : left * bnodes I f ⊨a impl (identity A)
  Iˡ⊨id = identity-intro A (left * bnodes I f) (λ x → ≈-refl ⌊ I ⌋)

  Iʳ⊨F : right * bnodes I f ⊨a impl F
  Iʳ⊨F = ⊨a-resp-≡³ I (on-bnode f (ind I) ∘ right) refl (impl F) I⊨F

  I⊨idF : bnodes I f ⊨a impl (identity A • F)
  I⊨idF = compose-resp-⊨a (identity A) F (bnodes I f) Iˡ⊨id Iʳ⊨F
```

```
-∏U:---  LeftUnit.agda   Bot L64   Git:master  (Agda:Checked)----------------
```

Fig. 3. Example of Agda proof mechanization

The proofs that ABoxes satisfy the equations of a symmetric monoidal category are then direct. The coherence properties (which only involve compositions of wiring morphisms) follow because wiring respects composition and tensor:

$$\mathsf{wiring}(f); \mathsf{wiring}(g) \equiv \mathsf{wiring}(f \circ g) \qquad \mathsf{wiring}(f) \otimes \mathsf{wiring}(g) \equiv \mathsf{wiring}(f \uplus g)$$

Theorem 1. ABox *forms a symmetric monoidal category.*

The proof of this theorem, including the definitions it relies on, is approximately 3,000 lines of Agda code [9]. An example lemma is shown in Figure 3.

6 Conclusions and further work

We have presented the first treatment of integrity constraints for Linked Data which makes use of a partition between local entities, for which a dataset is authoritative, and imported entities, where complete information is not known. We have given the first categorical presentation of datasets, and as a consequence, we have the first formal treatment of acyclic dependency graphs.

There are open questions raised by this model, of which the most important is its algorithmic properties: is integrity constraint satisfaction decidable, and if so, what is its complexity, and can it be reduced to existing decision problems?

Our model only treats acyclic dependency graphs, via symmetric monoidal categories. A categorical treatment of cyclic graphs uses *traced* monoidal categories (introduced by Joyal, Street and Verity [11], and discussed by Selinger [16]). Cyclic graphs require the existence of fixed points which unfortunately do not respect integrity constraint satisfaction, for example the fixed point of the identity morphism is equivalent to an empty dataset, which will not satisfy existential or disjunctive integrity constraints. The situation is similar to that of complete metric spaces: not all functions have fixed points, but contraction maps do.

Our model assumes the existence of ambient TBoxes S and T, which must be agreed upon by all datasets. This requirement is quite strong, and the model would be improved by allowing authoritative ontologies as well as datasets. This is related to the notion of *modularity* of ontologies [7].

The mechanized proofs of our model [9] are given in Agda [1], which as well as a proof assistant is a programming language which compiles to Haskell [3]. We hope to extend our proofs to a Semantic Web library, which will support the development of provably correct programs to process Linked Data.

References

1. The Agda programming language. `http://wiki.portal.chalmers.se/agda/`
2. The friend of a friend (FOAF) project, `http://www.foaf-project.org/`
3. The Haskell programming language. `http://haskell.org/`
4. Beckett, D., Berners-Lee, T.: Turtle - terse RDF triple language (2008), `http://www.w3.org/TeamSubmission/turtle/`
5. Berners-Lee, T.: Linked data (2006), `http://www.w3.org/DesignIssues/LinkedData.html`
6. Cyganiak, R., Jentzsch, A.: Linking open data cloud diagram, `http://lod-cloud.net/`
7. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artificial Intelligence Research 31, 273–318 (2008)
8. Jacobs, I., Walsh, N.: Architecture of the World Wide Web, volume one. W3C Recommendation (2004), `http://www.w3.org/TR/webarch/`
9. Jeffrey, A.S.A.: Agda libraries for the semantic web. `https://github.com/agda/agda-web-semantic/` (2011)
10. Joyal, A., Street, R.: The geometry of tensor calculus I. Advances in Mathematics 88(1), 55–112 (1991)
11. Joyal, A., Street, R., Verity, D.: Traced monoidal categories. Math. Proc. Cambridge Phil. Soc. 3, 447–468 (1996)
12. Mac Lane, S.: Categories for the Working Mathematician. Springer, 2nd edn. (1998)
13. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Semantics 7(2), 74–89 (2009)
14. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. J. ACM 57(5), 1–62 (2010)
15. Reiter, R.: What should a database know? J. Log. Program. 14, 127–153 (1992)
16. Selinger, P.: A survey of graphical languages for monoidal categories. In: Coecke, B. (ed.) New Structures for Physics, Lecture Notes in Physics, vol. 813, chap. 4, pp. 289–356. Springer (2011)
17. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Extending OWL with integrity constraints. In: Proc. Workshop on Description Logics. pp. 137–148 (2010)

Local Closed World Semantics: Keep it simple, stupid!

Adila Krisnadhi, Kunal Sengupta, and Pascal Hitzler

Wright State University, Dayton OH 45435, USA
{adila,kunal,pascal}@knoesis.org

Abstract. A combination of open and closed-world reasoning (usually called local closed world reasoning) is a desirable capability of knowledge representation formalisms for Semantic Web applications. However, none of the proposals made to date for extending description logics with local closed world capabilities has had any significant impact on applications. We believe that one of the key reasons for this is that current proposals fail to provide approaches which are intuitively accessible for application developers and at the same time are applicable, as extensions, to expressive description logics such as $\mathcal{SROIQ}$, which underlies the Web Ontology Language OWL.
In this paper we propose a new approach which overcomes key limitations of other major proposals made to date. It is based on an adaptation of circumscriptive description logics which, in contrast to previously reported circumscription proposals, is applicable to $\mathcal{SROIQ}$ without rendering reasoning over the resulting language undecidable.

Keywords: description logic, closed world, circumscription, decidability

1 Introduction

The semantics of the Web Ontology Language OWL [16] (which is based on the description logic $\mathcal{SROIQ}$ [17]) adheres to the Open World Assumption (OWA). This means that statements which are *not* logical consequences of a given knowledge base are not necessarily considered false. The OWA is reasonable in a World Wide Web context (and thus for Semantic Web applications), however situations naturally arise where it would be preferable to use the Closed World Assumption (CWA), that statements which are *not* logical consequences of a given knowledge base are always considered false. The CWA is applicable, e.g., when data is being retrieved from a database, or if data can otherwise be considered *complete* with respect to the application at hand (see, e.g., [14, 34]).

As a consequence, efforts have been made to combine OWA and CWA modeling for the Semantic Web (see Section 4), and knowledge representation languages which have both OWA and CWA modeling features are said to adhere to the *Local Closed World Assumption* (LCWA). Most of these combinations are derived from non-monotonic logics which have been studied in logic programming [18] or on first-order predicate logic [28, 29, 35]. Furthermore, many of them have a *hybrid* character, meaning that they achieve the LCWA by combining, e.g. description logics with (logic programming) rules.

Of the approaches which provide a seamless (non-hybrid) integration of OWA and CWA, there are not that many, and each of them has its drawbacks. This is despite the fact that the modeling task, from the perspective of the application developer, seems rather simple: Users would want to specify, simply, that individuals in the extension of a predicate should be exactly those which are *necessarily required* to be in the extension, i.e., extensions should be *minimized.* Thus, what is needed for applications is a simple, intuitive approach to closed world modeling, which can be easily picked up by application developers.

Among the primary approaches to non-monotonic reasoning, there is exactly one approach which employs the minimization idea in a very straightforward and intuitively simple manner, namely *circumscription* [28]. However, a naive transfer of the circumscription approach to description logics, which was done in [4, 5, 15], appears to have three primary drawbacks.

1. The approach is undedicable for expressive description logics (e.g., for the description logic $\mathcal{SROIQ}$) unless awkward restrictions are put in place. More precisely, it is not possible to have non-empty TBoxes plus minimization of roles if decidability is to be retained.
2. Extensions of minimized prediates can still contain elements which are not named individuals (or pairs of such, for roles) in the knowledge base, which is not intuitive for modeling (see also [14]).
3. Complexity of the approach is very high.

The undecidability issue (point 1) hinges, in a sense, also on point 2 above. In this paper, we provide a modified approach to circumscription for description logics, which we call *grounded circumscription*, which remedies both of points 1 and 2. We are not yet addressing the complexity issue; this will be done in future work. Our idea is simple yet effective: we modify the circumscription approach from [4, 5, 15] by adding the additional requirement that extensions of minimized predicates may only contain named individuals (or pairs of such, for roles). In a sense, this can be understood as porting a desirable feature from (hybrid) MNKF description logics [9, 20, 21, 32] to the circumscription approach. In another (but related) sense, it can also be understood as employing the idea of DL-safety [33], respectively of DL-safe variables [24] or nominal schemas [22, 23].

Note that we do not claim that our approach is the only road to take—we rather view it as one step on the quest of designing suitable LCWA languages for the Semantic Web. Indeed, we mainly intend to highlight that there is a plethora of methods how to obtain local closed world versions of description logics (and thus of OWL), see e.g. [25, 26], and all of them are potential alternatives to the *big three* (circumscription [28], autoepistemic logic [29], and default logic [35]). The Semantic Web community needs a systematic investigation of options for modeling local closed world aspects, which are not ideologically bound to approaches which have been developed for different purposes in the KR community.

The paper is structured as follows. In Section 2 we introduce the semantics of grounded circumscription. In Section 3 we show that the resulting language is decidable. In Section 4 we discuss related work, and conclude with a discussion of further work in Section 5.

2 Grounded Circumscription

We now describe a very simple way for ontology designers to model local closed world aspects in their ontologies: simply use a description logic (DL) knowledge base (KB) as usual, and augment it with *meta*-information which states that some predicates (concept names or role names) are *closed*. Semantically, those predicates are considered minimized, i.e. their extensions contain only what is absolutely required, and furthermore only contain *known* (or *named*) individuals, i.e., individuals which are explicitly mentioned in the KB. In the case of concept names, the idea of restricting their extensions only to known individuals is similar to the notion of nominal schema [23] (and thus, DL-safe rules [24, 33]) and also the notion of DBox [38], while the minimization idea is borrowed from circumscription [28], one of the primary approaches to non-monotonic reasoning.

In the earlier efforts to carry over circumscription to DLs [4, 5, 14, 15], circumscription is realized by the notion of *circumscription pattern*. A circumscription pattern consists of three disjoint sets of predicates (i.e., concept names and role names) which are called *minimized*, *fixed* and *varying* predicates, and a preference relation on interpretations. The preference relation allows us to pick *minimal* models as the *preferred* models with respect to inclusion of the extension of the minimized predicates.

Our formalism simplifies the circumscription approach by restricting our attention to models in which the extension of the minimized predicates may only contain known individuals from the KB. Moreover, we divide predicates in the KB only into two disjoint sets of minimized and non-minimized predicates.[1] The non-minimized predicates would be viewed as varying in the more general circumscription formalism mentioned above.

Let N_C, N_r, and N_I be disjoint, countably infinite sets of *concept names*, *role names*, and *individual names*, resp. Let $\mathcal{L}$ be a standard description logic whose concepts and roles are formed based on the signature that consists of N_C, N_R, and N_I, together with a set of standard DL (concept and role) constructors [2]. The only non-standard DL constructor that is needed in this paper is the role constructor *concept product*, written $C \times D$ with C, D concepts in $\mathcal{L}$, which allows a role to be constructed from the Cartesian product of two concepts [23, 37]. In addition, we define an *$\mathcal{L}$-KB* as a set of concept inclusion axioms $C \sqsubseteq D$ where $C, D \in \mathsf{N}_C$, role inclusion axioms $r \sqsubseteq s$ where $r, s \in \mathsf{N}_\mathsf{r}$, and assertions of the form $C(a)$ and $r(a, b)$ where $C \in \mathsf{N}_C, r \in \mathsf{N}_r$ and $a, b \in \mathsf{N}_I$.

The semantics for $\mathcal{L}$ is defined in terms of *interpretations* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty set called the *domain* and $\cdot^\mathcal{I}$ is an *interpretation function*

[1] Fixed predicates can be simulated in the original circumscriptive DL approach if negation is available, i.e., for fixed class names, class negation is required, while for fixed role names, role negation is required. The latter can be added to expressive DLs without jeopardizing decidability [23, 40].

that maps each concept name to a subset of $\Delta^{\mathcal{I}}$, each role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual name to an element of $\Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is extended to complex concepts and roles in the usual way for $\mathcal{L}$, and for concept products, $(C \times D)^{\mathcal{I}} = \{(x, y) \mid x \in C^{\mathcal{I}}, y \in D^{\mathcal{I}}\}$. We say that $\mathcal{I}$ *satisfies* (*is a model of*): a concept inclusion axiom $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; a role inclusion axiom $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$; a concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$; and a role assertion $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. We also say that $\mathcal{I}$ *satisfies* (*is a model of*) an $\mathcal{L}$-KB K if it satisfies every axioms in K.

The non-monotonic feature of the formalism is given by restricting models of an $\mathcal{L}$-KB such that the extension of closed predicates may only contain individuals (or pairs of them) which are explicitly occurring in the KB, plus a minimization of the extensions of these predicates. We define a function Ind that maps each $\mathcal{L}$-KB to the set of individual names it contains, i.e., given an $\mathcal{L}$-KB K, $\mathsf{Ind}(K) = \{b \in \mathsf{N}_I \mid b \text{ occurs in } K\}$. Among all possible models of K that are obtained by the aforementioned restriction to $\mathsf{Ind}(K)$, we then select a model that is minimal w.r.t. concept inclusion or role inclusion.

Definition 1. *A* GC-$\mathcal{L}$-knowledge base *(KB)—GC stands for* grounded circumscription*—is a pair* (K, M) *where* K *is an* $\mathcal{L}$*-KB and* $M \subseteq \{A \in \mathsf{N}_C \mid A \text{ occurs in } K\} \cup \{r \in \mathsf{N}_r \mid r \text{ occurs in } K\}$*. For every concept name and role name* $W \in M$*, we say that* W *is* closed with respect to K*. For any two models* $\mathcal{I}$ *and* $\mathcal{J}$ *of* K*, we furthermore say that* $\mathcal{I}$ *is* smaller than $\mathcal{J}$ w.r.t. M*, written* $\mathcal{I} \prec_M \mathcal{J}$*, iff all of the following hold: (i)* $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ *and* $a^{\mathcal{I}} = a^{\mathcal{J}}$ *for every* $a^{\mathcal{I}} \in \Delta^{\mathcal{J}}$*; (ii)* $W^{\mathcal{I}} \subseteq W^{\mathcal{J}}$ *for every* $W \in M$*; and (iii) there exists a* $W \in M$ *such that* $W^{\mathcal{I}} \subset W^{\mathcal{J}}$

We now define models of GC-$\mathcal{L}$-KBs as follows.

Definition 2. *An interpretation* $\mathcal{I}$ *is a* GC-model *of a GC-*$\mathcal{L}$*-KB* (K, M) *if all of the following hold: (i)* $\mathcal{I}$ *is a model of* K*; (ii) for each concept name* $A \in M$*,* $A^{\mathcal{I}} \subseteq \{b^{\mathcal{I}} \mid b \in \mathsf{Ind}(K)\}$*; (iii) for each role name* $r \in M$*,* $r^{\mathcal{I}} \subseteq \{b^{\mathcal{I}} \mid b \in \mathsf{Ind}(K)\} \times \{b^{\mathcal{I}} \mid b \in \mathsf{Ind}(K)\}$*; and (iv)* $\mathcal{I}$ *is minimal w.r.t.* M*, i.e., there is no model* $\mathcal{J}$ *of* K *such that* $\mathcal{J} \prec_M \mathcal{I}$*.*

The notion of logical consequence is defined as usual: An axiom α is a logical consequence (a *GC-inference*) of a given GC-$\mathcal{L}$-KB (K, M) if and only if α is true in all GC-models of (K, B).

Our formalism here is inspired by one of the approaches described by Makinson in [26], namely restricting the set of valuations to get more logical consequences than what we can get as classical consequences. Intuitively, this approach is a simpler version of the circumscription formalism for DLs as presented in [5, 15] in the sense that concept names and role names are either varying or minimized, i.e., no predicate is considered fixed. Indeed, every GC-model of a KB is also a circumscriptive model,[2] hence every circumscriptive inference is also a valid GC-inference.

[2] This can be seen, e.g., by a straightforward proof by contradiction.

To give an example, consider the knowledge base K consisting of the axioms

$$\begin{array}{ll}\mathtt{hasAuthor}(\mathtt{paper1},\mathtt{author1}) & \mathtt{hasAuthor}(\mathtt{paper1},\mathtt{author2}) \\ \mathtt{hasAuthor}(\mathtt{paper2},\mathtt{author3}) & \top \sqsubseteq \forall\mathtt{hasAuthor.Author}.\end{array}$$

Then $(\leq 2\ \mathtt{hasAuthor.Author})(\mathtt{paper1})$ is not a logical consequence of K under the classical description logic semantics. However, if we assume that we have complete information on authorships relevant to the application under consideration, then it would be reasonable to *close* parts of the knowledge base in the sense of the LCWA. In the original approach to circumscriptive DLs, we could close the class name `Author`, but to no avail. But if we close `hasAuthor`, we obtain $(\leq 2\ \mathtt{hasAuthor.Author})(\mathtt{paper1})$ as a logical consequence. However, closure of roles in the original circumscriptive DL approach leads to undecidability [5]. The GC-semantics, in contrast, is decidable even under role closure (see Section 3 below), and also yields the desired inferences.

Are there inferences which hold with respect to the GC-semantics but not with respect to the original circumscriptive DL approach? There are, but it seems difficult to find a *convincing* example which might indicate practical relevance. If this is indeed the case, then we could argue that the original circumscriptive approach is too sceptical with respect to application requirements, in addition to the decidability issue already noted.

The following is an *academic* example, adapted from [15], which shows the different inferencing capabilities of the GC-semantics versus the original circumscriptive DL semantics. Consider the knowledge base K_1 consisting of the following axioms, where `EndangeredSpecies` is a minimized class name.

$$\mathtt{Bear}(\mathtt{polarBear})$$
$$\exists\mathtt{isHabitatFor}.(\mathtt{Bear}\sqcap\mathtt{EndangeredSpecies})(\mathtt{arcticSea})$$

In the original circumscriptive DL approach, there is a model in which the extensions of both `Bear` and `EndangeredSpecies` share a common element distinct from `polarBear`, hence it cannot be concluded that `polarBear` is an `EndangeredSpecies`. Under the GC-semantics, however, this can be concluded. This is due to the fact that there are no individuals other than `polarBear` in the knowledge base. Indeed, if we assume that there is another individual, say, `blueWhale`, then the conclusion no longer holds even under the GC-semantics.

Is the conclusion under the GC-semantics desirable, that `polarBear` is an `EndangeredSpecies`? We believe so, because we are essentially restricting our world to one individual. I.e., if we would like to reject the conclusion, we should rather question the adequacy of our modeling, than of the semantics. However, this discussion seems to be quite academic, since the situation above is not that of a realistic knowledge base, where we could reasonably assume the presence of other individuals, such as `blueWhale`, such that the arguable inference no longer holds even with respect to the GC-semantics.[3] And indeed it should not hold

[3] The situation might be different with respect to knowledge bases *under development*, but this would rather be an interface issue.

in this case under an intuitive reading of the knowledge base: If there is also a second individual `blueWhale`, then we have no reason to assume that it must be `polarBear` which is an `EndangeredSpecies` (unless, of course, we also state that `blueWhale` must not be a `Bear`).

3 Decidability Considerations

As noted earlier, circumscription in many expressive DLs is undecidable [5]. Undecidability even extends to the basic DL $\mathcal{ALC}$ when non-empty TBoxes are considered and roles are allowed as minimized predicates. Such a bleak outlook would greatly discourage useful application of circumscription, despite the fact that there is a clear need of such a formalism to model LCWA.

Our formalism aims to fill this gap by offering a simpler approach to circumscription in DLs that is decidable provided that the underlying DL is also decidable. The decidability result is obtained due to the imposed restriction of minimized predicates to known individuals in the KB as specified in Definition 2. Let $\mathcal{L}$ be any standard DL. We consider the following reasoning task of *GC-KB satisfiability*: "given a GC-$\mathcal{L}$-KB (K, M), does (K, M) have a GC-model?" and show in the following that this is decidable. Note that other basic reasoning tasks can usually be reduced to this task [5, 15].

Assume that $\mathcal{L}$ is any (standard) DL, e.g., $\mathcal{ALCOB}(\times)$, featuring nominals, concept disjunction, concept products and role disjunctions.[4] We show that GC-KB satisfiability in $\mathcal{L}$ is decidable if satisfiability in $\mathcal{L}$ is decidable.

Let (K, M) be a GC-$\mathcal{L}$-KB. We assume that $M = M_A \cup M_r$ where $M_A = \{A_1, \ldots, A_n\}$ is the set of minimized concept names and $M_r = \{r_1, \ldots, r_m\}$ is the set of minimized role names. Now define a family of $(n+m)$-tuples as

$$\mathcal{G}_{(K,M)} = \{(X_1, \ldots, X_n, Y_1, \ldots, Y_m) \mid X_i \subseteq \mathsf{Ind}(K), Y_j \subseteq \mathsf{Ind}(K) \times \mathsf{Ind}(K)\}$$

with $1 \le i \le n, 1 \le j \le m$. Note that there are

$$\left(2^{|\mathsf{Ind}(K)|}\right)^n \cdot \left(2^{\mathsf{Ind}(K)^2}\right)^m = 2^{n\cdot|\mathsf{Ind}(K)|+m\cdot|\mathsf{Ind}(K)|^2} \tag{1}$$

of such tuples; in particular note that $\mathcal{G}_{(K,M)}$ is a finite set.

Now, given (K, M) and some $G = (X_1, \ldots, X_n, Y_1, \ldots, Y_m) \in \mathcal{G}_{(K,M)}$, let K_G be the $\mathcal{L}$-KB consisting of all axioms in K together with all of the following axioms, where the A_i and r_j are all the predicates in M—note that we require role disjunction and concept products for this.

$$A_i \equiv \bigsqcup\{a\} \quad \text{for every } a \in X_i \text{ and } i = 1, \ldots, n$$

$$r_j \equiv \bigsqcup(\{a\} \times \{b\}) \quad \text{for every pair } (a, b) \in Y_j \text{ and } j = 1, \ldots, m$$

Then the following result clearly holds.

[4] For concept products, see [23]—they can be eliminated if role constructors are available. For role disjunctions, see [40], where it is shown, amongst other things, that $\mathcal{ALCQIOB}$ is decidable.

Lemma 1. *Let (K, M) be a GC-$\mathcal{L}$-KB. If (K, M) has a GC-model $\mathcal{I}$, then there exists $G \in \mathcal{G}_{(K,M)}$ such that K_G has a (classical) model $\mathcal{J}$ which coincides with $\mathcal{I}$ on all minimized predicates. Likewise, if there exists $G \in \mathcal{G}_{(K,M)}$ such that K_G has a (classical) model $\mathcal{J}$, then (K, M) has a GC-model $\mathcal{I}$ which coincides with $\mathcal{J}$ on all minimized predicates.*

Observe that class disjunction, nominals, concept products, and role disjunction are needed to obtain Lemma 1. From [40] we know that adding role disjunction to $\mathcal{ALCQIO}$ retains decidability. Now consider the set

$$\mathcal{G}'_{(K,M)} = \{G \in \mathcal{G}_{(K,M)} \mid K_G \text{ has a (classical) model}\},$$

and note that this set is finite and computable in finite time since $\mathcal{G}_{(K,M)}$ is finite and $\mathcal{L}$ is decidable. Furthermore, consider $\mathcal{G}'_{(K,M)}$ to be ordered by the pointwise ordering $\prec$ induced by $\subseteq$. Note that the pointwise ordering of the finite set $\mathcal{G}'_{(K,M)}$ is also computable in finite time.

Lemma 2. *Let (K, M) be a GC-$\mathcal{L}$-KB and let*

$$\mathcal{G}''_{(K,M)} = \{G \in \mathcal{G}'_{(K,M)} \mid G \text{ is minimal in } (\mathcal{G}'_{(K,M)}, \prec)\}.$$

Then (K, M) has a GC-model if and only if $\mathcal{G}''_{(K,M)}$ is non-empty.

Proof. This follows immediately from Lemma 1 together with the following observation: Whenever K has two GC models $\mathcal{I}$ and $\mathcal{J}$ such that $\mathcal{I}$ is smaller than $\mathcal{J}$, then there exist $G_{\mathcal{I}}, G_{\mathcal{J}} \in \mathcal{G}'_{(K,M)}$ with $G_{\mathcal{I}} \prec G_{\mathcal{J}}$ such that $K_{G_{\mathcal{I}}}$ and $K_{G_{\mathcal{J}}}$ have (classical) models $\mathcal{I}'$ and J', respectively, which coincide with $\mathcal{I}$, respectively, $\mathcal{J}$, on the minimized predicates.

Theorem 1. *GC-KB-satisfiability is decidable.*

Proof. This follows from Lemma 2 since the set $\mathcal{G}''_{(K,M)}$, for any given GC-KB (K, M), can be computed in finite time, i.e., it can be decided in finite time whether $\mathcal{G}''_{(K,M)}$ is empty.

Some remarks on complexity are as follows. Assume that the problem of deciding KB satisfiability in $\mathcal{L}$ is in the complexity class C. Observe from equation (1) that there are exponentially many possible choices of the $(n+m)$-tuples in $\mathcal{G}_{(K,M)}$ (in the size of the input knowledge base). Computation of $\mathcal{G}'_{(K,M)}$ is thus in $\textsc{Exp}^{\mathrm{C}}$, and subsequent computation of $\mathcal{G}''_{(K,M)}$ is also in $\textsc{Exp}$. We thus obtain the following upper bound.

Proposition 1. *GC-KB satisfiability is in $\textsc{Exp}^{C}$, where C is the complexity class of the DL under consideration.*

Observe that the decidability proof gives rise to a straightforward implementation procedure, however this is certainly not a smart algorithm. As future work, it should be possible to adjust the tableaux algorithm from [15], which may also give rise to a sharpening of the upper bound on complexity.

4 Related Work

In this paper we have presented a new approach to DL reasoning under the Local Closed World Assumption (LCWA). There are several approaches described in the literature for LCWA which combine the OWA and CWA semantics, and in the following we briefly discuss some of the most important proposals.

Autoepistemic Logic [29, 30] is an approach followed by a number of authors. The semantics of autoepistemic logic have been defined using an autoepistemic operator **K** [7, 8] and has been studied for $\mathcal{ALC}$ and also for more expressive DLs. [7, 9] further provide an epistemic operator **A** related to negation-as-failure which allows for the modeling of default rules and integrity constraints.

Circumscription [28] is another approach taken to develop LCWA extentions of DLs [5, 14, 15]. [5] evaluates the complexities of reasoning problems in variations of DLs with circumscription. [14] provides examples to stress the importance of LCWA to provide an intuitive notion of matchmaking of resources in the context of Semantic Web Services. [15] provides an algorithmization for circumscriptive $\mathcal{ALCO}$ by introducing a preferential tableaux calculus, based on previous work on circumscription [4]. [19] proves a method to eliminate fixed predicates in circumscription patterns by adding negation of fixed predicates to the minimized set of predicates.

Some significant proposals involve the use of hybrid MKNF knowledge bases [32] which are based on an adaptation of the Stable Model Semantics [12] to knowledge bases consisting of ontology axioms and rules, thereby combining both open world and closed world semantics. A variant of this approach using the well-founded semantics, i.e., with a lower complexity, has also be presented [20, 21], and algorithms and implementations have been developed [1, 13].

[10] takes a hybrid approach to combine ontologies and rules by keeping the semantics of both parts separate, but also at the same time allowing for building rules on top of ontologies and vice versa with some limitations, again following the Stable Model Semantics. [11] provides a related well-founded semantics.

Some of the work related to LCWA also involves the use of integrity constraints (ICs) and of the Unique Name Assumption (UNA). An approach extending OWL ontologies to add ICs such that it adds non-montonicity to the DL is [31]. [39] provides semantics for OWL axioms to allow for IC and UNA to achieve local closed world reasoning.

In [38], the notion of *DBox* is introduced. A DBox consists of a set of (atomic) assertions such that the extension of a DBox predicate under any interpretation is exactly as defined by this set of assertions. In a sense, grounded circumscription encompasses this expressive feature but goes beyond it, while, as expected, loosing some of the desirable features of the more specialized DBox approach.

There are a number of other approaches which have been attempted in the past, but without follow-up work, e.g. [3, 6, 27, 36]. For some further pointers to the literature, please refer to [22].

5 Conclusion and Outlook

We have provided a new approach for incorporating the LCWA into description logics. Our approach, grounded circumscription, is a variant of circumscriptive description logics which avoids two major issues of the original approach: Extensions of minimized predicates can only contain named individuals, and we retain decidability even for very expressive description logics while we can allow for the minimization of roles.

A primary theoretical task is to investigate the complexity of our approach, but it can be expected that it is not going to be worse than the previous circumscription proposal. In fact, lower complexities should result in some cases, which may yield to tractable or data-tractable fragments.

Likewise, it should be possible to adapt the tableaux algorithm for circumscriptive description logics from [15] to our setting, and there may even be more efficient procedures.

From a more general perspective, it should be worthwhile to investigate further alternatives for incorporating closed world modeling into description logics. Preferably, one would like to obtain a language which is intuitively very simple, appeals to ontology engineers, and is computationally effective. Whether such a language exists, however, is an open question.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 "III: Small: TROn—Tractable Reasoning with Ontologies," and by State of Ohio Research Incentive funding in the Kno.e.CoM project. The first named author acknowledges support by a Fulbright Indonesia Presidential Scholarship PhD Grant 2010.

References

1. Alferes, J.J., Knorr, M., Swift, T.: Queries to Hybrid MKNF Knowledge Bases through Oracular Tabling. In: Proc. of the 8th International Semantic Web Conference. pp. 1–16. ISWC '09, Springer-Verlag (2009)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2007)
3. Baader, F., Hollunder, B.: Embedding Defaults into Terminological Knowledge Representation Formalisms. Journal of Automated Reasoning 14(1), 149–180 (1995)
4. Bonatti, P.A., Lutz, C., Wolter, F.: Expressive Non-Monotonic Description Logics Based on Circumscription. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06). pp. 400–410. AAAI Press (2009)
5. Bonatti, P.A., Lutz, C., Wolter, F.: The Complexity of Circumscription in Description Logic. Journal of Artificial Intelligence Research 35, 717–773 (2009)
6. de Bruijn, J., Pearce, D., Polleres, A., Valverde, A.: A Semantical Framework for Hybrid Knowledge Bases. Knowledge and Information Systems 25(1), 81–104 (2010)
7. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: An Epistemic Operator for Description Logics. Artificial Intelligence 100(1-2), 225–274 (1998)

8. Donini, F.M., Nardi, D., Rosati, R.: Autoepistemic Description Logics. In: Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97). pp. 136–141 (1997)
9. Donini, F.M., Nardi, D., Rosati, R.: Description Logics of Minimal Knowledge and Negation as Failure. ACM Trans. on Computational Logic (TOCL) 3(2), 177–225 (April 2002)
10. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description logics for the Semantic Web. Artificial Intelligence 172(12-13), 1495–1539 (August 2008)
11. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-Founded Semantics for Description Logic Programs in the Semantic Web. In: Antoniou, G., Boley, H. (eds.) Rules and Rule Markup Languages for the Semantic Web: 3rd International Workshop, RuleML 2004. Lecture Notes in Computer Science, vol. 3323, pp. 81–97. Springer (2004)
12. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
13. Gomes, A., Alferes, J., Swift, T.: Implementing Query Answering for Hybrid MKNF Knowledge Bases. In: Carro, M., Peña, R. (eds.) Practical Aspects of Declarative Languages, Lecture Notes in Computer Science, vol. 5937, pp. 25–39. Springer Berlin / Heidelberg (2010)
14. Grimm, S., Hitzler, P.: Semantic Matchmaking of Web Resources with Local Closed-World Reasoning. International Journal of Electronic Commerce 12(2), 89–126 (December 2007)
15. Grimm, S., Hitzler, P.: A Preferential Tableaux Calculus for Circumscriptive $\mathcal{ALCO}$. In: Polleres, A., Swift, T. (eds.) Proc. of the 3rd Int. Conference on Web Reasoning and Rule Systems (RR'09). Lecture Notes in Computer Science, vol. 5837, pp. 40–54. Springer Berlin (2009)
16. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009 (2009), available from http://www.w3.org/TR/owl2-primer/
17. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
18. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Semantics. CRC Press (2010)
19. Kleer, J.D., Konolige, K.: Eliminating the Fixed Predicates from a Circumscription. Artificial Intelligence 39(3), 391–398 (1989)
20. Knorr, M., Alferes, J., Hitzler, P.: Local Closed-World Reasoning with Description Logics under the Well-founded Semantics. Artificial Intelligence 175(9–10), 1528–1554 (2011)
21. Knorr, M., Alferes, J.J., Hitzler, P.: A Coherent Well-founded Model for Hybrid MKNF Knowledge Bases. In: Ghallab, M., et al. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence,Patras, Greece, July 21-25, 2008. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 99–103. IOS Press, Amsterdam, The Netherlands (2008)
22. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and Rules. In: Reasoning Web 2011. Lecture Notes in Computer Science, Springer, Heidelberg (2011), to appear
23. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Sadagopan, S., et al. (eds.) Proceedings of the 20th International World Wide Web Conference, WWW2011, Hyderabad, India, March/April 2011. pp. 645–654. ACM, New York (2011)

24. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: Sheth, A.P., et al. (eds.) Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Lecture Notes in Computer Science, vol. 5318, pp. 649–664. Springer (2008)
25. Makinson, D.: Bridges between Classical and Nonmonotonic Logic. Logic Journal of the IGPL 11(1), 69–96 (2003)
26. Makinson, D.: Bridges from Classical to Nonomonotonic Logic, Texts in Computing, vol. 5. King's College Publications (2005)
27. Matzner, T., Hitzler, P.: Any-world access to OWL from Prolog. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) Proceedings of the 30th Annual German Conference on Artificial Intelligence, Osnabrück, Germany, September 2007. Lecture Notes in Artificial Intelligence, vol. 4667, pp. 84–98. Springer (2007)
28. McCarthy, J.: Circumscription – A Form of Non-Monotonic Reasoning. Artificial Intelligence 13(1–2), 27–39 (1980)
29. Moore, R.: Possible-worlds Semantics for Autoepistemic Logic. In: Proceedings of the 1984 Non-monotonic Reasoning Workshop. AAAI, Menlo Park, CA (1984)
30. Moore, R.: Semantical Considerations on Nonmonotonic Logic. Artificial Intelligence 25(1) (1985)
31. Motik, B., Horrocks, I., Sattler, U.: Adding Integrity Constraints to OWL. In: Golbreich, C., Kalyanpur, A., Parsia, B. (eds.) Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions. vol. 258 (2007)
32. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. Journal of the ACM 57(5), 1–62 (2010)
33. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. Journal of Web Semantics 3, 41–60 (July 2005)
34. Patel, C., Cimino, J.J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching Patient Records to Clinical Trials Using Ontologies. In: Aberer, K., et al. (eds.) The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Lecture Notes in Computer Science, vol. 4825, pp. 816–829. Springer (2007)
35. Reiter, R.: A Logic for Default Reasoning. Artificial Intelligence 13, 81–132 (1980)
36. Ren, Y., Pan, J.Z., Zhao, Y.: Closed world reasoning for OWL2 with NBox. Journal of Tsinghua Science and Technology 15(6) (2010)
37. Rudolph, S., Krötzsch, M., Hitzler, P.: All Elephants are Bigger than All Mice. In: Baader, F., Lutz, C., Motik, B. (eds.) Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008. CEUR Workshop Proceedings, vol. 353 (2008)
38. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Boutilier, C. (ed.) IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 923–925 (2009)
39. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Fox, M., Poole, D. (eds.) Proc. of the 24th AAAI Conference on Artificial Intelligence, AAAI 2010. AAAI Press (2010)
40. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. thesis, RWTH Aachen, Germany (2001)

A Technique for Handling the Right Hand Side of Complex RIAs

Milenko Mosurović[1] and Nenad Krdžavac[2]

[1] Faculty of Mathematics and Natural Science,
University of Montenegro, Podgorica,
ul. Džordža Vašingtona bb, 81000 Podgorica, Montenegro
[2] School of Electrical and Computer Engineering of Applied Studies in Belgrade,
University of Belgrade, Serbia,
ul. Vojvode Stepe 283, 11000 Belgrade, Serbia

Abstract. This paper examines a new technique based on tableau, that allows one to introduce composition of roles from the right hand side of complex role inclusion axioms (RIAs). Our motivation comes from modeling product models in manufacturing systems. The series of papers, so far, have studied the extension of tableau algorithm for Description Logics (DLs) to capture complex RIAs. However, such RIAs permit only the left hand side of the composition of roles. To illustrate the technique, we extend $\mathcal{RIQ}$ DL with one RIA of the form $R \dot{\sqsubseteq} Q \circ P$.

Keywords: Description Logic, Manufacturing system,Tableau.

1 Introduction

Description Logics [1] are a well-established branch of logics for knowledge representation and reasoning about it. Recent research in DLs has usually focused on the logics of the so-called $\mathcal{SH}$ family as basis for the standard Web Ontology Languages (OWL) [10]. In particular, the DL $\mathcal{SHIQ}$ [8] is closely related to OWL-Lite and extends the basic $\mathcal{ALC}$ [1](the minimal propositionally closed DL) with inverse roles and number restrictions, as well as with role inclusions and transitive roles. The DL known as $\mathcal{SHOIQ}$ [7], underlying OWL-DL, further extends $\mathcal{SHIQ}$ with nominals. Logics, $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ were enhanced with regular role hierarchies in which the composition of a chain of roles may imply another role. These and other features were included in their extensions known as $\mathcal{SRIQ}$ [9] and $\mathcal{SROIQ}$ [5] respectively; the latter underlies the new OWL 2 [4] standard. For reasoning in them, the adaptations of the tableaux algorithms were proposed [9, 5]. In a pre-processing stage, the implications between roles, given by the role hierarchy, are captured in a set of non-deterministic finite state automata (NFA). The complexity of these logics is studied in [11]. Also, there exists another extensions of the logics with description graphs [14] and stratified ontologies [12]. Motivation for our research is based on modeling product models in manufacturing systems (see UML model on Figure 1) [3]. For example, when

an individual crankshaft in individual engine in an individual car, powers individual hubs in individual wheels in the same car, and not the hubs in the wheels in the other cars [13]. Such modeling example can be represented as RIAs with more then one role on the right hand side of role composition [13]. The aim of this paper is to show a technique that can allow the extension of $\mathcal{RIQ}$ DL [6] with a RIA of the form $R\dot{\sqsubseteq}Q \circ P$. The $\mathcal{RIQ}$ DL [6], is the fragment of $\mathcal{SRIQ}$ (without Abox, as well as, reflexive, symmetric, transitive, and irreflexive roles, disjoint roles, and the construct $\exists R.Self$) [9]. To avoid analysis of restrictions that roles must satisfy in new RIAs, we consider only one RIA of the form $R\dot{\sqsubseteq}Q \circ P$. Main idea is to define a new role (P^{-}, x) that remembers in which object is related to the role. We define new constructor which will deal with these roles.
The paper is organized as follows. Next section gives short overview of $\mathcal{RIQ}$ DL and its role hierarchy. Section (3) explains simple reduction problem and gives general idea. Section (4), outlines the extension of $\mathcal{RIQ}$ tableau, while section (5) gives formal proof of the correctness and termination of tableau algorithm. Finaly we give some remarks and explane future work.

2 Preliminaries

This section, in brief, outlines syntax and semantics of $\mathcal{RIQ}$ DL and regular role hierarchy. The alphabet of $\mathcal{RIQ}$ DL consists of set of concept names $\mathcal{N}_C$, set of role names $\mathcal{N}_R$ and finally, set of simple role names $\mathcal{N}_S \subset \mathcal{N}_R$. The set of roles is $\mathcal{N}_R \cup \{R^{-} | R \in \mathcal{N}_R\}$. According to [6], syntax and semantics of the $\mathcal{RIQ}$ DL concepts are given in definitions 1 and 2.

Definition 1. *Set of $\mathcal{RIQ}$ concepts is a smallest set such that*

- *every concept name and $\top$, $\bot$ are concepts, and,*
- *if C and D are concept and R is a role, S is simple role, n is non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, $\exists R.C$, $(\leq nS.C)$, $(\geq nS.C)$ are concepts.* $\Box$

The semantics of the $\mathcal{RIQ}$ DL is defined by using interpretation. An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of the interpretation. A valuation $\cdot^{\mathcal{I}}$ associates: every concept name C with a subset $C^{I} \subseteq \Delta^{\mathcal{I}}$; every role name R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ [6].

Definition 2. *An interpretation $\mathcal{I}$ extends to $\mathcal{RIQ}$ complex concepts and roles according to the following semantic rules:*

- *If R is a role name, then $(R^{-})^{\mathcal{I}} = \{\langle x, y\rangle : \langle y, x\rangle \in R^{\mathcal{I}}\}$,*
- *If R_1, R_2,..., R_n are roles then $(R_1 R_2 \ldots R_n)^{\mathcal{I}} = (R_1)^{\mathcal{I}} \circ (R_2)^{\mathcal{I}} \circ \cdots \circ (R_n)^{\mathcal{I}}$, where sign $\circ$ is a composition of binary relations,*
- *If C and D are concepts, R is a role, S is a simple role and n is a non-negative integer, then* [3]

[3] $\sharp M$ denotes cardinality of set M.

$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}, \bot^{\mathcal{I}} = \emptyset, (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}},$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},$
$(\exists R.C)^{\mathcal{I}} = \{x \, : \, \exists y. \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\},$
$(\forall R.C)^{\mathcal{I}} = \{x \, : \, \forall y. \langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\},$
$(\geq nS.C)^{\mathcal{I}} = \{x \, : \, \sharp\{y \, : \, \langle x, y\rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\},$
$(\leq nS.C)^{\mathcal{I}} = \{x \, : \, \sharp\{y \, : \, \langle x, y\rangle \in S^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}.$
Number restrictions ($\geq nS.C$) and ($\leq nS.C$), are restricted to simple roles, in order to have $\mathcal{RIQ}$ decidability. □

Strict partial order $\prec$ (irreflexive, transitive, and antisymmetric), on the set of roles, provides acyclicity [6]. Allowed RIAs in $\mathcal{RIQ}$ DL with respect to $\prec$, are expressions of the form $w \dot{\sqsubseteq} R$, where [6]:

1. R is a simple role name, $w = S$ and $S \prec R$ is a simple role,
2. $R \in \mathcal{N}_R \backslash \mathcal{N}_S$ is a role name and
   (a) $w = RR$, or
   (b) $w = R^-$, or
   (c) $w = S_1 \cdots S_n$ and $S_i \prec R$, for $1 \leq i \leq n$, or
   (d) $w = RS_1 \cdots S_n$ and $S_i \prec R$, for $1 \leq i \leq n$, or
   (e) $w = S_1 \cdots S_n R$ and $S_i \prec R$, for $1 \leq i \leq n$.

Note that the notion of simple role has the same meaning as defined in [5]. So, we use the simple role S carefully in allowed RIAs to avoid $R_1 \circ R_2 \sqsubseteq S$.

A $\mathcal{RIQ}$ RBox (role hierarchy) is a finite set $\mathcal{R}$ of RIAs. A role hierarchy $\mathcal{R}$ is regular if there exists strict partial order $\prec$ such that each RIA in $\mathcal{R}$ is regular [6]. An interpretation $\mathcal{I}$ satisfies a RIA $S_1 \cdots S_n \dot{\sqsubseteq} R$, if $S_1^{\mathcal{I}} \circ \cdots \circ S_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. A $\mathcal{RIQ}$ concept C is satisfiable w.r.t. RBox $\mathcal{R}$ if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$ and $\mathcal{I}$ satisfies all RIA in $\mathcal{R}$ [6, 11]. In this paper we extend regular $\mathcal{RIQ}$-RBox with one RIA of the form

$$w \dot{\sqsubseteq} Q \circ P \tag{1}$$

where $w = S_1 \circ S_2 \cdots S_n$, $S_i \prec Q$, $P \prec Q$ and there is no i, such that $P \prec S_i$. An interpretation $\mathcal{I}$ satisfies a RIA of the form $w \dot{\sqsubseteq} Q \circ P$, if $w^{\mathcal{I}} \subseteq Q^{\mathcal{I}} \circ P^{\mathcal{I}}$. In the rest of the paper we check satisfiability of concept C_0 w.r.t. defined RBox $\mathcal{R}$ and define $\mathcal{R}_{C_0} = \{R | R \text{ is role that occurs in } C_0 \text{ or } \mathcal{R}\}$.

3 The simple reduction and general idea

Tableau algorithm in [6] tries to construct a tableau for $\mathcal{RIQ}$-concept C. In preprocessing step the role hierarchy is translated into NFA, that are used, both, in the definition of a tableau and in the tableau algorithm. Intuitively, an automaton is used to memorize path between an object x that has to satisfy a concept of the form $\forall R.C$ and other objects, and then to determine which of these objects must satisfy C [6]. Similar idea can be used in extension $\mathcal{RIQ}$ with a RIA of the form $w \dot{\sqsubseteq} Q \circ P$. If an object x should satisfies concept $\forall Q.C$ then we should define structure that will remember path $w \circ P^-$ from the object x to objects that must satisfy concept C. If we extend $\mathcal{RIQ}$ DL with *Fun* [11], then the next lemma holds:

Lemma 1. *Let C_0 be $\mathcal{RIQ}$ concepts and $\mathcal{R}$ regular RBox with a RIA of the form $w\dot{\sqsubseteq}QP$, where $Fun(P^-)$ holds. Let U be a new role name. We define*

$$C_1 := \forall U.(\forall w.(\exists P^-.\top)) \sqcap \forall w.(\exists P^-.\top),$$

and set

$$\mathcal{R}_1 := \mathcal{R}\backslash\{w\dot{\sqsubseteq}QP\} \cup \{UU\dot{\sqsubseteq}U, U^-\dot{\sqsubseteq}U\} \cup \{R\dot{\sqsubseteq}U | R \in \mathcal{R}_{C_0}\} \cup \{wP^-\dot{\sqsubseteq}Q\}.$$

Then, $\mathcal{RIQ}$ concept C_0 is satisfiable w.r.t. RBox $\mathcal{R}$ iff concept $C_0 \sqcap C_1$ is satisfiable w.r.t. Rbox $\mathcal{R}_1$.

Proof. The proof is based on transformation from one interpretation to another one. □

Without restriction $Fun(P^-)$, lemma (1) do not holds. It is illustrated in example (1).

Example 1. The UML[4] model of a car, shown on Figure (1a), describes Car with following parts: $Engine$, $Wheel$, $Crankshaft$ and Hub. Role name $powers$ is part-part relation [13, 2], but role names $engineInCar$, $wheelInCar$, $hubInWheel$ and $crankshaftInEngine$ are part-of relations [2]. The model corresponds to next RIA of the form [13]:

$$engineInCar \circ crankshaftInEngine \circ powers \dot{\sqsubseteq} wheelInCar \circ hubInWheel \quad (2)$$

Let $\mathcal{I}$ be an interpretation, shown on Figure (1b), of the RIA of the form (2). The interpretation $\mathcal{I}$ satisfies RIA of the form $w\dot{\sqsubseteq}wheelInCar \circ hubInWheel$, but it does not satisfy RIA of the form $w \circ hubInWheel^-\dot{\sqsubseteq}wheelInCar$, where $w = engineInCar \circ crankshaftInEngine \circ powers$. □

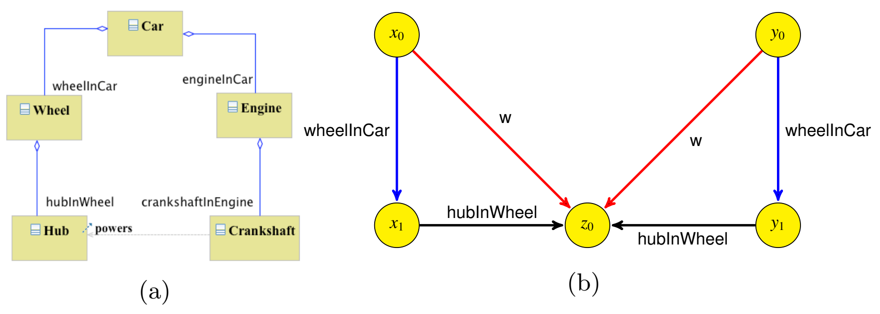


Fig. 1. (a) An UML product model (updated from [13]). (b) An interpretation $\mathcal{I}$ of RIA of the form (2).

[4] The Unified Modeling Language (http://www.uml.org/)

According to the Figure (1b), one can conclude that the restriction problems for reduction of RIAs is caused by the role $hubInWheel^-$. The role is not "unambiguously" determined. On the other side, by using the interpretation shown on the Figure (1b), it is obvious $\langle z_0, y_1\rangle \in (hubInWheel^-)^{\mathcal{I}}$ corresponds to object y_0, while $\langle z_0, x_1\rangle \in (hubInWheel^-)^{\mathcal{I}}$ corresponds to object x_0. In the other words, the condition of existence *unambiguously* role is connected to an object. To stressed which particular object corresponds to the role name, a new role (R, x) is defined. The role satisfies

$$(R, x)\dot{\sqsubseteq}R. \tag{3}$$

For example, in case of the interpretation, shown on Figure (1b), one can define new roles, as follows: $(hubInWheel^-, x_0)$, $(hubInWheel^-, y_0)$, which satisfy $\langle z_0, x_1\rangle \in (hubInWheel^-, x_0)^{\mathcal{I}}$, $\langle z_0, y_1\rangle \in (hubInWheel^-, y_0)^{\mathcal{I}}$, but $\langle z_0, x_1\rangle \notin (hubInWheel^-, y_0)^{\mathcal{I}}$. Now, one can define new tableau concept (constructor) denoted as $\overset{\exists}{\underset{\forall}{=}} (hubInWheel^-, x).D$. This constructor is used in the label of nodes of the tableau (see definition 3). Intuitively, the constructor serves to write the set of sub-concepts of the concept C_0 which have to hold in some node, i.e. if $Z = \{D|\overset{\exists}{\underset{\forall}{=}}(hubInWheel^-, x_0).D \in \mathcal{L}(z_0)\} = \{D|\forall wheelInCar.D \in \mathcal{L}(x_0)\} \neq \emptyset$ then there exists x_1 such that $\langle z_0, x_1\rangle \in \mathcal{E}((hubInWheel^-, x_0))$ and $Z \subseteq \mathcal{L}(x_1)$.

4 The extension of $\mathcal{RIQ}$ tableau

This section examines how to extend tableau for the $\mathcal{RIQ}$ DL with the new constructor. We denote, as defined in [6], $\mathcal{B}_R$ as NFA that corresponds to role R. We use a special automaton for word w, denoted with $\mathcal{B}_w$. For $\mathcal{B}$ an NFA and q a state of $\mathcal{B}$, B^q denotes the NFA obtained from $\mathcal{B}$ by making q the (only) initial state of $\mathcal{B}$ [5]. The language recognized by NFA $\mathcal{B}$ is denoted by $\mathcal{L}(\mathcal{B})$. The $clos(C_0)$ is the smallest set of concepts in negation normal form (NNF) which contains C_0, that is closed under $\dot{\neg}$ and sub-concepts [6]. For a set S the $fclos(C_0, \mathcal{R})$ and $efc(C_0, \mathcal{R}, S)$ can be defined as follows:

$fclos(C_0, \mathcal{R}) = clos(C_0) \cup \{\forall\mathcal{B}_R^q.D|\forall R.D \in clos(C_0)$ and q is a state in $\mathcal{B}_R\}$,

$efc(C_0, \mathcal{R}, S) = fclos(C_0, \mathcal{R}) \cup \{\forall\mathcal{B}_w^q.\overset{\exists}{\underset{\forall}{=}}(P^-, s).D|s \in S, \forall Q.D \in clos(C_0)\} \cup \{\overset{\exists}{\underset{\forall}{=}}(P^-, s).D|s \in S, \forall Q.D \in clos(C_0)\}$.

Let's denote

$$PL(\mathcal{B}_w) = \left\{\langle w', q\rangle\, |q \text{ is a state in } \mathcal{B}_w, (\forall w'' \in \mathcal{L}(\mathcal{B}_w^q))\big(w'w'' \in \mathcal{L}(\mathcal{B}_w)\big)\right\}.$$

Definition 3. *T=(S, $\mathcal{L}$, $\mathcal{E}$) is tableau for concept C_0 w.r.t. $\mathcal{R}$ iff*[5]

- *S is non-empty set,*
- $\mathcal{L} : S \to 2^{efc(C_0,\mathcal{R},S)}$,
- $\mathcal{E} : \mathcal{R}_{C_0} \cup \{(P^-, s)|s \in S\} \to 2^{S\times S}$
- $C_0 \in \mathcal{L}(s)$ *for some* $s \in S$

[5] w, Q and P refer to the RIA axiom of the form $w\dot{\sqsubseteq}Q \circ P$.

Next, $s, t \in S; C, C_1, C_2 \in fclos(C_0,R); R, S \in \mathcal{R}_{C_0}$ and $S^T(s,C)$ [5] satisfies rules (P1a), (P1b), (P2), (P3), (P4a), (P4b), (P5), (P6), (P7), (P8), (P9), (P10), (P13) defined in [5], and satisfies new rules:

- *(P6b) If $\forall Q.C \in \mathcal{L}(s)$, then $\forall \mathcal{B}_w.\overset{\exists}{\forall}(P^-, s).C \in \mathcal{L}(s)$.*
- *(P15a) $\forall Q.\top \in \mathcal{L}(s)$ for all $s \in S$.*
- *(P15b) If $\overset{\exists}{\forall}(P^-, s).C_1 \in \mathcal{L}(v)$, then there exists t with $\langle v, t\rangle \in \mathcal{E}(P^-, s)$ and $C_1 \in \mathcal{L}(t)$. Also, for all $C_2 \in fclos(C_0)$, if $\overset{\exists}{\forall}(P^-, s).C_2$ $\in \mathcal{L}(v)$ then $C_2 \in \mathcal{L}(t)$.*
- *(P15c) If $\langle v, t\rangle \in \mathcal{E}(P^-, s)$, then $\langle v, t\rangle \in \mathcal{E}(P^-)$.* □

Theorem 1. *Concept C_0 is satisfiable w.r.t. $\mathcal{R}$ iff there exists tableau for C_0 w.r.t. $\mathcal{R}$.*

Proof. For the if direction let $T = (S, \mathcal{L}, \mathcal{E})$ be a tableau for C_0 w.r.t $\mathcal{R}$. We define interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, with: $\Delta^{\mathcal{I}} = S$, $C^{\mathcal{I}} = \{s | C \in \mathcal{L}(s)\}$, for concept names C in $clos(C_0)$, and for roles names $R \neq Q$ and Q, we set
$R^{\mathcal{I}} = \{\langle s_0, s_n\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} |$ there are $s_1, \ldots, s_{n-1}$ with $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(S_{i+1})$ for $0 \leq i \leq n-1$ and $S_1 \cdots S_n \in \mathcal{L}(\mathcal{B}_R)\}$,
$Q^{\mathcal{I}} = \{\langle s_0, s_n\rangle \mid$ there are $s_1, \ldots, s_{n-1}$ with $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(S_{i+1})$ for $0 \leq i \leq n-1$ and $S_1 \cdots S_n \in \mathcal{L}(\mathcal{B}_Q)\} \cup \{\langle x, y\rangle | (\exists z)(\langle x, z\rangle \in w^{\mathcal{I}}$ and $\langle z, y\rangle \in \mathcal{E}((P^-, x)))\}$.

Let's prove that $\mathcal{I}$ is model for $\mathcal{C}_0$ and $\mathcal{R}$.

<u>$\mathcal{I}$ is model for $\mathcal{R}$</u>. Let's consider RIA of the form $w \dot{\sqsubseteq} Q \circ P$. If $\langle x, y\rangle \in w^{\mathcal{I}}$. According to (P15a) and (P6b) then $\forall \mathcal{B}_w.\overset{\exists}{\forall}(P^-, x).\top \in \mathcal{L}(x)$ holds. According to (P4a), (P15b), (P15c) and definition of $Q^{\mathcal{I}}, P^{\mathcal{I}}$ we have $(\exists t)\, \langle y, t\rangle \in \mathcal{E}((P^-, x)), \langle y, t\rangle \in \mathcal{E}(P^-)$ and $\langle x, t\rangle \in Q^{\mathcal{I}}$, and finally implies $\langle x, y\rangle \in (Q \circ P)^{\mathcal{I}}$.
<u>$\mathcal{I}$ is model for C_0</u>. It is enough to prove that $C \in \mathcal{L}(s)$ implies $s \in C^{\mathcal{I}}$ for all $s \in S$ and $C \in clos(C_0)$. Let's consider $C \equiv \forall Q.D$. For other cases the proof is the same as proof in [6].

Let $\forall Q.D \in \mathcal{L}(s)$ and $(s, t) \in Q^{\mathcal{I}}$. If $\exists S_1 \cdots S_{n-1} \in \mathcal{L}(\mathcal{B}_Q)$, so $\langle s_i, s_{i+1}\rangle \in \mathcal{E}(S_{i+1}), i = 0, \ldots, n-1, s_0 = s, s_n = t$, then the proof is the same as proof in [6]. In case of $(\exists z)\, \langle s, z\rangle \in w^{\mathcal{I}}$ and $\langle z, t\rangle \in \mathcal{E}(P^-, s)$. Based on the definition of $\omega^{\mathcal{I}}$ and (P6b) we have $\overset{\exists}{\forall}(P^-, s).D \in \mathcal{L}(z)$. According to (P15b), we have $D \in \mathcal{L}(t)$. By induction $t \in D^{\mathcal{I}}$, so we have $s \in (\forall Q.D)^{\mathcal{I}}$.
For the converse, suppose that $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is model for C_0 w.r.t. $\mathcal{R}$. Let's define tableau $T = (S, \mathcal{L}, \mathcal{E})$, as follows: $S = \Delta^{\mathcal{I}}$, $\mathcal{E}(R) = R^{\mathcal{I}}$, $\mathcal{E}((P^-, x)) = \{\langle y, z\rangle \in S^2 | \, \langle x, y\rangle \in w^{\mathcal{I}}, \langle x, z\rangle \in Q^{\mathcal{I}}, \langle z, y\rangle \in P^{\mathcal{I}}\}$, and

$\mathcal{L}(s) = \{C \in clos(C_0) | s \in C^{\mathcal{I}}\} \cup \{\forall \mathcal{B}_R.C | \forall R.C \in clos(C_0)$ and $s \in (\forall R.C)^{\mathcal{I}}\} \cup \{\forall \mathcal{B}_R^q.C \in fclos(C_0, \mathcal{R})|$ for all $S_1 \cdots S_n \in \mathcal{L}(\mathcal{B}_R^q), s \in (\forall S_1.$ $\forall S_2 \cdots \forall S_n.C)^{\mathcal{I}}$, and if $\varepsilon \in \mathcal{L}(\mathcal{B}_R^q)$ then $s \in C^{\mathcal{I}}\} \cup \{\forall\, Q.\top\} \cup \{\, \forall \mathcal{B}_w.\overset{\exists}{\forall}(P^-$ $, s).C | s \in (\forall Q.C)^{\mathcal{I}}\} \cup \{\, \forall \mathcal{B}_w^q.\overset{\exists}{\forall}(P^-, t).C | (\exists w')\, \langle w', q\rangle \in PL(\mathcal{B}_w)$ and $\langle t, s\rangle \in (w')^{\mathcal{I}}$ and $t \in (\forall Q.C)^{\mathcal{I}}\} \cup \{\overset{\exists}{\forall}(P^-, t).C | \, \langle t, s\rangle \in w^{\mathcal{I}}$ and $t \in (\forall Q.C)^{\mathcal{I}}\}$.

Let's prove that T is tableau for C_0 w.r.t. $\mathcal{R}$. We consider only new rules (see definition 3). From definition $\mathcal{E}(P^-, x)$ and $\mathcal{E}(P)$ we prove (P15c). From the definition of $\mathcal{L}(s)$ we have that (P15a) and (P6b) holds. Let's prove rule (P15b). Suppose that $\overset{\exists}{\forall}(P^-, s).C_1 \in \mathcal{L}(v)$. From the definition of $\mathcal{L}(v)$ follows

$\langle s,v\rangle \in w^{\mathcal{I}}$ and $s \in (\forall Q.C)^{\mathcal{I}}$. Because of $\mathcal{I} \models w \dot{\sqsubseteq} Q \circ P$ we have that there exists z such that $\langle s,z\rangle \in Q^{\mathcal{I}}$ and $\langle z,v\rangle \in P^{\mathcal{I}}$ i.e. $\langle v,z\rangle \in \mathcal{E}((P^-,s))$. On the other hand, from $s \in (\forall Q.C)^{\mathcal{I}}$ and $\langle s,z\rangle \in Q^{\mathcal{I}}$ follows $z \in C_1^{\mathcal{I}}$, so $C_1 \in \mathcal{L}(z)$. If $\frac{\exists}{\forall}(P^-,s).C_2 \in \mathcal{L}(v)$ then $s \in (\forall Q.C_2)^{\mathcal{I}}$, so $z \in C_2^{\mathcal{I}}$, i.e. $C_2 \in \mathcal{L}(z)$. □

5 Tableau algorithm

The tableau algorithm generates completion tree.

Definition 4. *(Completion tree) Completion tree for C_0 w.r.t $\mathcal{R}$ is labelled tree $G = (V,E,\mathcal{L},\dot{\neq})$ where each node $x \in V$ is labelled with a set $\mathcal{L}(x) \subseteq efc(C_0,\mathcal{R},V) \cup \{\leq\ mR.C| \leq nR.C \in clos(C_0), m \leq n\}$. Each edge $\langle x,y\rangle \in E$ is labelled with a set $\mathcal{L}\langle x,y\rangle \subseteq \mathcal{R}_{C_0} \cup \{(P^-,s)|s \in V\}$. Additionally, we care of inequalities between nodes in V, of the tree G, with a symmetric binary relation $\dot{\neq}$.*
If $\langle x,y\rangle \in E$, then y is called successor of the x, but x is called predecessor of y. Ancestor is the transitive closure of predecessor, and descendant is the transitive closure of successor. A node y is called an R-successor of a node x if, for some R' with $R' \sqsubseteq_* R$, $R' \in \mathcal{L}(\langle x,y\rangle)$. A node y is called a neighbour (R-neighbour) of a node x if y is a successor (R-successor) of x or if x is a successor ($Inv(R)$-successor) of y. For $S \in \mathcal{R}_{C_0}$, $x \in V$, $C \in clos(C_0)$ we define set $S^G(x,C) = \{y|y \text{ is } S-neighbour \text{ of } x \text{ and } C \in \mathcal{L}(y)\}$ □

Definition 5. *A tree G is said to contain a clash if there is a node x such that:*

- *$\perp \in \mathcal{L}(x)$, or*
- *for a concept name A, $\{A,\neg A\} \subseteq \mathcal{L}(x)$, or*
- *there exists a concept $(\leq nS.C) \in \mathcal{L}(x)$ and $\{y_0,\ldots,y_n\} \in S^G(x,C)$ with $y_i \dot{\neq} y_j$ for all $0 \leq i < j \leq n$.* □

In order to provide termination of the algorithm, in [6] blocking techniques are used, and fact that the set of nodes' labels is finite. In our tableau definition (3), if S is infinite set then $efc(C_0,\mathcal{R},S)$ is also infinite. So, number of different $\mathcal{L}(s)$ is infinite. Also, sets $\mathcal{L}(s)$ can be infinite. To ensure that sets $\mathcal{L}(s)$ are finite, we define additional restriction on the set of RIA of the form $w \dot{\sqsubseteq} Q \circ P$. Let's suppose that language $\mathcal{L}(\mathcal{B}_w)$ is finite.

If $\forall \mathcal{B}_w^q.\frac{\exists}{\forall}(P^-,s).C \in \mathcal{L}(t)$ then there exists $(w',q) \in PL(\mathcal{B}_w)$ and $(s,t) \in \mathcal{E}(w')$. If $n = \sharp fclos(C_0,\mathcal{R})$, $l = max\{len(w')|(\exists q)(w',q) \in PL(\mathcal{B}_w)\}$ and number of *successors* is less than m (different than $P-neighbours$), then [6]: $\sharp\mathcal{L}(t) \leq n\cdot m^l \cdot \sharp PL(\mathcal{B}_w)$. To illustrate the technique in an understandable way, we consider only special case, when $\mathcal{L}(\mathcal{B}_w) = \{R\}$.

Definition 6. *Let $G = (V,E,\mathcal{L},\dot{\neq})$ be completion tree and $f : V \to V$ is a function.*

1. *We say that $\mathcal{L}(x)$ $f-match$ with $\mathcal{L}(y)$, denoted as $\mathcal{L}(x) \sim^f \mathcal{L}(y)$, if*

[6] Because of $\mathcal{L}(\mathcal{B}_w)$ is finite, then l, $\sharp PL(\mathcal{B}_w)$ are also finite.

   - $f(x) = y$,
   - $\mathcal{L}(x) \cap fclos(C_0, \mathcal{R}) = \mathcal{L}(y) \cap fclos(C_0, \mathcal{R})$,
   - $R \in \mathcal{L}(\langle z, x\rangle) \Leftrightarrow R \in \mathcal{L}(\langle f(z), y\rangle)$,
   - $\overset{\exists}{\forall}(P^-, z).C \in \mathcal{L}(x) \Leftrightarrow \overset{\exists}{\forall}(P^-, f(z)).C \in \mathcal{L}(y)$.
2. *We say that* $\mathcal{L}(\langle x, y\rangle)$ $f-$*match with* $\mathcal{L}(\langle u, v\rangle)$*, denoted with* $\mathcal{L}(\langle x, y\rangle) \sim^f \mathcal{L}(\langle u, v\rangle)$*, if*
   - $\mathcal{L}(\langle x, y\rangle) \cap \mathcal{R}_{C_0} = \mathcal{L}(\langle u, v\rangle) \cap \mathcal{R}_{C_0}$,
   - $(\forall s \in V)((P^-, s) \in \mathcal{L}(\langle x, y\rangle) \Leftrightarrow (P^-, f(s)) \in \mathcal{L}(\langle u, v\rangle))$. □

Definition 7. *(Blocking) A node x is label blocked if there is a function $f : V \to V$ and there are predecessors x', y, y' of the node x, such that*

- $x' \neq y$,
- *x is successor of x' and y is successor of y'*,
- $\mathcal{L}(x) \sim^f \mathcal{L}(y)$, $\mathcal{L}(x') \sim^f \mathcal{L}(y')$,
- $\mathcal{L}(\langle x, x'\rangle) \sim^f \mathcal{L}(\langle y, y'\rangle)$.

In this case we say that y blocks x. □

A node is blocked if it is label blocked or its predecessor is blocked. If the predecessor of a node x is blocked, then we say that x is indirectly blocked [5]. There is an algorithm that checks whether a node y blocks node x. It is enough to consider nodes x, y and their predecessors x' and y' and (finite number of) R-neighbours of these four nodes. For the nodes, function f can be nondeterministically defined and check the rules in the definition (7). It is also possible to check the rules algorithmically, because the rules use only finite sets.

The non-deterministic tableau algorithm can be described as follows:

- Input: Concept C_0 and RBox $\mathcal{R}$,
- Output: "Yes" if concept C_0 is satisfiable w.r.t. RBox $\mathcal{R}$, otherwise "No"
- Method:
  1. step: Construct tree $G = (V, E, \mathcal{L}, \dot{\neq})$, where $V = \{x_0\}$, $E = \emptyset$, $\mathcal{L}(x_0) = \{C_0\}$. Go to step 2.
  2. step: Apply an expansion rule (see table 1) to the tree G, while it is possible. Otherwise, go to step 3.
  3. step: If the tree does not contain *clash* return "Yes", otherwise return "No".

Theorem 2. *1. Tableau algorithm terminates when started with C_0 and $\mathcal{R}$,*
2. Tableau algorithm returns answer "Yes" iff there exists tableau of the concept C_0 w.r.t $\mathcal{R}$.

Proof. (a) $\exists$-rule and $\geq$-rule generate finite number of successors of node x. So, the set $\mathcal{L}(x)$ is finite and the number of (P^-, y)-successors of node x is finite. There is limited number the possible labels of pairs $(x', x) \in E$ that will lead the blocking of tree nodes. It means, the tree generated by the algorithm is finite. According to [6], the rule which generates node y and remove rule $\leq$, will not be applied, again. This means that the algorithm can applied only finite number of expansion rules.

$\sqcap$-rule:	If then	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$, $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:	If then	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{E\}$, for some $E \in \{C_1, C_2\}$
$\exists$-rule:	If then	$\exists S.C \in \mathcal{L}(x)$, x is not blocked, and x has no S-negihbour y where $C \in \mathcal{L}(y)$ create new node y where $\mathcal{L}(\langle x, y\rangle) := \{S\}$ and $\mathcal{L}(y) := \{C\}$
$\forall_1$-rule:	If then	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and $\forall \mathcal{B}_S.C \notin \mathcal{L}(x)$ $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\forall \mathcal{B}_S.C\}$
$\forall_2$-rule:	If then	$\forall \mathcal{B}^p.C \in \mathcal{L}(x)$, x is not indirectly blocked, $p \overset{S}{\rightarrow} q \in \mathcal{B}^p$ and there is an S-neighbour y of x with $\forall \mathcal{B}^q.C \notin \mathcal{L}(y)$ $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\forall \mathcal{B}^q.C\}$
$\forall_3$-rule:	If then	$\forall \mathcal{B}.C \in \mathcal{L}(x)$, x is not indirectly blocked, $\varepsilon \in \mathcal{L}(\mathcal{B})$ and $C \notin \mathcal{L}(x)$ $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C\}$
$\forall_4$-rule:	If then	$\forall Q.C \in \mathcal{L}(x)$, x is not indirectly blocked, and $\forall \mathcal{B}_w.\overset{\exists}{\forall}(P^-, x).C \notin \mathcal{L}(x)$ $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\forall \mathcal{B}_w.\overset{\exists}{\forall}(P^-, x).C\}$
$\forall_5$-rule:	If then	$\forall Q.\top \notin \mathcal{L}(x)$, x is not indirectly blocked $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\forall Q.\top\}$
$(\overset{\exists}{\forall})_1$-rule:	If then	$\overset{\exists}{\forall}(P^-, x).C \in \mathcal{L}(y)$, y is not blocked and there is no z with $(P^-, x) \in \mathcal{L}(\langle y, z\rangle)$ create new node z with $(P^-, x) \in \mathcal{L}(\langle y, z\rangle), P^- \in \mathcal{L}(\langle y, z\rangle)$
$(\overset{\exists}{\forall})_2$-rule:	If then	$\overset{\exists}{\forall}(P^-, x).C \in \mathcal{L}(y)$, y is not blocked, there is z with $(P^-, x) \in \mathcal{L}(\langle y, z\rangle)$ and $C \notin \mathcal{L}(z)$ $\mathcal{L}(z) \rightarrow \mathcal{L}(z) \cup \{C\}$
choose-rule:	If then	$(\leq nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an S-neighbour y of x $\{C, \dot{\neg} C\} \cap \mathcal{L}(y) = \emptyset$ $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{E\}$, for some $E \in \{C, \dot{\neg} C\}$
$\geq$-rule:	If then 	(1) $(\geq nS.C) \in \mathcal{L}(x)$, x is not blocked, and (2) there are not n S-neighbours $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \dot{\neq} y_j$ for $1 \leq i < j \leq n$, create n new nodes $y_1, \ldots y_n$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$ and $y_i \dot{\neq} y_j$ for $1 \leq i < j \leq n$
$\leq$-rule:	If then 	(1) $(\leq nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked (2) $\sharp S^{\mathbf{G}}(x, C) > n$ and there are $y, z \in S^{\mathbf{G}}(x, C)$ with not $y \dot{\neq} z$ and y is not root node nor an ancestor of z (1) $\mathcal{L}(z) \rightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ (2) if z is an ancestor of x, then $\mathcal{L}(\langle z, x\rangle) \rightarrow \mathcal{L}(\langle z, x\rangle) \cup Inv(\mathcal{L}(\langle x, y\rangle))$ else $\mathcal{L}(\langle x, z\rangle) \rightarrow \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle x, y\rangle)$ (3) set $u \dot{\neq} z$, for all u with $u \dot{\neq} y$ (4) remove y and sub-tree below y from G

Table 1. Expansion rules for a tableau algorithm (updated from [5])

(b) For the if direction, suppose that the algorithm returns "Yes". It means that the algorithm generated tree $G = (V, E, \mathcal{L}, \dot{\neq})$ without clash and there are no expansion rules (see table 1) that can be applied.
A path [5, 8] is a sequence of pairs of nodes of G of the form

$$p = \langle (x_0, x'_0), \ldots, (x_n, x'_n) \rangle \, . \tag{4}$$

For such a path, we define $Tail(p) = x_n$ and $Tail'(p) = x'_n$. We denote the path

$$\big\langle (x_0, x'_0), (x_1, x'_1), \ldots, (x_n, x'_n), (x_{n+1}, x'_{n+1}) \big\rangle \, . \tag{5}$$

with $\big\langle p, (x_{n+1}, x'_{n+1}) \big\rangle$. If node x is label blocked then corresponding function f is denoted with f_x. The node x is blocked with node $f_x(x)$. We use function: $G(x, z) = z$, if x is not blocked, or $G(x, z) = f_x(z)$, if x is blocked.
The set $Paths(G)$ is defined inductively, as follows:
– If $x_0 \in V$ is the root of tree then $\langle x_0, x_0 \rangle \in Paths(G)$,
– If $p \in Paths(G)$ and $z \in V$and z is not indirectly blocked, such that $\langle Tail(p), z \rangle \in E$, then $\langle p, (G(z, z), z) \rangle \in Paths(G)$.

Let's define structure $\mathcal{T} = \{\mathcal{S}, \mathcal{L}', \mathcal{E}\}$ as follows:
$\mathcal{S} = Paths(G)$,
$\mathcal{E}(S) = \{\langle p, q \rangle \in Paths(G) \times Paths(G) | q = \langle p, (G(z, z), z) \rangle$ and $S \in \mathcal{L}(\langle Tail(p), z \rangle)$ or $p = \langle q, (G(z, z), z) \rangle$ and $Inv(S) \in \mathcal{L}(\langle Tail(q), z \rangle)\}$, for $S \in \mathcal{R}_{C_0}$,
$\mathcal{E}(P^-, r) = \{\langle p, q \rangle \in \mathcal{E}(P^-) | \, \langle r, p \rangle \in \mathcal{E}(R)$ and $(P^-, G(Tail'(p), Tail'(r)))$ $\in \mathcal{L}(G(Tail'(p), Tail'(p)), Tail'(q))\}$,
$\mathcal{L}'(p) = \mathcal{L}(Tail(p)) \cap fclos(C_0, \mathcal{R}) \cup \{\forall \, R. \frac{\exists}{\forall}(P^-, p).C | \forall Q.C \in \mathcal{L}(Tail(p))\} \cup$ $\{\frac{\exists}{\forall}(P^-, r).C | \, \langle r, p \rangle \in \mathcal{E}(R)$ and $\frac{\exists}{\forall}(P^-, Tail'(r)).C \in \mathcal{L}(Tail'(p))\}$.

Let's prove that $\mathcal{T}$ is tableau for C_0 w.r.t R. We consider only (P15b) property, and avoid already defined properties in [6]. New properties (P6b), (P15a), (P15c) imply from $\forall_4$, $\forall_5$ and $(\frac{\exists}{\forall})_1$.

Suppose $\frac{\exists}{\forall}(P^-, r).C \in \mathcal{L}'(p)$ then $\langle r, p \rangle \in \mathcal{E}(R)$ and $\frac{\exists}{\forall}(P^-, Tail'(r)).C \in \mathcal{L}(Tail'(p))$. Because of $\langle r, p \rangle \in \mathcal{E}(R)$, four cases are possible:

1. $p = \langle r, (G(z, z), z) \rangle$ and $G(z, z) = z$
2. $p = \langle r, (G(z, z), z) \rangle$ and $G(z, z) \neq z$
3. $r = \langle p, (G(z, z), z) \rangle$ and $G(z, z) = z$
4. $r = \langle p, (G(z, z), z) \rangle$ and $G(z, z) \neq z$

The subcases above are analyzing on the similar way and we consider the most complex of them i.e. case (2). The $Tail'(r)$ is not blocked, so $Tail'(r) = Tail(r)$, while z is blocked by $G(z, z)$. From $\langle r, p \rangle \in \mathcal{E}(R)$ blocking definition we have $R \in \mathcal{L}(\langle Tail(r), z \rangle)$ and $R \in \mathcal{L}(G(z, Tail(r)), G(z, z))$, while, from $\frac{\exists}{\forall}(P^-, Tail'(r)).C \in \mathcal{L}(z)$ we have $\frac{\exists}{\forall}(P^-, G(z, \, Tail(r)).C \in \mathcal{L}(G(z, z))$. According to the rule $(\frac{\exists}{\forall})_1$, we have that there exists node y such that
$P^-, (P^-, G(z, Tail(r)) \in \mathcal{L}(\langle G(z, z), y \rangle)$. Let $q = \langle p, (G(y, y), y) \rangle$ then $\langle p, q \rangle \in \mathcal{E}(P^-)$ and $(P^-, G(Tail'(p), Tail'(r))) \in \mathcal{L}(G(Tail'(p), \, Tail'(p)), Tail'(q))$, so $\langle p, q \rangle \in \mathcal{E}(P^-, r)$. Having regard to the rule $(\frac{\exists}{\forall})_2$ we conclude that property (P15b) holds.

For the only-if direction, the proof is the same as proof in [6] (i.e., we take a tableau and use it to steer the application of the non-deterministic rules). □

6 Conclusions and future works

This paper shortly examines how to handle complex RIAs with more than one role from the right hand side of the composition of roles in $\mathcal{RIQ}$ DL. Although the proof was conducted for RIA of the form $R\dot{\sqsubseteq}Q\circ P$, we can apply the technique to RIA of the form $S_1S_2\cdots S_n\dot{\sqsubseteq}R_1R_2\cdots R_m$, with restriction that corresponding languages are finite. Our future work will be focused on the problem which conditions should satisfy role if we have more than one RIAs, to be mention technique could be applied. Also, we will do research on RIA of the form $w\dot{\sqsubseteq}QP$ when the language $\mathcal{L}(\mathcal{B}_w)$ is infinite.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook - Theory, Implementation and Application.* Cambridge University Press, second edition, 2007.
2. C. Bock. UML 2 Composition Model. *Journal of Object Technology*, 3(10):47-73, 2004.
3. C. Bock, X. F. Zha, H-W. Suh, J-H. Lee, Ontological Product Modeling for Collaborative Design, U.S. National Institute of Standards and Technology, Technical Report NISTIR 7643, 2009.
4. B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. P. Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web.* 6(4): 309-322, 2008.
5. I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible $\mathcal{SROIQ}$. *In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006).* pages 57-67, 2006.
6. I. Horrocks, U. Sattler. Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. *Artificial Intelligence*, 160(1-2):79-104, 2004.
7. I. Horrocks, U. Sattler. A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning.* Springer Verlag, 39(3): 245-429, 2007.
8. I. Horrocks, and U. Sattler. Optimised Reasoning for SHIQ. *In Proceedings of the 15th European Conf. on Artificial Intelligence (ECAI 2002)*, pages 277-281, 2002.
9. I. Horrocks, O. Kutz, and U. Sattler. The irresistible $\mathcal{SRIQ}$. *In Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions*, 2005.
10. I. Horrocks, , P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics.* 1(1):7-26, 2003.
11. Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$. *In Proceedings of Description Logics Workshop (DL 2008), CEUR-Workshop.* Vol. 353, 2008.
12. Y. Kazakov. An Extension of Complex Role Inclusion Axioms in the Description Logic $\mathcal{SROIQ}$. *In Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR 2010).* pages 472-487, Edinburgh, UK, 2010.
13. N. Krdžavac, C. Bock. Reasoning in Manufacturing Part-Part Examples with OWL 2. U.S. National Institute of Standards and Technology, Technical Report NISTIR 7535, 2008.
14. B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Articial Intelligence Research.* 36: 165-228, 2009.

Bidirectional reachability-based modules

Riku Nortje[1,2], Katarina Britz[1,2], and Thomas Meyer[1,3]

[1]CSIR Meraka Institute, Pretoria, South Africa
[2]University of South Africa, Pretoria, South Africa
[3]University of KwaZulu-Natal, Durban, South Africa
Email: nortjeriku@gmail.com; {arina.britz;tommie.meyer}@meraka.org.za

Abstract. We introduce an algorithm for MinA extraction in $\mathcal{EL}$ based on bidirectional reachability. We obtain a significant reduction in the size of modules extracted at almost no additional cost to that of extracting standard reachability-based modules. Bidirectional modules are related to nested locality modules, but are aimed specifically at MinA extraction and are generally smaller. For acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept inclusions, all MinAs can be extracted without the need for subsumption testing.

1 Introduction

Module extraction plays an important role in the design, reuse and maintenance of ontologies as well as aiding in the optimization of reasoning services [9]. When used to optimize reasoning services such as subsumption testing and MinA extraction, reachability-based modules have been criticized for only considering the subsumee of a subsumption entailment during the module extraction process [2], thus not sufficiently reducing the size of modules.

In this paper we address this shortcoming of reachability-based modules, with the aim of improving MinA extraction, as follows: We introduce a top-down heuristic which considers only the subsumer of an entailment and then combine it with standard reachability-based modules to form a bidirectional version of reachability. This new bidirectional version of the heuristic thus considers both the subsumee and subsumer in a subsumption entailment between concept names. For relatively sparse graphs this significantly reduces the size of modules extracted with almost no additional cost to that of extracting standard reachability-based modules.

Given a subsumption statement between single concept names, we show that every MinA is in fact a bidirectional reachability-based module in terms of itself. Using this property we implement very fast algorithms to extract all MinAs for acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept inclusions without performing a single subsumption test, thereby significantly reducing the runtime complexity of MinA extraction for these TBoxes.

In Section 2 we give a brief introduction to description logics and the notations as used in this paper. Section 3 introduces reachability-based modules [9], the new top-down reachability heuristic and finally defines bidirectional reachability-based modules. Then in Section 4 we investigate the relationship between MinAs and the inexpressive Horn DL $\mathcal{HL}$ and extend the findings to $\mathcal{EL}$ TBoxes consisting of primitive concept inclusions. Lastly in Section 5 we provide empirical results of the various algorithms presented as tested on three generally large real world biomedical ontologies.

2 Preliminaries

In the standard set-theoretic semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists of a non-empty set Δ^I (the *domain* of I) and a function $\cdot^I$ (the *interpretation function* of I) which maps each atomic concept A to a subset A^I of Δ^I, and each atomic role r to a subset r^I of $\Delta^I \times \Delta^I$. The interpretation function is extended to arbitrary concept and role descriptions, with the specifics depending on the particular description logic under consideration.

A DL knowledge base consists of a *TBox* which contains *terminological axioms* and an *ABox* which contains *assertions*; for the purposes of this paper we concern ourselves only with Tbox statements, or *general concept inclusions* (GCIs) of the form $C \sqsubseteq D$, where C and D are (possibly complex) concept descriptions. Here C is referred to as the *subsumee* and D as the *subsumer*. An interpretation I *satisfies* $C \sqsubseteq D$, written $I \Vdash C \sqsubseteq D$, iff $C^I \subseteq D^I$. In this paper, when the left hand side of a GCI consists of only a single concept name, the statement is referred to as a primitive concept inclusion.

An interpretation I satisfies a DL TBox $\mathcal{T}$ iff it satisfies every statement in $\mathcal{T}$. A TBox $\mathcal{T}$ *entails* a DL statement ϕ, written as $\mathcal{T} \models \phi$, iff every interpretation that satisfies $\mathcal{T}$ also satisfies ϕ.

Roughly speaking, DLs are defined by the constructors they provide. In this paper we consider the DLs $\mathcal{HL}$ and $\mathcal{EL}$. The constructors allowed for $\mathcal{EL}$ are conjunction ($\sqcap$) and existential restriction ($\exists$), with semantics defined as follows: $(C \sqcap D)^I = C^I \cap D^I$; $(\exists r.C)^I = \{x \in \Delta^I \mid \exists y \in \Delta^I : (x, y) \in r^I \wedge y \in C^I\}$. The only concept constructor allowed for $\mathcal{HL}$ is conjunction, with semantics as for $\mathcal{EL}$. Both $\mathcal{HL}$ and $\mathcal{EL}$ also have the distinguished top concept $\top$ with semantics $\top^I = \Delta^I$. Normalization for $\mathcal{HL}$ only allows GCIs of the form $A \sqsubseteq B$ and $A_1 \sqcap A_2 \sqsubseteq B$. For $\mathcal{EL}$, GCIs of the form $A \sqsubseteq \exists r.B$ and $\exists r.A \sqsubseteq B$ are also allowed. Given any concept description or subsumption statement α, Sig(α) is defined as the union of all concept and role names occurring in α.

Definition 1. (Module) *Let $\mathcal{L}$ be an arbitrary description language, $\mathcal{O}$ an $\mathcal{L}$ ontology, and σ a statement formulated in $\mathcal{L}$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a* module for σ in $\mathcal{O}$ *(a* σ-module in $\mathcal{O}$*) whenever: $\mathcal{O} \models \sigma$ if and only if $\mathcal{O}' \models \sigma$. We say that $\mathcal{O}'$ is a module for a signature $\boldsymbol{S}$ in $\mathcal{O}$ (an* $\mathbf{S}$-module in $\mathcal{O}$*) if, for every $\mathcal{L}$ statement σ with Sig(σ) $\subseteq \boldsymbol{S}$, $\mathcal{O}'$ is a σ-module in $\mathcal{O}$. Given the statement σ, if there is no $\mathcal{O}'' \subset \mathcal{O}'$ such that $\mathcal{O}'' \models \sigma$ then $\mathcal{O}'$ is a minimal σ-module.*

Given a subsumption statement $\sigma = A \sqsubseteq B$, a MinA is defined as a minimal set of axioms $\mathcal{O}'$ such that $\mathcal{O}' \models A \sqsubseteq B$. Though these are not usually referred to as modules in the literature, MinAs are by definition minimal modules for a specific statement of interest.

3 Bidirectional Reachability-based Modules for $\mathcal{EL}$

Extracting modules aims to preserve both subsumption and non-subsumption relationships in a subset of an ontology. This can be understood as the reachability problem in a directed graph [1], considering concept names as nodes and explicit subsumption relationships as edges in the graph, where each inclusion axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ essentially specifies a collection of hyperedges from the connected node $\text{Sig}(\alpha_L)$ to each of the symbols in $\text{Sig}(\alpha_R)$.

Definition 2. (Bottom-up reachability-based modules [9])[1] *Let $\mathcal{O}$ be an $\mathcal{EL}$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of S-reachable names in $\mathcal{O}$ is defined inductively as follows: (i) x is S-reachable in $\mathcal{O}$, for every $x \in S$; and (ii) for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is S-reachable in $\mathcal{O}$ for every $x \in Sig(\alpha_L)$, then y is S-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_R)$. We call an axiom $\alpha_L \sqsubseteq \alpha_R$ S-reachable in $\mathcal{O}$ if every element of $Sig(\alpha_L)$ is S-reachable in $\mathcal{O}$. The bottom-up reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}_S^{reach}$, consists of all S-reachable axioms in $\mathcal{O}$.*

When S is the single concept A, we write A-reachable and $\mathcal{O}_A^{reach}$. For $\mathcal{EL}$, axioms of the form $\top \sqsubseteq \alpha_R$ are such that $\text{Sig}(\top) = \emptyset$, thus they will form part of every reachability-based module extracted. Bottom-up reachability-based modules are in fact equivalent to $\bot$-locality based modules [4, 8].

A criticism that may be raised against these bottom-up reachability-based modules is that they contain many irrelevant axioms and in some cases do not reduce the size of the ontology at all [2]. This stems from the fact that $\mathcal{O}_A^{reach}$ considers only the subsumee A in $\mathcal{O} \models A \sqsubseteq B$; the subsumer B is never used to eliminate unwanted axioms. For example:

Example 1. Given the ontology $\mathcal{O} = \{A \sqsubseteq \exists r.D, \exists r.D \sqsubseteq B, E \sqsubseteq B, A \sqsubseteq F\}$, as well as the entailment $\mathcal{O} \models A \sqsubseteq B$, $\mathcal{O}_A^{reach}$ consists of axioms $\{A \sqsubseteq \exists r.D, \exists r.D \sqsubseteq B, A \sqsubseteq F\}$. $A \sqsubseteq F$ is irrelevant in terms of $\mathcal{O} \models A \sqsubseteq B$, yet it is included in $\mathcal{O}_A^{reach}$.

For large ontologies many such irrelevant axioms may be included in a bottom-up reachability-based module. We introduce modules based on the subsumer of an entailment namely top-down reachability-based modules. Formally:

[1] The original definition by Suntisrivaraporn does not have the qualifier 'bottom-up', but because we introduce 'top-down' reachability-based modules later on in Definition 3, the qualifier is used to avoid confusion.

Definition 3. (Top-down reachability-based module) *Let $\mathcal{O}$ be an $\mathcal{EL}$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of $\overleftarrow{\mathrm{S}}$-reachable names in $\mathcal{O}$ is defined inductively as follows: (i) x is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$, for every $x \in S$; and (ii) for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ for some $x \in Sig(\alpha_R)$, then y is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_L)$. We call an axiom $\alpha_L \sqsubseteq \alpha_R$ $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ if some element of $Sig(\alpha_R)$ is $\overleftarrow{\mathrm{S}}$-reachable. The top-down reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}^{reach}_{\overleftarrow{\mathrm{S}}}$, consists of all $\overleftarrow{\mathrm{S}}$-reachable axioms from $\mathcal{O}$.*

Algorithm 1 extracts a top-down reachability based module, given an $\mathcal{EL}$ TBox $\mathcal{O}$ and a signature S as input. `active-axioms`(x) are all those, and only those axioms $(\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O}$ such that $x \in \mathrm{Sig}(\alpha_R)$, thus every such axiom is also by definition top-down reachable. For a signature S we define `active-axioms` $(S) := \bigcup_{x \in S}$`active-axioms`(x).

Algorithm 1 (Extract top-down reachability-based module)

```
Procedure extract-top-down-module($\mathcal{O}$, $S$)
Input: $\mathcal{O}$ - $\mathcal{EL}$ ontology; $S$ - signature
Output: $\mathcal{O}_S$: top-down reachability-based module for $S$ in $\mathcal{O}$
1: $\mathcal{O}_S := \emptyset$; queue := active-axioms($S$)
3: while not empty(queue) do
4:   ($\alpha_L \sqsubseteq \alpha_R$) := fetch(queue)
5:   $\mathcal{O}_S := \mathcal{O}_S \cup \{\alpha_L \sqsubseteq \alpha_R\}$
6:   queue := queue $\cup$ (active-axioms(Sig($\alpha_L$)) $\setminus \mathcal{O}_S$)
7: return $\mathcal{O}_S$
```

Theorem 1. *[5] Let $\mathcal{O}$ be an $\mathcal{EL}$ ontology, n the number of axioms in $\mathcal{O}$, and $S \subseteq Sig(\mathcal{O})$ a signature. Algorithm 1 terminates after $O(n)$ steps and returns the top-down reachability-based module for S in $\mathcal{O}$.*

It is easy to show that top-down reachability-based modules are equivalent to a subset of $\top$-locality modules [4, 8]. These modules can be criticized in a similar manner to bottom-up reachability-based modules, in that they include many irrelevant axioms. Combining $\bot$-locality modules with $\top$-locality based modules allows us to extract so called nested locality modules denoted by $\top\bot$ or $\bot\top$ [8]. We introduce a slightly different form of module called *bidirectional reachability-based modules*, aimed towards finding small modules preserving subsumption relationships between single concept names.

Definition 4. (Bidirectional reachability-based module [6]) *The bi-directional reachability-based module, denoted $\mathcal{O}^{reach}_{A \leftrightarrow B}$, for the statement $A \sqsubseteq B$ in terms of $\mathcal{O}$, is defined as the set of all axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that: for every $x_i \in Sig(\alpha_L)$, x_i is A-reachable in terms of $\mathcal{O}$, and α_R is $\overleftarrow{\mathrm{B}}$-reachable in terms of $\mathcal{O}$. Any non-empty subset $\mathcal{O}' \subseteq \mathcal{O}^{reach}_{A \leftrightarrow B}$ such that $\mathcal{O}'^{reach}_{A \leftrightarrow B} = \mathcal{O}'$ is called a bidirectional reachability-based sub-module of $\mathcal{O}$ for the statement $A \sqsubseteq B$. $\mathcal{O}'^{reach}_{A \leftrightarrow B}$ is minimal if there exists no $\mathcal{O}'' \subset \mathcal{O}'$ such that $\mathcal{O}''^{reach}_{A \leftrightarrow B} = \mathcal{O}''$.*

These modules differ from nested locality modules as follows: Given a subsumption statement $A \sqsubseteq B$, and the $\bot\top$ module $\mathcal{O}'$, then $\mathcal{O}'$ will contain all axioms for the signature $S = \{A, B\}$, thus it will include all the axioms for the entailments $\mathcal{O}' \models A \sqsubseteq B$ and $\mathcal{O}' \models B \sqsubseteq A$. A bidirectional reachability-based module $\mathcal{O}''$, however, only contains axioms for the entailment $\mathcal{O}'' \models A \sqsubseteq B$. Using the notation that $\bot\{A, B\}$ represents the $\bot$ locality module for the signature $\{A, B\}$ the relationship between these modules can be illustrated as follows:

$$\mathcal{O}^{reach}_{A\leftrightarrow B} \subseteq \bot\ \{A\}\top\{B\} \subseteq \bot\ \{A\}\top\{B\} \cup \bot\ \{B\}\top\{A\} \subseteq \bot\top\ \{A, B\} \subseteq \bot\ \{A, B\}$$

The following example shows the relationship between a bidirectional reachability-based module, bidirectional reachability-based sub-modules and minimal bidirectional reachability-based modules.

Example 2. Given the ontology $\mathcal{O}$ consisting of the set of axioms: $\{\alpha_1 : A \sqsubseteq C_1$, $\alpha_2 : A \sqsubseteq D$, $\alpha_3 : D \sqsubseteq C_3$, $\alpha_4 : C_1 \sqsubseteq \exists R.C_2$, $\alpha_5 : C_2 \sqsubseteq C_3$, $\alpha_6 : C_3 \sqcap C_4 \sqsubseteq B$, $\alpha_7 : \exists r.C_3 \sqsubseteq C_4$, $\alpha_8 C_3 \sqsubseteq B$, $\alpha_9 : C_2 \sqsubseteq E$, $\alpha_{10} : E \sqsubseteq F\}$, as well as the statement $\mathcal{O} \models A \sqsubseteq B$, we have that:

- $\mathcal{O}^{reach}_{A} = \mathcal{O}$,
- $(\mathcal{O}^{reach}_{A})^{reach}_{\overleftarrow{B}} = \mathcal{O}^{reach}_{A\leftrightarrow B}$ consist of axioms: $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8\}$
- Given that the sets $\mathcal{O}_0$, $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are defined as follows: $\mathcal{O}_0 = \{\alpha_1, \alpha_4, \alpha_5, \alpha_6, \alpha_7\}$, $\mathcal{O}_1 = \{\alpha_2, \alpha_3, \alpha_8\}$, $\mathcal{O}_2 = \{\alpha_1, \alpha_4, \alpha_5, \alpha_8\}$, $\mathcal{O}_3 = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7\}$, then
  - $\mathcal{O}_0$, $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are bidirectional reachability-based sub-modules $\mathcal{O}_i \subseteq \mathcal{O}^{reach}_{A\leftrightarrow B}$, that is, $\mathcal{O}_i = \mathcal{O}_i{}^{reach}_{A\leftrightarrow B}$.
  - $\mathcal{O}_1$ and $\mathcal{O}_2$ are both minimal bidirectional reachability-based modules with $\mathcal{O}_1$ being the only one of these sets that is both a minimal bidirectional reachability-based module and a MinA for the statement $A \sqsubseteq B$ such that $\mathcal{O}_1 \models A \sqsubseteq B$.
  - $\mathcal{O}_3$ is a MinA for the statement $A \sqsubseteq B$ such that $\mathcal{O}_3 \models A \sqsubseteq B$ but $\mathcal{O}_3$ is not a minimal bidirectional reachability-based module.

The algorithms for both bottom-up and top-down reachability based modules extraction methods may now be applied in any order and in sequence to extract bidirectional reachability-based modules. Since $\mathcal{O}^{reach}_{\overleftarrow{B}}$ is in general very large, we prefer to extract $(\mathcal{O}^{reach}_{A})^{reach}_{\overleftarrow{B}}$.

An interesting property of bidirectional reachability-based modules for $\mathcal{EL}$ is that every MinA for a subsumption statement is a bidirectional reachability-based module in terms of itself. Formally:

Theorem 2. *[5] Given an $\mathcal{EL}$ TBox $\mathcal{T}$ and the statement $A \sqsubseteq B$ such that $\mathcal{T} \models A \sqsubseteq B$. Let $M_1 \subseteq \mathcal{T}$ be a MinA such that $M_1 \models A \sqsubseteq B$, then $M_1{}^{reach}_{A\leftrightarrow B} = M_1$.*

4 MinA extraction

By Theorem 2 every MinA is a bidirectional reachability-based module. In this section we show that for the DL $\mathcal{HL}$ every MinA is a minimal bidirectional reachability-based module and that this property can be extended to acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept definitions. We also provide algorithms to compute and extract all MinAs for the given TBoxes.

Every MinA in $\mathcal{HL}$ is a *minimal* bidirectional reachability-based module in terms of itself. This is quite a subtle point, because MinAs are already minimal. Note, however, that MinAs are minimal with respect to the property of entailing a given statement of interest, whereas bidirectional reachability-based sub-modules are minimal with respect to the syntactic requirement for both bottom-up reachability and top-down reachability.

Theorem 3. *[5] Given a acyclic $\mathcal{HL}$ TBox $\mathcal{T}$ in normal form, the statement $A \sqsubseteq B$ and a MinA M_1 such that $M_1 \models A \sqsubseteq B$, then M_1 is a minimal bidirectional reachability-based module ${M_1}^{reach}_{A \leftrightarrow B}$ in terms of M_1.*

Next we show that every minimal bidirectional reachability-based module in $\mathcal{HL}$ for a statement $A \sqsubseteq B$ corresponds to a MinA.

Theorem 4. *[5] Given an acyclic $\mathcal{HL}$ TBox $\mathcal{T}$ in normal form, the statement $A \sqsubseteq B$ and a minimal bidirectional reachability-based module ${M_1}^{reach}_{A \leftrightarrow B}$, then $M_1 \models A \sqsubseteq B$.*

Theorems 3 and 4 allows us to conclude that there is a one-to-one correspondence between minimal bidirectional reachability-based modules and MinAs in $\mathcal{HL}$.

Corollary 1. *There is a one-to-one correspondence between MinAs and minimal bidirectional reachability-based modules in $\mathcal{HL}$.*

In order to extract all minimal bidirectional reachability-based modules we propose an algorithm originally inspired by the Earley [3] algorithm for parsing Context Free Grammars (CFG). Given a string to parse and a CFG the algorithm computes all possible parse trees in polynomial time. We employ a variation of the algorithm in order to compute a representation of all possible bidirectional reachability-based modules in $\mathcal{HL}$. A CFG consists of a set of CFG production rules formally defined as:

Definition 5. (CFG production rules) *Let X represent a single non-terminal, the symbol 'a' represents a single terminal and α and σ represent mixed strings of terminals and non-terminals, including the* null *string. CFG production rules have the form $X \to \alpha\sigma$ or $X \to a$.*

Any $\mathcal{HL}$ TBox can be transformed to an equivalent CFG by step by step transformation process [6, 5], with the reachability preserving CFG for an HL TBox is defined as:

Definition 6. ***Reachability preserving CFG for a $\mathcal{HL}$ TBox.***
Let $\mathcal{T}$ be an $\mathcal{HL}$ TBox in normal form and $A \sqsubseteq B$ a statement such that $\mathcal{T} \models A \sqsubseteq B$, then the reachability preserving CFG, denoted $\mathcal{CFG}_{\mathcal{T}}$, is a minimal set

of CFG production rules such that for each axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{T}$: if Sig($\alpha_L$) = $\emptyset$ the rule $x_i \rightarrow A \in \mathcal{CFG_T}$ for each $x_i \in$ Sig(α_R); for all other axioms the rule $x_i \rightarrow$ Sig(α_L) $\in \mathcal{CFG_T}$; where the symbol A represents the only terminal symbol and the set Sig($\mathcal{T}$)$\backslash A$ represents the set of non-terminals.

The conversion process may be illustrated by the following example:

Example 3. Given the acyclic $\mathcal{HL}$ TBox $\mathcal{T}$ in normal form: $\mathcal{T} = \{A \sqsubseteq B_1, A \sqsubseteq B_2, B_1 \sqsubseteq C_1, B_1 \sqsubseteq D, B_2 \sqcap C_1 \sqsubseteq D, \top \sqsubseteq B_2\}$. Then CFG$_\mathcal{T}$ is given by:
$\{ B_1 \rightarrow A, B_2 \rightarrow A, C_1 \rightarrow B_1, D \rightarrow B_1, D \rightarrow B_2C_1\}$

Once the TBox has been converted to a CFG, we employ a parallel breadth-first algorithm, adapted from the Earley [3] algorithm, further optimized and improved from an algorithm earlier presented in [6]. The algorithm computes and indexes a representation of all bi-direction sub-modules in polynomial time.

Algorithm 2 (Sub-module computation) [5] *The algorithm consists of two sub-parts, the* predictor *and* completer*. For each state in CHART, the state $(X \rightarrow \alpha\beta)$ is evaluated and the appropriate sub-part executed:*
***Input**: Reachability preserving CFG for an $\mathcal{HL}$ TBox;*
***Output**: Reference table CHART capturing a representation of all $\mathcal{HL}$ sub-modules.*

1. ***Predictor:** Given the state $(X \rightarrow Y_1 \dots Y_n)$, for all Y_i such that $(Y_i \rightarrow \sigma)$ $\notin$ CHART, add all rules $(Y_i \rightarrow \sigma)$ to CHART.*
2. ***Completer:** If state = $(X \rightarrow Z_1 \dots Z_m)$ with all Z_i terminals, then*
   - *add a pointer to this state in the completion table for X,and*
   - *if X is not a terminal symbol, then mark it a terminal symbol, and*
   - *if X is a new terminal symbol, then call the completer for each rule $(Y \rightarrow \dots X \dots) \in$CHART such that all symbols on the right hand side of the rule are terminal symbols.*

The algorithm executes all states iteratively in a top-down manner until no new states are available for processing. Given the statement $A \sqsubseteq B$, the production rule $S \rightarrow B$ is used to initialize CHART.

Theorem 5. *Given a acyclic $\mathcal{HL}$ TBox $\mathcal{T}$ in normal form and the statement of interest such that $\mathcal{T} \models A \sqsubseteq B$, with CFG$_\mathcal{T}$ the context free grammar associated with $\mathcal{T}$. If n is the number of production rules in CFG$_\mathcal{T}$, then Algorithm 2 computes a representation of all possible bidirectional reachability-based modules in $O(n^2)$ worst case running time.*

Once Algorithm 2 terminates, the chart returned contains a representation of all possible bidirectional reachability-based modules, and hence a representation of all MinAs. This set is essentially an indexed bidirectionally reachable module.

In order to obtain all individual MinAs from the CHART returned by Algorithm 2, we introduce an algorithm to extract all minimal bidirectional reachability-based modules from it. Due to the space limitations of this paper we do not give an implementation of the algorithm but refer the interested reader to [5].

Theorem 6. *Given the indexed bidirectional reachability-based $\mathcal{HL}$ CFG for the statement $A \sqsubseteq B$, $CFG_{\mathcal{O}}$ and the reference table CHART returned by Algorithm 2, the algorithm to extract all individual MinAs will extract all MinAs M_i such that $M_i \models A \sqsubseteq B$. Each M_i will be extracted in $O(m^2)$ worst case running time, where $m = |Sig(CFG_{\mathcal{O}})|$.*

The algorithm introduced may be extended to extract all MinAs for acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept definitions. However, we show that though all MinAs for these TBoxes are minimal bidirectional reachability-based modules, the converse does not hold.

Example 4. Let $\mathcal{T}$ be an acyclic $\mathcal{EL}$ TBox consisting of only primitive concept definitions. Further, let M_1 be a minimal bidirectional reachability-based module for the statement $A \sqsubseteq B$ consisting of the axioms $A \sqsubseteq \exists r.C$ and $C \sqsubseteq B$. Then ${M_1}^{reach}_{A \leftrightarrow B} = M_1$ and M_1 is minimal, but $M_1 \not\models A \sqsubseteq B$ unless $M_1 \models B \sqsubseteq \bot$. Hence M_1 is not a MinA for $\mathcal{T} \models A \sqsubseteq B$.

Theorem 7. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox consisting of only primitive concept definitions in normal form, and let $A \sqsubseteq B$ be a statement such that $\mathcal{T} \models A \sqsubseteq B$. Then for every minimal bidirectional reachability-based module N_i such that $\alpha_L \sqsubseteq \exists r.C \in N_i$ we have that $N_i \not\models A \sqsubseteq B$. Further, for every minimal bidirectional reachability-based module M_i such that $\alpha_L \sqsubseteq \exists r.C \notin M_i$ we have that $M_i \models A \sqsubseteq B$.*

Consequently, the algorithms presented may be used in order to extract all minimal modules and thus MinAs for acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept definitions. When a minimal module includes axioms containing existential restrictions this module may simply be discarded as not being a MinA. The algorithm is complete in that it will extract all MinAs. However, since not all minimal modules extracted are MinAs, it is no longer sound. Soundness may however be obtained by simply making the test for the inclusion of existential restrictions part of the algorithm. When extending the problem of finding MinAs to general $\mathcal{EL}$ TBoxes, a staight forward extraction process is no longer possible and every possible matching between symbols needs to be calculated by the algorithm. Thus a simple iteration of all minimal bi-directional reachability based modules in order to find a single MinA results in an algorithm that runs in exponential worst case time.

Theorem 8. *Let $\mathcal{T}$ be an acyclic general $\mathcal{EL}$ TBox in normal form and $A \sqsubseteq B$ a statement of interest. Let CHART represent the resultant reference set returned by Algorithm 2. Let M_1 be a MinA such that $M_1 \models A \sqsubseteq B$ and P_1 represent the set of production rules for M_1. Now let m_i be the number of times a symbol C_i occurs on the right hand side of all production rules in P_1 and let k_i be the number of entries in CHART[C_i]. Then for the n possible symbols in P_1 there are a total of $\prod_{i=1}^{n} \sum_{j=1}^{j=m_i \leq k_i} C\binom{k_i}{j}$ bidirectional reachability-based modules.*

Though we believe that this theoretical worst case complexity will not pose a problem for real world $\mathcal{EL}$ medical ontologies, subsumption testing will be required once each minimal module have been extracted.

5 Empirical Results

In this section we test the algorithms presented in this paper and evaluate their performance in terms of three real world biomedical ontologies[2]: $\mathcal{O}_{Snomed}$ - The Systematized Nomenclature of Medicine, Clinical Terms; $\mathcal{O}_{Nci}$ - The Thesaurus of the US National Cancer Institute and $\mathcal{O}_{Go}$ - The Gene Ontology.

The algorithms presented were all implemented in Java as part of a plugin for the Protégé 4.1 (beta) ontology editor. All single threaded algorithms were tested on a Intel Quad Core based computer, with 6 Gig of RAM, running on Microsoft Windows 7 x64 and hosted in a 64 bit Java virtual machine. We did not implement nor utilise an optimized subsumption testing algorithm for inexpressive DLs. Subsumption testing were done by the standard HerMit[3] reasoner where neccesary.

Table 1 show the results of all bidirectional reachability-based modules extracted. The columns in the table are organised as follows: Ontology – the ontology for which the modules are being extracted; $\mid \mathcal{O}_A^{reach} \mid$ – the number of axioms in the reachability-based modules for all concepts $A \in$Sig($\mathcal{O}$); T($\mathcal{O}_A^{reach}$) – the average time, in seconds, required by the algorithm to extract all reachability-based modules; $\mid \mathcal{O}_{A \leftrightarrow B}^{reach} \mid$ – the average number of axioms for all bidirectional reachability-based modules; T($\mathcal{O}_{A \leftrightarrow B}^{reach}$) – the additional time, in seconds, required to extract the bidirectional reachability-based modules, i.e. Total time = T($\mathcal{O}_A^{reach}$) + T($\mathcal{O}_{A \leftrightarrow B}^{reach}$).

Average Values

Ontology	$\mathcal{O}_A^{reach}$	T($\mathcal{O}_A^{reach}$)	$\mathcal{O}_{A\leftrightarrow B}^{reach}$	T($\mathcal{O}_{A\leftrightarrow B}^{reach}$)
$\mathcal{O}_{Go}$	13.16	0.000032	4.48	0.000006
$\mathcal{O}_{Nci}$	25.68	0.000048	5.59	0.000006
$\mathcal{O}_{Snomed}$	27.70	0.040725	18.40	0.000175

Maximum Values

Ontology	$\mathcal{O}_A^{reach}$	T($\mathcal{O}_A^{reach}$)	$\mathcal{O}_{A\leftrightarrow B}^{reach}$	T($\mathcal{O}_{A\leftrightarrow B}^{reach}$)
$\mathcal{O}_{Go}$	68	0.000417	20.15	0.000666
$\mathcal{O}_{Nci}$	398	0.001916	55.00	0.000569
$\mathcal{O}_{Snomed}$	254	0.217781	222.06	0.004843

Median Values

Ontology	$\mathcal{O}_A^{reach}$	T($\mathcal{O}_A^{reach}$)	$\mathcal{O}_{A\leftrightarrow B}^{reach}$	T($\mathcal{O}_{A\leftrightarrow B}^{reach}$)
$\mathcal{O}_{Go}$	10	0.000026	3.86	0.000005
$\mathcal{O}_{Nci}$	11	0.000026	4.37	0.000005
$\mathcal{O}_{Snomed}$	16	0.001800	6.66	0.000008

Table 1. Bidirectional reachability-based module extraction

From the table we see that bidirectional reachability-based modules are between 30% and 80% smaller than standard reachability-based modules and may

[2] http://lat.inf.tu-dresden.de/systems/cel/

[3] http://hermit-reasoner.com/

be extracted at the additional cost of between 0.4% and 19.0% in the running time of the algorithm. The average runtime increases for the GO and NCI ontologies tested may seem excessively high. However, we note that the running times are measured in the low microsecond range. At these extremely small intervals the accuracy of our measuring tools is very low and the true runtime performance of the algorithms only becomes evident in relatively large ontologies. Therefore, the runtime performance of the algorithms for the SNOMED ontology gives a more accurate measure of the true performance of the algorithms. In terms of median values extracting bidirectional reachability-based modules results in very stable performance across all ontologies tested, with an approximate 59% decrease in the size of all modules extracted.

The MinA extraction algorithms were tested as follows: for every concept name $A \in$ in Sig($\mathcal{O}$) we extracted $\mathcal{O}^{reach}_{A}$, then Algorithm 2 was called for each concept name $B \in$Sig($\mathcal{O}^{reach}_{A}$) in order to extract $\mathcal{O}^{reach}_{A\leftrightarrow B}$. For each of these indexed bidirectional reachability-based modules we then extracted all possible minimal bidirectional reachability-based modules M_i. The standard HerMit reasoner was then called to test if $M_i \models A \sqsubseteq B$. This subsumption test is irrelevant and is only included for the sake of interest.

The columns in Table 2 are organised as follows: Ontology – the ontology for which the MinAs are being extracted; $\mid \mathcal{O}^{reach}_{A\leftrightarrow B} \mid$ – the average number of axioms for all bidirectional reachability-based modules; $\mid Min(\mathcal{O}^{reach}_{A\leftrightarrow B}) \mid$ – the average number of minimal bidirectional reachability-based modules; T($Min(\mathcal{O}^{reach}_{A\leftrightarrow B})$) – the additional time, in seconds, required to extract all minimal bidirectional reachability-based modules; %MinAs – the percentage of minimal bidirectional reachability-based modules that are MinAs; |MinA| – the average size of each MinA and T(MinA) – the additional time required to test subsumption for all minimal modules, i.e. to calculate the total time to extract all MinAs from the ontology = T($Min(\mathcal{O}^{reach}_{A\leftrightarrow B})$) + T(MinA).

Average Values

Ontology	$\mid \mathcal{O}^{reach}_{A\leftrightarrow B} \mid$	$\mid Min(\mathcal{O}^{reach}_{A\leftrightarrow B}) \mid$	T($Min(\mathcal{O}^{reach}_{A\leftrightarrow B})$)	%MinAs	|MinA|	T(MinA)
$\mathcal{O}_{Go}$	13	2.720188	0.000023	89.18%	3.298866	0.005472
$\mathcal{O}_{Nci}$	26	2.180851	0.000014	91.47%	3.721915	0.002842

Median Values

Ontology	$\mid \mathcal{O}^{reach}_{A\leftrightarrow B} \mid$	$\mid Min(\mathcal{O}^{reach}_{A\leftrightarrow B}) \mid$	T($Min(\mathcal{O}^{reach}_{A\leftrightarrow B})$)	%MinAs	|MinA|	T(MinA)
$\mathcal{O}_{Go}$	10	1.500000	0.000013	100.00%	3.000000	0.002327
$\mathcal{O}_{Nci}$	11	1.000000	0.000010	100.00%	3.500000	0.001452

Table 2. MinA extraction

On average there are between 2 and 3 minimal bidirectional reachability-based modules for each possible subsumption statement. From these, about 90% are MinAs, each of which contains between 3 and 4 axioms on average. The total additional time required to extract all minimal bidirectional reachability-based modules, and thus MinAs, is in the low microsecond range. The most expensive costs incurred was subsumption testing, with a total running time for testing all minimal modules in the low to mid millisecond range. This testing however is unnecessary and is only included to illustrate the costs involved in testing all minimal modules for subsumption.

Once a bottom-up reachability-based module has been extracted, the additional runtime costs incurred to extract a bidirectional reachability module together with all minimal bidirectional reachability-based modules, and thus MinAs, is less that 1% of the cost of performing a subsumption test on a single MinA. This makes MinA extraction, for acyclic $\mathcal{EL}$ TBoxes consisting of only primitive concept definitions, negligible. The average reduction of 59% in the number of axioms for bidirectional reachability-based modules tested here, over that of standard reachability-based modules, indicates that for more expressive DLs in the $\mathcal{EL}$ family, bidirectional reachability-based modules may yield a significant improvement during MinA extraction to standard black-box algorithms.

Extension of the techniques presented here to more expressive DLs using hypergraph grammars, and relating it to the techniques and complexity results presented in [7], are topics of further research. We thank the anonymous reviewers for their comments on related and further work.

References

1. Ausiello, G., Franciosa, P.G., Frigioni, D.: Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In: Proceedings of the Seventh Italian Conference on Theoretical Computer Science (ICTCS), LNCS, vol. 2202, pp. 312–327. Springer, London, UK (2001)
2. Du, J., Qi, G., Ji, Q.: Goal-directed module extraction for explaining OWL DL entailments. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) Proceedings ISWC'09, LNCS, vol. 5823, pp. 163–179. Springer, Berlin Heidelberg (2009)
3. Earley, J.: An efficient context-free parsing algorithm. Communications of the Association for Computing Machinery 13(2), 94–102 (1970)
4. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research 31, 273–318 (2008)
5. Nortjé, R.: Module extraction for inexpressive description logics. Master's thesis, University of South Africa (2011), submitted.
6. Nortjé, R., Britz, K., Meyer, T.: Finding $\mathcal{EL}^+$ justifications using the Earley parsing algorithm. In: Meyer, T., Taylor, K. (eds.) Australasian Ontology Workshop 2009 (AOW 2009). CRPIT, vol. 112, pp. 27–35. ACS, Melbourne, Australia (2009)
7. Peñaloza, R., Sertkaya, B.: On the complexity of axiom pinpointing in the EL family of description logics. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proceedings KR-10. AAAI Press, Toronto, Canada (2010)
8. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) 22nd International Workshop on Description Logics (DL2009). CEUR-WS, Oxford, UK (2009)
9. Suntisrivaraporn, B.: Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. Ph.D. thesis, Technical University of Dresden (2009)

On the Problem of Weighted Max-DL-SAT and its Application to Image Labeling

Carsten Saathoff, Stefan Scheglmann, and Steffen Staab

Institute for Web Science and Technologies
University of Koblenz-Landau, Germany
http://west.uni-koblenz.de
saathoff,schegi,staab@uni-koblenz.de

Abstract. For a number of problems, such as ontology learning or image labeling, we need to handle uncertainty and inconsistencies in an appropriate way. Fuzzy and Probabilistic Description Logics are the two major approaches for performing reasoning with uncertainty in Description Logics, but modeling problems such as image labeling still remains difficult and handling inconsistencies is only supported to a limited extent. In this paper, we propose Max-DL-SAT and Weighted Max-DL-SAT as new reasoning services for Description Logics knowledge bases, which applies the idea behind Weighted Max-SAT to Description Logics and leads to a more intuitive representation of certain problems. It supports handling of uncertainty and inconsistencies. The contribution of this paper is threefold: We define a novel reasoning service on Description Logics knowledge bases, introduce an algorithm for solving such problems, and show the application of it to the problem of image labeling.

1 Introduction

Solutions to a number of real world problems are often subject to a set of potentially contradicting constraints, for which a completely satisfying solution does not exist, e.g., in Computer Aided Design or information extraction from text. In Max-SAT, these problem are modeled as a boolean a formula for which one seeks an assignment of truth values that satisfies a maximal number of clauses. In Weighted Max-SAT clauses are associated with weights that model the importance or reliability of certain clauses and the goal is to maximize the accumulated weight of the satisfied clauses in a solution. However, Max-SAT and Weighted Max-SAT are both limited to propositional logic and finding an appropriate problem representation is often hardly intuitive. Ontologies, on the other hand, allow for modeling domains in a more intuitive manner. Description Logics have widely been adopted to model ontologies and to provide reasoning services in a variety of domains. In problems like image labeling [13, 2], we encounter assertions that are associated with a degree and we have to cope with many, potentially contradicting assertions produced by automatic and not fully reliable methods. In order to apply ontological reasoning to such problems, we require new reasoning services. State of the art extensions to Description Logics, such as Fuzzy [15] or Probabilistic Description Logics [8] cover most of these aspects, but other problems still need to be solved. Reasoning on Probabilistic Description Logics still has difficulties regarding the efficiency of the reasoning process. Fuzzy Description Logics can be reasoned about efficiently under min/max co-norm.

In this paper, we introduce a novel reasoning service for handling uncertainty and inconsistencies in Description Logic knowledge bases, called Weighted Max-DL-SAT. We consider Description Logic knowledge bases containing a set of weighted and potentially contradicting axioms. Based on this, we compute a set of consistent axioms with a maximal, accumulated weight. Weighted Max-DL-SAT allows for the almost direct reuse of existing ontologies and provides a very intuitive way of modeling problems that have a Max-SAT like structure, e.g., the aforementioned image labeling problem. In summary, the paper provides a threefold contribution:

1. We define a novel type of reasoning problem, called Weighted Max-DL-SAT.
2. We introduce an algorithm for solving Weighted Max-DL-SAT problems based on the Hitting Set Tree algorithm [12, 6].
3. We apply Weighted Max-DL-SAT to the problem of image labeling.

The rest of the paper is structured as follows: In the next section we give a formal introduction to the Description Logic $\mathcal{ALC}$ and extend its definition to *weighted ontologies*. Then, we introduce the problem of Weighted Max-DL-SAT. Based on this formalizations, we introduce our approach to solve Weighted Max-DL-SAT problems. Afterwards, we introduce an example where we applied Weighted Max-DL-SAT to the domain of spatial reasoning in the context of image labeling, and finally discuss the related work and conclude the paper.

2 Knowledge Representation Using Description Logics

Description Logics constitutes a class of knowledge representation languages that allow for expressing complex concepts in terms of a set of basic constructors. In this section, we specifically introduce the Description Logic $\mathcal{ALC}$, the Attributive Concept Language with Complements. Let N_C and N_R be two disjoint sets of symbols, called the set of *concept* and *role names*, respectively. We will write A, B for concept names, and R for role names. $\top$ and $\bot$ are special concepts, called *Top* and *Bottom*, respectively. A *concept description* C in $\mathcal{ALC}$ is syntactically defined by the following abstract syntax rule:

$$C \rightarrow \top|\bot\ |A|\forall R.C|\exists R.C|C \sqcap D|C \sqcup D|\neg C\}. \tag{1}$$

The semantics of a concept description is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}} = \{a_i, \ldots, a_n\}$ is called the domain of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is called the interpretation function. The interpretation function maps each concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The semantics of the constructors are defined as follows.

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $\bot^{\mathcal{I}} = \emptyset$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$

– $(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

Furthermore, we define a T-Box $\mathcal{T}$ as a set of terminological axioms of the form $C \sqsubseteq D$, whereby C and D are concepts. We say that D subsumes C, and an interpretation $\mathcal{I}$ satisfies an axiom $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We define a disjointness between two axioms C and D as $C||D = C \sqsubseteq \neg D$. The A-Box $\mathcal{A}$ is defined as a set of concept assertions $a : C$, where a is an individual name and C a concept description, and role assertions $(a, b) : R$ with a, b individual names and R a role name. Both concept and role assertions area also called *assertional axioms*. A Description Logics ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$consists of a $\mathcal{T}$-Box $\mathcal{T}$ and an $\mathcal{A}$-Box $\mathcal{A}$.

We extend this definition of an ontology to a *weighted ontology*:

Definition 1 (Weighted ontology). *A weighted ontology is an ontology $\mathcal{O} := \{\alpha_1, \ldots, \alpha_n\}$ such that for every $\mathcal{T}$-Box or $\mathcal{A}$-Box axiom $\alpha \in \mathcal{O}$ an associated weight $w_\alpha \in \mathbb{R}^+$ exists. In case of concrete axioms, we specify the weight in square brackets, i.e., $C \sqsubseteq D[w]$ for $\mathcal{T}$-Box axioms, $a : C[w]$ for concept assertions, and $(a, b) : R[w]$ for role assertions.*

We can now define the reasoning problem Weighted Max-DL-SAT. Our basis is a weighted Description Logic ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ Each axiom in $\mathcal{O}$ is associated with a weight, which represents the importance or reliability of this axiom to be satisfied.

The problem is to find a consistent subset of the ontology with a maximal summed weight. Formally, we define the Weighted Max-DL-SAT problem as an optimization problem as follows:

$$\text{argmax}_{S \subseteq \mathcal{O} \text{ s.t. } S \text{ consistent}} (\sum_{\alpha \in S} w_\alpha) \tag{2}$$

The result is $\mathcal{O}_r = \mathcal{T}_r \cup \mathcal{A}_r$, a maximal consistent sub-ontology, such that $\mathcal{O}_r \subseteq \mathcal{O}$, $\mathcal{O}_r$ is consistent, and the accumulated weight of all axioms $\alpha_i \in \mathcal{O}_r$ is maximal. In $\mathcal{O}_r$ we call $\mathcal{A}_r$ the consistent Sub-$\mathcal{A}$-Box and $\mathcal{T}_r$ the consistent Sub-$\mathcal{T}$-Box.

3 Solving Weighted Max-DL-SAT Problems

In order to obtain a consistent sub-ontology, we need to resolve all inconsistencies in $\mathcal{O}$. To do so, we have to calculate the weight-minimal set of axioms $\mathcal{O}^-$, such that $\mathcal{O}_r = \mathcal{O} \setminus \mathcal{O}^-$ is consistent. This problem has strong relations to axiom-pinpointing [14], which identifies and eliminates inconsistencies in ontologies. Axiom-pinpointing algorithms compute a minimal set of axioms causing a single inconsistency in an ontology $\mathcal{O}$. Such a set, we call a minimal inconsistent sub-ontology [3] M and it is defined as follows:

Definition 2 (Minimal Inconsistent Sub-Ontology). *A Minimal Inconsistent Sub-Ontology (M) of an ontology $\mathcal{O}$, is defined as a subset $M \subseteq \mathcal{O}$, such that M is inconsistent and $\forall \alpha \in M : M \setminus \{\alpha\}$ is consistent.*

Every M causes a single inconsistency in a particular $\mathcal{O}$. If we remove one axiom of such a M from $\mathcal{O}$, we eliminate this cause of inconsistency in $\mathcal{O}$.

We can formulate this problem as a Weighted Hitting Set Problem. We first give the definition of this set of problems and then explain how Weighted Max-DL-SAT maps to a Weighted Hitting Set Problem:

Definition 3 (Weighted Hitting Set Problem [12]). *Given a set $\mathcal{G}$ and a set of subsets $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n \subseteq \mathcal{G}$ Each element in $\mathcal{G}$ has a positive weight $w_a, a \in \mathcal{G}$ We are looking for a hitting set $\mathcal{H} \subseteq \mathcal{G}$ such that*

– $\mathcal{H} \cap \mathcal{M}_i \neq \emptyset, i = 1, ..., n$
– $\sum_{a \in \mathcal{H}} w_a$ *is minimal*

Now let $\mathcal{O}$ be our ground set, $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$ the set of all minimal inconsistent sub-ontologies, and the hitting set $\mathcal{H}$ the set $\mathcal{O}^-$ of axioms to be removed from $\mathcal{O}$. Obviously $\mathcal{O}_r$ is weight maximal, when $\mathcal{O}^-$ is weight minimal.

To calculate $\mathcal{O}^-$ for inconsistent ontologies, we propose an adaptation of the Hitting Set Tree (HST) algorithm. The HST algorithm produces a tree T starting with our $\mathcal{O}$ as root node N_1. It calculates a minimal inconsistent Sub-Ontology (MISO) M_j for every node $N_j \in T$. For every axiom α_i in M_j of N_j the algorithm introduces a new sub-node $N_{ji} \in T$. The edge to N_{ji} is labeled with α_i and w_{α_i} and the current ontology for a node N_{ji} is $O_j \setminus \{\alpha_i\}$. To solve our Weighted Max-DL-SAT problem, we have to calculate the cheapest path w.r.t the accumulated weights from the root to a leaf in T. The accumulated axioms of this path represent $\mathcal{O}^-$.

Algorithm 1 Weighted Hitting Set Tree algorithm for computing a solution to Weighted Max-DL-SAT problems.

```
 1: $\mathcal{O}^- \leftarrow \emptyset$                              ▷ initialize result with empty set
 2: $w_{\mathcal{O}^-} \leftarrow \infty$                             ▷ set upper bound to infinity
 3: function WHST($O$, $P$)
 4:     $w_P \leftarrow \sum_{\alpha \in P} w_P$                       ▷ accumulate path weight
 5:     if $w_P < w_{\mathcal{O}^-}$ then                                   ▷ check upper bound
 6:         $M \leftarrow$ calcSingleMISO($O$)                ▷ calculate $M$ for current ontology
 7:         if $M \neq \emptyset$ then
 8:             $M' \leftarrow M$
 9:                                          ▷ $\forall \alpha \in M$ in decreasing order call WHST
10:                                                    ▷ for $O \setminus \{\alpha\}, P \cup \{\alpha\}$
11:             while $M' \neq \emptyset$ do
12:                 Select $\alpha \in M'$ s.t. $\forall \alpha' \in M' \rightarrow w_{\alpha'} > w_\alpha$
13:                 $M' \leftarrow M' \setminus \{\alpha\}$
14:                 WHST($O \setminus \{\alpha\}, P \cup \{\alpha\}$)
15:             end while
16:         else                                  ▷ if current path weight < upper bound
17:             $\mathcal{O}^- \leftarrow P$                                      ▷ set result to path
18:             $w_{\mathcal{O}^-} \leftarrow w_P$                       ▷ set upper bound to path weight
19:         end if
20:     end if
21: end function
```

In [6] Kalyanpur et. al have shown the completeness of the Hitting Set Tree algorithm regarding the calculation of all justifications for an ontology. Algorithm 1 depicts the concrete $WHST$ algorithm used in our implementation. To increase efficiency, we use a branch & bound like strategy to prune subtrees where no further improvements of the results could be achieved. We use the accumulated weight of an already calculated root-to-leaf path as upper bound, line 19. Initially this bound is set to infinity, line 2. With this lower bound, we can prune any branch of the subtree that could not contain a smaller total path weight. Only if the weight of the current path is lower than this upper bound, a branch has to be considered, line 5.

Algorithm 2 depicts the minimal inconsistent sub-ontology (MISO) calculation [3]. It iteratively adds the axioms α with the smallest weight w_α to the intermediate ontology O until it becomes inconsistent, lines $3-6$. Then, we shrink O by iteratively removing the axioms α with the biggest weight w_α if this does not turn O consistent again, lines $8-14$. Thus, we are guaranteed to end up with an M, a small, still inconsistent set of axioms in O.

Algorithm 2 Black-box algorithm for computing a minimal inconsistent sub-ontology for $\mathcal{O}$.

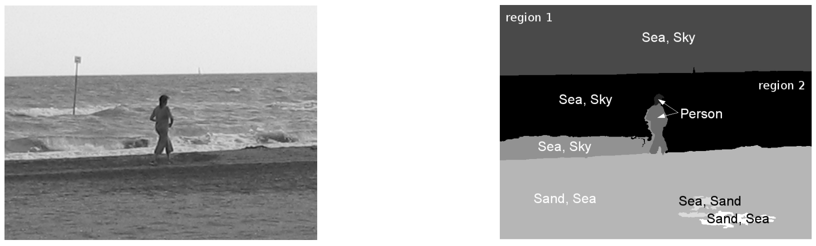
```
 1: function CALCSINGLEMISO($\mathcal{O}$)
 2:     $O \leftarrow \emptyset$                                        ▷ initialize intermediate ontology
 3:     while $O$ is consistent do                     ▷ grow intermediate ontology until inconsistency
 4:         Select axiom $\alpha \in \mathcal{O} \setminus O$ s.t. $\forall \alpha' \in \mathcal{O} \setminus O \rightarrow w_{\alpha'} \geq w_\alpha$
 5:         $O \leftarrow O \cup \{\alpha\}$
 6:     end while
 7:     $O' \leftarrow O$
 8:     while $O' \neq \emptyset$ do             ▷ shrink intermediate ontology to minimal inconsistent set
 9:         Select axiom $\alpha \in O'$ s.t. $\forall \alpha' \in O' \rightarrow w_{\alpha'} \leq w_\alpha$
10:         $O' \leftarrow O' \setminus \{\alpha\}$
11:         if $O \setminus \{\alpha\}$ is inconsistent then
12:             $O \leftarrow O \setminus \{\alpha\}$
13:         end if
14:     end while
15:     return $O$
16: end function
```

4 Applying Weighted Max-DL-SAT to Automatic Image Labeling

As an example, we present the application of Weighted Max-DL-SAT to the interesting problem of automatically assigning labels to image regions. Typically, these labels refer to ”semantic” concepts and provide the means to index regions within an image based on terms understandable for humans. Determining the right labels for a given region is a hard problem, since there is no direct mapping of computable low-level features to the meaning of a region. Automatic methods model regions within images using a set of features and then usually apply machine learning methods in order to learn and subsequently detect a set of possible semantic concepts.

These methods exploit only low-level features extracted from regions of the image, but do not take any context, e.g., spatial context, into account. However, context and background knowledge play a crucial role in automatic image labeling [13, 2]. In our experiments, we utilize spatial relations between image regions as background knowledge to validate the semantic concepts given for specific region.

Figure 1 shows an example of an image (a) and the associated regions with simplified, but still ambiguous hypotheses (b) produced by a classifier. The output of the machine-learning-based classification is used as input to our reasoning process. As background knowledge, we consider knowledge about feasible spatial relations between the semantic concepts, such as *above, below, left, right*. For example, a valid relation might be that *sea* is never depicted above *sky*.



(a) input image (b) output from ML-based classification

Fig. 1. Input to the reasoning process

4.1 Data Set

The data set consists of 922 images depicting outdoor scenes and was split into 400 training and 522 test images. These images have been segmented using an automatic segmentation algorithm and manually assigned a label from the set of concepts: Sky, Sea, Sand, Road, Building, Foliage, Person, Boat, Mountain, Snow. This dataset has been published[1] and used in previous experiments [13, 10] for the task of spatial reasoning. In addition, the data set also contains different low-level features for each region, different hypotheses generated based on the training data using different classification methods, and a set of extracted fuzzy spatial relations. For our experiment, we used the labels produced by the maximum-likelihood classifier as input to our reasoner.

4.2 Representing Image Labeling with Weighted Max-DL-SAT

The background knowledge is depicted as a $\mathcal{T}$-Box. For each label, we create an atomic concept L. Furthermore, we make all label concepts disjoint and add an axiom $L_1 || \ldots || L_n[w_n]$.

The background knowledge about spatial relations has been modeled as a set of binary constraints defining for each label L to which other labels $L'_1, \ldots, L'_n$ it might be related by the spatial relation S. To present such knowledge about spatial relations

[1] http://mklab.iti.gr/project/scef

in a Description Logic $\mathcal{T}$-Box, we use universal quantification, like $L \sqsubseteq \forall S.(L'_1 \sqcup \ldots \sqcup L'_n)[w_n]$. Thus, for the two labels *Sky* and *Sea* used in figure 1, we would add two axioms $sky \sqsubseteq \forall above.(sea \sqcup sky \sqcup sand \sqcup \ldots)[w_m]$ and $sea \sqsubseteq \forall above.(sea \sqcup sand \sqcup \ldots)[w_m]$. These axioms assure, that sea might only be depicted above other sea regions, while sky might be depicted above sky or sea regions. They are all associated with an very high weight, since we consider the background knowledge as crisp, and therefore do not accept any solutions where any of these axioms is removed. Obviously, this requires that the $\mathcal{T}$-Box is consistent, which is the case in our experiments. Furthermore, the axioms are learned following the approach presented in [13].

Each image i is modeled in a separate $\mathcal{A}$-Box. To generate the $\mathcal{A}$-Box, we use the hypothesis generated by the machine-learning classification process. For each region, we create a single individual r_i. Now, let $w_{i,l}$ be the degree of confidence in the dataset for region r_i labeled with label l. Then we add the concept assertion $r_i : L[w_{i,l}]$ to the knowledge base for each label produced by the classifier for the region. For the two regions *region1* and *region2* depicted in figure 1 this will result in: $region1 : sky[w_{region1,sky}]$, $region1 : sea[w_{region1,sea}]$, $region2 : sky[w_{region2,sky}]$ and $region2 : sea[w_{region2,sea}]$. Additionally to the hypothesis about associated semantic concepts the classification process also generates knowledge about spatial relations between the single regions. To present the spatial knowledge in our ontology, we add for all known relations role assertions like $(r_i, r_j) : S[w_m]$ to the knowledge base. We set the weight of such assertions to very high value, because we do not the accept a solution where one of the spatial relations was removed in order to find a solution. For the regions *region1* and *region2* from figure 1, this will lead to the two role assertion $(region1, region2) : above[w_m]$ and $(region2, region1) : below[w_m]$. Together with the $\mathcal{T}$-Box depicting the background knowledge, this $\mathcal{A}$-Box results in an individual ontology $\mathcal{O}_i$ for each image i. As we can see this ontology contains contradicting statements with: $sea \sqsubseteq \forall above.(sea)[w_m]$, $(region1, region2) : above[w_m]$, $region1 : sea[w_{region1,sea}]$ and $region2 : sky[w_{region2,sky}]$. Such an inconsistent ontology $\mathcal{O}_i$ is the input to our reasoning process.

4.3 Results

In Table 4.3, we have summarized the accuracy of the classifier, Weighted Max-DL-SAT, and the binary integer programming approach presented in [13]. Using Weighted Max-DL-SAT, we can significantly improve the classification rate as provided by the classifier based solely on low-level features. However, we also see that a more specialized method performs clearly better. The latter observation was expected. The BIP approach can employ a more specialized objective function that incorporate the degree of confidence provided with the fuzzy spatial relations, and it employ all fuzzy spatial relations available, not only the one with the highest degree. This information is not used in our modeling of the problem.

Nevertheless, the experiments show that a generic approach based on Description Logics can be applied to a problem like spatial reasoning and leads to a clear improvement. Furthermore, the difference between the specialized method and Weighted Max-DL-SAT is not very large. Specifically, the parameters used for the knowledge extraction have not been optimized in our experiments for Weighted Max-DL-SAT, while

	Classifier	Max-DL-SAT	BIPs
building	0.92	0.90	0.96
foliage	0.70	0.85	0.90
mountain	0.74	0.92	0.91
person	0.58	0.77	0.84
road	0.75	0.56	0.92
sailing-boat	0.49	0.47	0.93
sand	0.67	0.69	0.63
sea	0.71	0.78	0.75
sky	0.17	0.52	0.51
snow	0.71	0.81	0.85
overall	0.62	0.75	0.79

Table 1. Per concept and overall accuracy of the classifier, Weighted Max-DL-SAT, and the Binary Integer Programming approach [13].

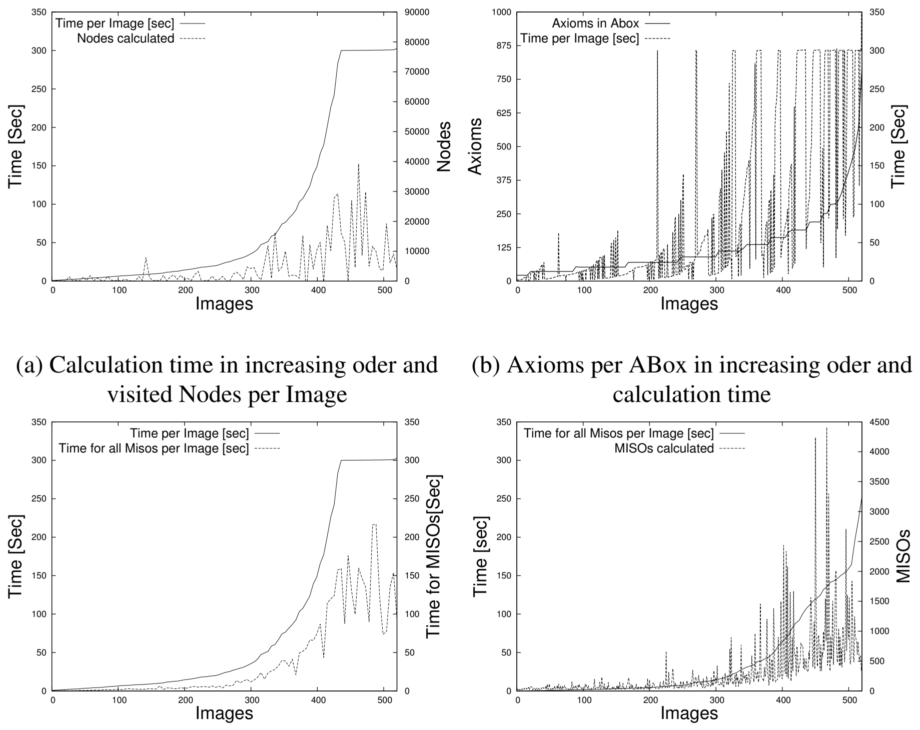
in [13] experiments with optimized parameters were reported. The gained generality of the approach comes at the cost of a loss in accuracy.

The figures in 2 show the system performance results from our experiment. In each figure, we compare two different values (left and right y-axies) per image (x-axis). We sorted the images in increasing order by the first value (left). In figure 2 (a), we show the relation between the over all calculation time per image and the number of nodes per image. In our Experiments, we limited the calculation time per image to $300sec$. We can observe only a weak relationship between calculation time and the number of visited nodes, a tendency towards the more nodes are visited the longer the calculation takes. We can observe multiple outliers especially images with a relative small number of visited sodes compared to the calculation time. Due to the heuristic character of Branch & Bound and because of the calculation of an NP-complete problem like the Weighted Hitting Set Tree, we have to expect such outliers. Figure 2 (b) shows the number of $\mathcal{A}$-Box axioms per image in increasing oder and over all calculation time. Again we can observe multiple outlier but the relation seems to be stronger. Images with more axioms in the $\mathcal{A}$-Box more often tend to exceed the calculation time cap. In figure 2 (c), we show the relation between the over all system performance per image and the time consumption for MISO calculation per image. The MISO calculation time seems to represent a relatively large proportion of the over all system performance. This could be a interesting point for further optimizations. The system could benefit from more detailed studies to increase the efficiency of the MISO calculation. The last figure, 2 d shows the relation between the time consumption for MISO calculation per image and the number of MISOs per image. Here we can also observe an clear relationship. This observation also indicates that where is a potential for further optimizations of the MISO calculation.

All these behavior result give clear hints about further optimizations of the systems performance. A promising starting point for further optimizations seems to be the MISO calculation. The MISO calculation takes an important part of the over all calculation time and the calculation time for all MISO increases similar to the number of MISOs.

5 Related Work

The issues of integrating uncertainty into Description Logics and reasoning with such uncertainty in Description Logics have already been addressed in different ways by



(a) Calculation time in increasing oder and visited Nodes per Image

(b) Axioms per ABox in increasing oder and calculation time

(c) Calculation time in increasing oder and MISO Time

(d) Calculation time for all MISOs in increasing oder and MISOs per Image

Fig. 2. System performance

several researchers. [5, 8, 7] present probabilistic extensions to OWL or investigations on reasoning services for such extension. Some of these approaches allow only for terminological knowledge like [5] others for terminological as well as assertional knowledge [8]. The approaches also differ in the underlying probabilistic reasoning formalism. Two reasoning formalisms could be found in these publications, a formalism based on reasoning in probabilistic logics [5, 8] and a formalism based on inferencing in Bayesian networks [7]. But unfortunately all of these approaches suffer from serve problems regarding the efficiency of reasoning on such knowledge bases. Another approach to uncertainty extension to Description Logics is the use of fuzzy set theory to express the uncertainty. In [16, 15] Straccia presents a general approach to a fuzzy version of $\mathcal{SHOIN}(D)$, the underlying Description Logic of $OWL - DL$. He shows the representation and reasoning capabilities of fuzzy $\mathcal{SHOIN}(D)$. As mentioned in the introduction, reasoning on fuzzy Description Logics can be performed quite efficiently. However, the fuzzy semantics are often mislead by single axioms with a high or low weight, repsectively. In general, both approaches are able to handle degrees associated with axioms, but they are not suitable for handling inconsistencies in every respect.

Reasoning under inconsistency plays a role in the field of ontology learning [4] and ontology debugging [3]. One important method, about finding explanations for a given consequence, e.g., a minimal subset of an ontology that has a particular inconsistency in that ontology as consequence, is axiom pinpointing. Generally, we can distinguish axiom pinpointing methods into two different categories glass-box approaches and black-box approaches. In [1] Baader et al. introduce a glassbox approach for axiom pinpointing. In this paper, we focus on a black-box approach inspired by the work of Kalyanpur et. al. [6].

Interpreting images and extraction of deep-level semantics, can not be done sufficiently only on low-level features. Images often show scenes illustrating abstract concepts like events. To perceive such an event concept, additional background knowledge about the whole scene is required. In [11] Möller et .al introduced abduction as a new inferencing service on Description Logic $\mathcal{A}$-Boxes that enables able to reason from effects (observations/features) to causes (explanations/semantics). In contrast to the approach of Möller where new knowledge is extracted through abduction, our approach focuses on verification of knowledge against a specific model. The approach presented in [2] aims to enhance the semantic image description with the use of fuzzy Description Logics. Based on fuzzy Description Logic knowledge bases specialized reasoning services are used to, e.g. solve inconsistencies resulting from the classification process or extract implicit semantics but all these approaches suffer from the particularities coming with the use of fuzzy Description Logics.

6 Conclusions

We have introduced Weighted Max-DL-SAT as a service for modeling and solving problems with inconsistencies and uncertainty using Description Logics. A core feature of our approach is the ability to handle uncertainty similar to fuzzy or probabilistic Description Logics whereas inconsistency is handled like in a crisp Description Logics manner. This combination of features is useful to many different problems, like ontology learning, semantic information extraction or image labeling. The evaluation on image labeling indicates that we achieve a slightly improvement of the results compared to a classifier based solely on low-level features. Compared to a highly specialized method Weighted Max-DL-SAT looses a bit of accuracy but this was the expected price for the gain of generality of the method. With optimized parameters used for the knowledge extraction and a adjusted modeling, we expect further improvements.

In our future work, we will concentrate on the improvement of the performance of maximal consistent sub-ontology calculation. Our experience has shown that it could be promising to improve the MISO calcualtions in this context. The multiple outlier observed in our results showed us that it might be useful to consider approaches other than out WHST based black box method. On this account, we work on an glass box approach to be able to compare it to our black box method. Some of our results also indicate that the consistency checking in approach is a large cost factor, so the integration of Description Logic approximation techniques, like in [9] could be promising. In the next implementations, we will focus on these three promising optimization strategies for Weighted Max-DL-SAT. In addition, we will apply Weighted Max-DL-SAT to different other interesting problems.

References

1. Baader, F., Pealoza, R.: Axiom pinpointing in general tableaux. J. Log. Comput. 20(1), 5–34 (2010), `\url{http://dblp.uni-trier.de/db/journals/logcom/logcom20.html#BaaderP10}`
2. Dasiopoulou, S., Kompatsiaris, I., Strintzis, M.: Investigating fuzzy DLs-based reasoning in semantic image analysis. Multimedia Tools and Applications 49(1), 167–194 (2010)
3. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) Proceedings of the 4th International Semantic Web Conference (ISWC'05). LNCS, vol. 3729, pp. 353–367. Springer, Galway, Ireland (November, 6–10 2005)
4. Haase, P., Völker, J.: Ontology learning and reasoning – dealing with uncertainty and inconsistency. In: Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW) (November 2005)
5. Heinsohn, J.: Probabilistic description logics. In: UAI. pp. 311–318 (1994)
6. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of owl dl entailments. In: ISWC/ASWC. pp. 267–280 (2007)
7. Koller, D., Levy, A., Pfeffer, A.: P-classic: A tractable probabilistic description logic. In: Proceedings of AAAI-97. pp. 390–397 (1997)
8. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence 172(6-7), 852 – 883 (2008), `http://www.sciencedirect.com/science/article/B6TYF-4R1MF4C-1/2/36132fd965af5a169b53a197616f4721`
9. Pan, J.Z., Thomas, E.: Approximating owl-dl ontologies. In: AAAI. pp. 1434–1439. AAAI Press (2007), `http://dblp.uni-trier.de/db/conf/aaai/aaai2007.html#PanT07`
10. Papadopoulos, G.T., Saathoff, C., Grzegorzek, M., Mezaris, V., Kompatsiaris, Y., Staab, S., Strintzis, M.G.: Comparative evaluation of spatial context techniques for semantic image analysis. In: Proceedings of 10th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS) (2009), `http://kodemaniak.de/publications/wiamis2009_submission_81.pdf`
11. Peraldi, I.S.E., Kaya, A., Mller, R.: Formalizing multimedia interpretation based on abduction over description logic aboxes. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009), `http://dblp.uni-trier.de/db/conf/dlog/dlog2009.html#PeraldiKM09`
12. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. 32(1), 57–95 (1987), `http://dblp.uni-trier.de/db/journals/ai/ai32.html#Reiter87`
13. Saathoff, C., Grzegorzek, M., Staab, S.: Labelling image regions using wavelet features and spatial prototypes. In: Semantic Multimedia, Third International Conference on Semantic and Digital Media Technologies, SAMT 2008, Koblenz, Germany. pp. 89–104 (2008), `http://www.uni-koblenz.de/~saathoff/publications/samt08.pdf`
14. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: ESWC. pp. 226–240 (2005)
15. Straccia, U.: A fuzzy description logic. In: Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence. Madison, US (1998), `citeseer.nj.nec.com/straccia98fuzzy.html`
16. Straccia, U.: Towards a fuzzy description logic for the semantic web (preliminary report). In: 2nd European Semantic Web Conference (ESWC-05). pp. 167–181. No. 3532 in Lecture Notes in Computer Science, Springer Verlag, Crete (2005), `http://faure.iei.pi.cnr.it/~straccia/download/papers/ESWC05/ESWC05.pdf`

Reasoning in resource-constrained environments: a matchmaking engine over relational Knowledge Bases

Eufemia Tinelli[1], Francesco M. Donini[2], Michele Ruta[1], and Eugenio Di Sciascio[1]

[1] Politecnico di Bari, via Re David 200, I-70125, Bari, Italy
{e.tinelli,m.ruta,disciascio}@poliba.it
[2] Università della Tuscia, via S. Carlo 32, I-01100, Viterbo, Italy
donini@unitus.it

Abstract. We present a framework for logic-based matchmaking on ALN ABoxes stored in a relational database. The proposed approach allows both non-standard reasoning and subsumption check be performed only via standard SQL queries. Main contribution is in the SQL implementation of the following features: (i) compliance with four match classes (*i.e.*, exact, full, partial and potential); (ii) rank computation for each matching outcome and (iii) preferences management in the user query. Performance evaluation carried out on a PostgreSQL 8.4 engine reports reasonable results in terms of scalability and turnaround times for large scale data sets.

1 Introduction

Benefits introduced by semantic technologies are well-known in a number of frameworks where simplistic keyword-based searches are not enough. Inference services, both standard and non-standard [11], allow to match requests and resources based on the actual meaning of their descriptions and –more interesting– to provide classification and logic-based ranking. Beyond obviously good matches, such as *exact* or *full* ones, we deem so called *potential or intersection* matches (where requests and supplied resources have something in common and no conflicting characteristics) as more interesting and useful from the user perspective. *Partial or disjoint* matches (where requests and supplies have some conflicting features) can also be considered worthwhile in all scenarios when nothing better exists. In those cases, one can be interested in understanding the conflict degree between perspective matching descriptions. What usually prevents a widespread usage of semantic approaches is that they require heavy computational capabilities, and response times are often unacceptable in common applications as soon as real (or realistic) data sets have to be faced. Furthermore, current systems usually allow a requester only to express her mandatory requirements and there is no possibility to grade user preferences in a more fine grained way. The problem of finding efficient reasoning strategies has been widely studied (see [8, 20, 14] among others). Basically, Knowledge Compilation [7] has been employed for making computationally acceptable the reasoning, splitting query answering in two phases: (i) KB is pre-processed, thus parsing it in a proper data structure (*off-line reasoning*); (ii) the query is answered exploiting the structure coming from the first phase (*on-line reasoning*).

This paper presents an automated matchmaking framework, which exploits Knowledge Representation (KR) and reasoning techniques as well as Description Logics (DLs) formalisms, to retrieve the *best* supplied resources w.r.t. a user request, ranked according to the semantic distance from the request itself. Knowledge Bases (KBs) –stored in a relational database– are used, so that inferences are performed via standard SQL queries. The proposed matchmaker leverages KB pre-processing to reduce on-line reasoning overhead. Relevant provided features include: (i) it copes with several match classes; (ii) it allows to assign a relevance degree to each feature in the user query and (iii) it is able to return a logic-based explanation of the ranking results. The paper presents both the modeling approach allowing to translate a given KB into the reference relational database and the incremental building of SQL sub-queries allowing to matchmake and rank results. An experimental evaluation –using PostgreSQL 8.4 DBMS– has been carried out, showing the effectiveness of the proposal and its scalability. Matchmaker performances have been compared with the ones provided by *MaMaS-tng*[3] reasoner with reference to the same set of non-standard inference services [10].

The remainder of the paper is organized as follows. In the next section, a survey of most significant related work is presented; subsequently, Section 3 introduces the proposed framework and approach and Section 4 reports on a performance evaluation of the implemented approach. Conclusions and future research directions close the paper.

2 Background

Several systems and approaches have been presented in literature, where database technology is used to both persistently store knowledge and make scalable queries on it [5, 18]. They are mainly classified according to the language (*i.e.*, RDF(S) [4] or OWL [5]) they adopt for defining ontologies. In what follows, most relevant frameworks will be surveyed to allow a comparison with the approach we propose here.

Oracle Spatial 11g[6] is the first enterprise-oriented, scalable and reliable data management platform for RDF-based applications. It supports query answering for RDF(S) and OWLPrime. Based on a graph data model, RDF triples are made persistent, indexed and queried, similarly to other object/relational data types. *Owlgres*[7] is a DL-Lite [9] reasoner implementation for PostgreSQL. A distinguishing feature is that, along with standard inferences (*e.g.*, subsumption), it supports conjunctive query answering over ABoxes in a secondary storage (typically an RDBMS) so coping with large datasets. A comparable system using RDBMS to deal with large sets of data is *QuOnto*[8], a DL-Lite reasoner providing consistency check and conjunctive query replying services. Neither QuOnto nor OWLgres return a ranked list of results. Further ontology storage systems –such as *DLDB* [19] and *Sesame on PostgreSQL* [6]– adopt binary tables, one

[3] http://sisinflab.poliba.it/MAMAS-tng/
[4] http://www.w3.org/TR/rdf-primer/
[5] http://www.w3.org/TR/owl2-overview/
[6] http://www.oracle.com/technology/tech/semantic_technologies/index.html
[7] http://pellet.owldl.com/owlgres/
[8] http://www.dis.uniroma1.it/q̃uonto/

for each class in the TBox; whereas SOR (Scalable Ontology Repository) [17] exploits four kinds of tables for managing OWL-Lite constructs: atomic tables (for primitive concepts and properties), TBox axiom tables, ABox fact tables and class constructor tables. But the most popular and recent OWL storage is OWLIM [15]. It is a Sesame plug-in able to add a robust support for the semantics of RDFS, OWL Horst and OWL2 RL. A possible optimization is obtained by caching the classification hierarchy in the database as it is implemented in *Instance Store* (***iS***) [4], an engine for reasoning over OWL KBs specifically adopted in biomedical-informatics. A highly-scalable OWL reasoner is SHER (Scalable Highly Expressive Reasoner) [13] enabling conjunctive query answering. It supports a subset of OWL-DL excluding nominals, and it relies on an indexing technique of ABox instances in the database. SHER embeds Pellet to infer implicit information from indexed data and to obtain explanations for inconsistencies. PelletDB[9] provides an OWL 2 reasoning system specifically built for enterprise semantic applications. It combines Pellet's OWL capabilities and scalable native reasoning of Oracle Database 11g so ensuring performance improvements w.r.t. to the use of such technologies separately. Differently from the previous approaches, the most widespread DL-reasoner, *i.e.*, KAON2[10], does not implement the tableaux calculus, but it reduces a SHIQ(D) knowledge base to a disjunctive datalog program. An inference engine for answering conjunctive queries has been so developed applying well-known deductive database techniques.

All the cited systems, although often allow an expressiveness greater than the one enabled by the engine proposed here, are only able to return either exact matches (*i.e.*, instance retrieval) or query answering. On the contrary, we use an enriched relational schema to provide a logic-based ranked list of results and the possibility to implement a semantic explanation of outcomes.

3 Proposed Approach

Description Logics are the reference formalisms we adopt in this paper. In particular, we refer to (a syntactic variant of) $\mathcal{ALN}$, whose allowed constructs are: conjunction $C \sqcap D$, universal quantification $\forall R.C$, and unqualified number restriction $(\geq nR), (\leq nR)$. A simple terminology $\mathcal{T}$ is hypothesized which contains inclusion axioms $A \sqsubseteq C$, concept definitions $A = C$, and disjointness axioms $A \sqcap B \sqsubseteq \bot$. If both the requested and the supplied resources are expressed in $\mathcal{ALN}$ w.r.t. an ontology $\mathcal{T}$, it is possible to exploit their formal semantics during the classification and matching processes. Recall that (see [12, 16] for further details) given a TBox $\mathcal{T}$, a *match degree* between a request D and a supplied resource C (both expressed w.r.t. $\mathcal{T}$) can be evaluated as:

– **Exact**. All the features requested in $\mathcal{D}$ are exactly provided by $\mathcal{C}$, and vice versa—in formulae, $\mathcal{T} \models \mathcal{D} \Leftrightarrow \mathcal{C}$.

– **Full-Subsumption**. All the features requested in $\mathcal{D}$ are contained in $\mathcal{C}$—in formulae, $\mathcal{T} \models \mathcal{C} \Rightarrow \mathcal{D}$.

– **Potential-Intersection**. There is a nonempty intersection among the features offered in $\mathcal{C}$ and the ones requested in $\mathcal{D}$—in formulae, $\mathcal{T} \not\models \neg(\mathcal{D} \sqcap \mathcal{C})$.

[9] http://clarkparsia.com/pelletdb/

[10] http://kaon2.semanticweb.org/

– Partial-Disjoint. Some features requested in $\mathcal{C}$ are conflicting with some other ones offered in $\mathcal{D}$—in formulae, $\mathcal{T} \models \neg(\mathcal{D} \sqcap \mathcal{C})$.

The proposed approach implements all the above match types. However, it is possible to add further user-oriented match classes via the incremental building of match requests by means of SQL sub-queries. Concepts are normalized according to the *Concept-Centered Normal Form* (CCNF), [1, Ch.2], through the recursive application of the formulas in Figure 1, until no rule is applicable at every nesting level.

TBox reduction	Concept reduction	$\bot$-reduction
	$\forall\rho.(D \sqcap E) \rightarrow \forall\rho.D \sqcap \forall\rho.E$	
$A \rightarrow A \sqcap C$	$(\geq nR) \sqcap (\geq mR) \rightarrow (\geq nR)$	$\forall\rho.(\geq nR) \sqcap \forall\rho.(\leq mR) \rightarrow \forall\rho.\bot$
if $A \sqsubseteq C \in \mathcal{T}$	if $n > m$	if $n > m$
$A \rightarrow C$	$(\leq nR) \sqcap (\leq mR) \rightarrow (\leq nR)$	$\forall\rho.(\forall R.\bot) \sqcap \forall\rho.(\geq nR) \rightarrow \forall\rho.\bot$
if $A = C \in \mathcal{T}$	if $n < m$	$\forall\rho.A \sqcap \forall\rho.B \rightarrow \forall\rho.\bot$
	$\forall R.\bot \rightarrow \leq 0R$	
		where A and B are disjoint concept names, *i.e.*, $A \sqcap B \sqsubseteq \bot \in \mathcal{T}$.

Fig. 1. Rules for CCNF. The symbol ρ is a sequence of role names $\rho = R_1 \cdots R_n$, so that $\forall\rho.C$, means $\forall R_1.(\ldots.(\forall R_n.C)\ldots)$. We include the case $\rho = \varepsilon$ (empty sequence), when $\forall\rho.C$ is just C.

The proposed classification is based on a role-free ABox, where each assertion $C(a)$ means that supply a offers features C. Of course, each individual a is involved in one assertion only, while the same features C could be offered by more than one supply. To store a supply $C(a)$ in a database, we divide a C in four groups of conjuncts $C_n \sqcap C_\sharp \sqcap C_{\forall.n} \sqcap C_{\forall.\sharp}$, being C_n the concept names, $C_\sharp$ the number restrictions, $C_{\forall.n}$ the conjuncts of the form $\forall R_1.(\ldots.(\forall R_n.A)\ldots)$ and $C_{\forall.\sharp}$ the conjuncts of the form $\forall R_1.(\ldots.(\forall R_n.D)\ldots)$ where D is a number restriction.

A proper design of the Entity-Relationship (E-R) model is a fundamental prerequisite to correctly store both ABox instances and all the TBox $\mathcal{T}$axioms to be used in the further reasoning stages. In the provided model: (i) entities are chosen in a way to describe all the basic information elements used in the matchmaking process; (ii) numerical features (*e.g.*, *price* or *quantity*) could be very useful in several scenarios (*e.g.*, e-commerce) but they are not closely related to the semantic description of a resource; anyway as such resource information are structured by definition, they will be more easily managed directly by the DBMS. They are named *structured conditions*. Once a concept C has been put in CCNF, the assertions $C(a)$ will be stored in the database, by assigning identifiers to given elements of the syntactic tree of C, and then linking such identifiers by suitable database relations. The logic model for the database storing conjuncts of the normalized form is reported in Figure 2. As an example, Table RESOURCE stores data related to a given resource whereas Table DL_ASSERTION stores the individual describing a resource along with data expressing both cardinality and type of normalized elements. Tables CONCEPT_NAME, NUMBER_RESTRICTION, UNIV_NAME and UNIV_NUMBER respectively store the conjuncts C_n, $C_\sharp$, $C_{\forall.n}$ and $C_{\forall.\sharp}$ of C. A nesting level will be assigned based on how many $\forall$-quantifiers have a given concept C in their scope. For example, $\forall R.C$ has a nesting level 1, $\forall R.\forall S.A$ has nesting level 2, and so on. The attribute `level` of both Table UNIV_NAME and Table UNIV_NUMBER, refers to the assigned nesting degree. Moreover, the attribute `r_type` allows to dis-

$CONCEPT\_NAME(\underline{id\_name}, name)$

$DISJOINT(\underline{id\_name}, id\_name\_disj)$

$NUMBER\_RESTRICTION(\underline{id\_number}, role, r\_type)$

$UNIV\_NAME(\underline{id\_univ\_name}, role\_list, id\_name, level)$

$UNIV\_NUMBER(\underline{id\_univ\_number}, role\_list, id\_number, level)$

$RESOURCE(\underline{id\_resource}, \cdots structured\_conditions \cdots)$

$DL\_ASSERTION(\underline{id\_assert}, owl, n\_name, n\_number, n\_univ\_name, n\_univ\_number, id\_resource)$

$ASSERT\_CONCEPT\_NAME(\underline{id\_assert, id\_name})$

$ASSERT\_NUMBER\_RESTRICTION(\underline{id\_assert, id\_number}, value)$

$ASSERT\_UNIV\_NAME(\underline{id\_assert, id\_univ\_name})$

$ASSERT\_UNIV\_NUMBER(\underline{id\_assert, id\_univ\_number}, value)$

Fig. 2. DataBase logic model

concept_name	
id_name	*name*
1	A
2	B
...	...

number_restriction		
id_number	*role*	*r_type*
1	R	min
2	R	max
3	T	min
4	T	max
...	...	...

univ_name			
id_univ_name	*role_list*	*id_name*	*level*
1	R.S	2	2
...	...	...	...

univ_number			
id_univ_number	*role_list*	*id_number*	*level*
1	R	4	1
...	...	...	...

assert_concept_name	
id_assert	*id_name*
100	1
...	...

assert_number_restriction		
id_assert	*id_number*	*value*
100	1	3
...	...	...

assert_univ_name	
id_assert	*id_univ_name*
100	1
...	...

assert_univ_restriction		
id_assert	*id_univ_number*	*value*
100	1	6
...	...	...

Fig. 3. Tables filled to store $C(a)$ with $id\_assert = 100$

tinguish numeric restriction cardinalities: $r\_type = max$ (resp. $r\_type = min$) states a $\leq n\ R$ (resp. $\geq n\ R$) restriction. Finally, actual data in individual descriptions are also stored in tables (whose name starts with ASSERT). They link the assertion identifier to its atomic conjuncts storing also numeric values of restrictions for elements in the form $C_\sharp$ and $C_{\forall.\sharp}$. Hence, if the system assigns to $C(a)$ identifier the value 100 and the normalized concept C contains the following conjuncts: A, $\geq 3\ R$, $\forall R.\forall S.B$ and $\forall R. \leq 6\ T$, then the system fills the tables in Figure 3. The presented modeling approach translates an assertion $C(a)$ of size n into $c \cdot n$ database tuples, so it increases the storage size, almost linearly. Nevertheless, such a drawback is largely repaid in terms of flexible match classes management, quick logic-based ranking and explanation of results through enumeration of additional, missing and fulfilled features[11].

3.1 Match classes and ranking function

This subsection reports on queries needed for extracting resources $C_1, C_2, \ldots$ in an exact/full/partial/potential correspondence with a user request D. Queries are incrementally built, according to both number and type of atomic elements composing the

[11] The extraction of conflicting characteristics has not been implemented yet because we do not cache partial matches, exploiting them just as intermediate results.

$$disj(A, B) \tag{1}$$
$$\forall R. \ldots \forall S. \forall T.A \tag{2}$$
$$\forall R. \ldots \forall S. \forall T.B \tag{3}$$
$$\forall R. \ldots \forall S. \exists T \tag{4}$$
$$\forall R. \ldots \exists S \tag{5}$$
$$\ldots \tag{6}$$
$$\exists R \tag{7}$$

Fig. 4. The unsatisfiability pattern in $\mathcal{ALN}$.

description as well as on user constraints. In what follows, we assume that requests D are already in CCNF.

An **Exact match** happens when request and supplied resources are logically equivalent, hence both the so-called structured conditions and all the atomic elements have to correspond, while `n_name`, `n_number`, `n_univ_name` and `n_univ_number` attributes must be equal. In fact, in order to detect an exact match the supply must have exactly the same features of the request and nothing else. As **Full Match** queries simply aim to detect subsumption relationships, we do not deal with them here. On the contrary, we will focus on **Partial** and **Potential Match**, which are strictly related. Actually, a Potential Match is simply a not Partial one. A resource C is a **Potential Match** for a given request D if they do not have conflicting features (*i.e.*, $C \sqcap D \neq \perp$). In case of conflicts, the subset containing not allowed features is the **Partial Match** outcome. The Potential Match results can be obtained by retrieving all the stored supplies excluding Partial Matches. A Partial Match between a resource C and a request D amounts to check whether $C \sqcap D$ is unsatisfiable and why, and such a test in $\mathcal{ALN}$ amounts to check the presence in $C \sqcap D$ of the pattern outlined in Figure 4. There $disj(A, B)$ denotes either two disjoint names, or two incompatible number restrictions, and $\exists R$ denotes a concept in the form $(\geq\ n\ R)$ for some $n > 0$. For roles S, T ans so on the same conditions hold. However, in the proposed approach, such a pattern is split between the database tuples representing $C(a)$, and the SQL query Q_D representing D. Intuitively, for every subconcept of D in the form (2), Q_D looks in the DB for tuples representing those subconcepts of C in the form (3)–(7) which are *not* already in D. Since the selection of the correct pattern to search is leaded by D, the worst case is represented by a request D containing a subconcept C in the form (2) with a role depth n whereas no other subconcept in the form (3)–(7) belonging to the same C pattern is in D. In this case, the $n + 1$ missing subconcepts, required to determine an unsatisfiability pattern for C, have to be looked up in the DB. In particular, one SQL WHERE condition is built in Q_D for each subconcept to search.

To better clarify user request translation into the SQL standard query, a toy example of worst case search, is briefly reported, in accordance with the pattern in Figure 4. Let us suppose a normalized request D - $\forall R.\forall S.A$ ($n = 2$) and two normalized supplies: C_1 - $\forall R.\forall S.B \sqcap \forall R.(\geq\ 1\ S) \sqcap (\geq\ 1\ S)$, C_2 - $(\geq\ 1\ S)$. In order to retrieve a potential match, we have to detect the partial matches *i.e.*, instances represented by tuples in the form (3)–(7), and to discard them from the final results set. As above mentioned, three

WHERE conditions are needed. The SQL query retrieving partial matches w.r.t. D is reported hereafter:

```
SELECT id_assert
FROM assert_univ_name A NATURAL JOIN univ_name
WHERE (level=2 AND role='R.S' AND id_name IN (SELECT id_name_disj
                                               FROM disjoint NATURAL JOIN concept_name
                                               WHERE name='A'))
AND (EXISTS(SELECT *
            FROM (assert_univ_number NATURAL JOIN univ_number)
             NATURAL JOIN number_restriction
            WHERE id_assert=A.id_assert
            AND role_list='R' AND role='S' AND r_type='min' AND value>=1))
AND (EXISTS(SELECT *
            FROM assert_number_restriction NATURAL JOIN number_restriction
            WHERE id_assert=A.id_assert
            AND role='R' AND r_type='min' AND value>=1))
```

Since the previous query returns the supply C_1, the potential matches set is only composed by supply C_2. Moreover, C_2 has $\forall R.\forall S.A$ as missing features (*explanation process*) and a rank equal to 0 as explained in the following (*ranking process*). For the Potential Match results, the logic-based ranking is obtained implementing the ranking function in [10] by aggregating tables with match results. The basic idea is to compute the semantic distance between the normalized forms of both the user request D and the retrieved supply C. To this purpose we introduce 4 tables named CONCEPT_NAME_SCORE, NUMBER_RESTRICTION_SCORE, UNIV_NAME_SCORE and UNIV_NUMBER_SCORE corresponding to the structure of tables CONCEPT_NAME, NUMBER_RESTRICTION, UNIV_NAME and UNIV_NUMBER respectively, enhanced by the attribute `score`. In fact, they store D features with the related user preference (a value between 1 and 5) and, if the user does not set scores for requested features, the matchmaker considers the default value 1. In particular, the results ranking is calculated via the formula (1) rank=(no. fulfilled features of C)/(no. features of D) in case no scores have been set and, as preliminary investigation, via the formula (2) rank=(score sum for fulfilled C features)/(scores sum for D features) otherwise.

4 System and Performance Evaluation

The proposed matchmaker acts as a Java application. A prototypical testing GUI has been developed in order to enable users: 1) to edit/import the request directly in OWL or in DIG [3] (which is more compact); 2) to weigh each normalized concept in the request; 3) to choose the match class to search for and 4) to show the ranked list of results. Experiments have been carried out exploiting an Intel Core i3 PC, equipped with 4 GB RAM. System evaluation goals were: (i) *approach outcome and scalability* –even if existing OWL benchmarks allow a comprehensive evaluation of most common reasoner capabilities [22, 21], unfortunately none is able to execute non-standard services we refer here. Hence, in order to evaluate both matchmaker correctness and performance, only a strict comparison with MaMas-tng results can be carried out; (ii) *data complexity* –a given query is chosen and the system behavior has been evaluated as a function of dataset size; (iii) *expression complexity* –a given dataset is chosen and the system behavior has been evaluated as a function of the execution time of arbitrarily selected queries.

Dataset. In accordance with the goals and assumptions in Section 3, we will use two different domain ontologies: 1) the "Clothing" one (composed by 157 classes and 18 roles) and 2) the "Hotel" one (composed by 68 classes and 12 roles). The former has many concept names whereas the latter has many concept descriptions. Following the "Hotel" ontology structure, it is possible to define individuals with roles nesting level generally higher than the ones of the "Clothing" ontology. Moreover, we have implemented a synthetic KB instances generator, able to automatically build satisfiable instances referred to a given ontology. In this way, we can build data sets having different size, ranging from 100 to 10000 individuals, and instances with a given structure (*i.e.*, number of concept names, number of restrictions, etc.). Finally, several queries have been defined for each knowledge domain. Due to lack of space, we only report on the retrieval times for two queries of average expressiveness respectively referred to the "Clothing" and the "Hotel" ontology:
Q_1 - *"I'm looking for a medium size bluejeans with five pockets and a casual style suitable for spring climate, for both young and adult people"* classified as `n_name=5`, `n_number=18` and `n_univ_name=10` in its normalized form;
Q_2 - *"I'm looking for a twin bed room with some included options (specifically, air conditioning and high speed Internet connection) in a four star hotel near Termini Station in Rome"* classified as `n_name=1`, `n_number=3`, `n_univ_name=10` and `n_univ_number=4` in its normalized form.

Data and expression complexity. The application has been tested by means of several queries with different expressiveness applied to several data sets in order to obtain a comprehensive evaluation of the approach. Our tests measure the retrieval time calculated as average time over ten repetitions. Tests have been performed composing both requests with few generic features and requests including more features with an higher specificity (*e.g.*, similar to the previous ones). Results show that retrieval times moderately increase addressing to the system more complex queries. For this reason, Figure 5 only reports on retrieval times for the requests Q_1 and Q_2. Times have been computed also considering the request normalization process. From the performance comparison standpoint, MaMaS-tng reached via its DIG interface based on HTTP Post has been compared with our relational knowledge based matchmaker running on a remote PostgreSQL server. All tests are reported in Figure 5. Note that the retrieval time difference –given the same instance number for the ontologies– is due to the different complexity of them, as said before.

Moreover, tests have proved that retrieval time of Potential Match (with and without ranking) are higher than the ones of the other match classes (as expected) whereas Exact Match and Full Match have comparable retrieval times. In fact, Potential Match requires a more complex structure of SQL sub-queries and it deals with a higher number of intermediate results (*i.e.*, tuples). Retrieval times for "Clothing" dataset of 10000 instances are justified by the presence of potential matches only by construction. Basically, it can be concluded that retrieval times linearly increase with the data size, in case of up to 5000 individuals more or less. Such outcomes are justified by the higher number of returned instances when datasets increase and –on the other hand– they suggest a proper table partition of the database is needed. The approach scalability is proved by the comparison with retrieval times produced by MaMaS-tng reasoner. In particular, our

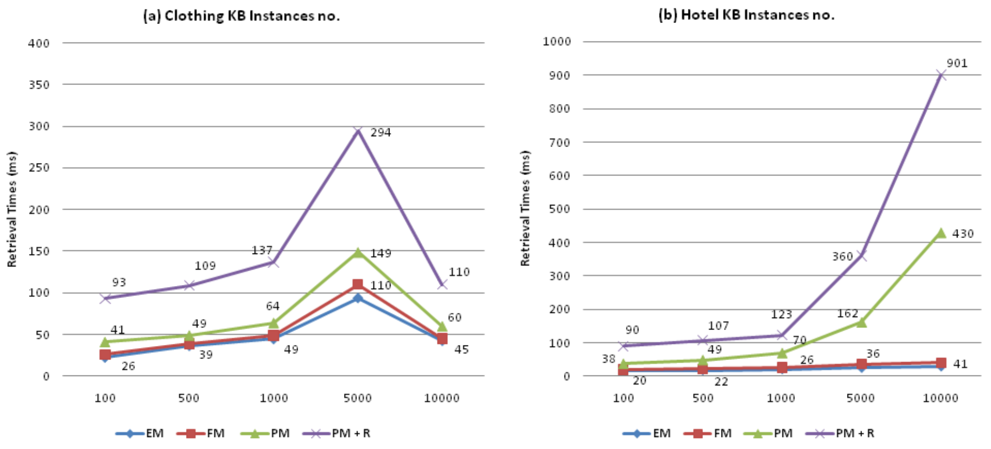


Fig. 5. Proposed system retrieval times (in ms) - [E,F,P]M=[Exact, Full, Potential] Match

higher retrieval time (*i.e.*, Ranked Potential Match - PM+R) as been used as baseline for the further comparison with MaMaS-tng.

Approach outcome. As said, MaMas-tng has been used as comparison term to evaluate output correctness. Results show that the matchmaker proposed here retrieves the same ranked list of results for each match class. The ranking assigned to each potential result has been computed both by MaMaS-tng (using $rankPotential$ [12] algorithm) and by the proposed system (using the default values for the request features weights). Best results for MaMaS-tng have a semantic distance w.r.t. the request equal to 0. So for a significant comparison, we have re-computed the previous ranking formula as: $rank\_value = num_D - num_C$, where num_D refers to request features whereas num_C sums supply features matching the requested ones. Table 1 reports on MaMaS-tng performance on the same datasets and the same queries used for results in Figure 5. Given a request D and a supplied resource C, MaMaS-tng allows to determine the match type ($matchType(D, C)$) –see ask $mT(D, C)$ in Table 1– and to calculate a ranking value ($rank(D, C)$) –see ask $r(D, C)$ in Table 1. It does not provide functions to retrieve all the individuals satisfying a requested match class as implemented in the matchmaker proposed here. So, in order to compare the matchmakers performance, it has been considered the ranked potential match computation, which corresponds to the previous two *asks* for MaMaS-tng (see Table 1 for details).

Table 1. MaMaS-tng retrieval times (in ms) for both "Clothing" and "Hotel" ontologies

PM+R=Potential Match and Ranking

Clothing	**100**	**500**	**1000**	**5000**	-	**Hotel**	**100**	**500**	**1000**	**5000**
PM + R	*93*	*109*	*137*	*294*	-	*PM + R*	*90*	*107*	*123*	*360*
r(D,C)	19771	112934	265811	N/A	-	r(D,C)	11624	54434	106347	1205115
mT(D,C)	20488	115811	269208	N/A	-	mT(D,C)	23040	101258	219400	2382376

Basically, a shallow examination of results shows highest loading times obtained with the proposed matchmaking approach. Nevertheless, it has to be noticed –as mentioned in Section 3– that the proposed approach includes a time-consuming pre-processing phase. So, knowledge bases loading times are obviously higher than in case of MaMaS-tng (see Table 2 where $M - tng$ column refers to MaMaS-tng and DB one is about our approach). Anyway, the KB loading is an off-line and *una tantum* process, performed once when the system is set and not repeated during reasoning phases. Moreover, if the TBox has not been modified then it is possible to store incrementally only new instances, drastically reducing load times. It has to be also said that, MaMaS-tng is not able to load large KBs (*i.e.*, for “Clothing” ontology, previewed 5000 ABox instances cannot be uploaded).

Table 2. Knowledge bases loading times (in ms) for both “Clothing” and “Hotel” ontologies

Clothing	**100**	**500**	**1000**	**5000**	–	**Hotel**	**100**	**500**	**1000**	**5000**
M-tng	995	4057	9599	N/A	–	M-tng	529	2858	5553	239775
DB	77605	410819	856286	4213688	–	DB	52358	333520	601448	3409431

5 Conclusion and Future Work

Motivated by the need to efficiently cope with large datasets in semantic matchmaking, we presented a logic-based framework exploiting a flexible knowledge modeling. A user request is structured as set of normalized features also weighted according to the relevance assigned by the user. By exploiting only SQL queries, the system is able to detect resources falling in several match classes also ranking results. Current implementation refers to $\mathcal{ALN}$, although as pointed out in [2] renewed interests in light-weight DLs for large ontologies and non-standard services has been observed, in order to successfully use semantic technologies in real-world applications.

Preliminary performance evaluation on various datasets show an efficient behavior also considering that optimization techniques such as the transitive closure modeling and the implementation of table partitioning have not been implemented yet. Future work aims at testing further devised strategies for score calculation along with a full optimization of the database and at evaluating performance with other existing OWL-DL storage engines with reference to comparable match classes, *i.e.*, exact and full.

6 Acknowledgments

The authors acknowledge partial support of Apulia Region Strategic Project PS_125 and the reviewers for useful comments and suggestions.

References

1. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook, 2nd edition. Cambridge University Press (2007)

2. Baader, F.: What's new in description logics. Informatik-Spektrum pp. 1–9 (2011), 10.1007/s00287-011-0534-y
3. Bechhofer, S., Möller, R., Crowther, P.: The DIG Description Logic Interface. In: DL'03. CEUR Workshop Proceedings, vol. 81 (2003)
4. Bechhofer, S., Horrocks, I., Turi, D.: The OWL Instance Store: System Description. In: CADE '05. pp. 177–181 (2005)
5. Bock, J., Haase, P., Ji, Q., Volz, R.: Benchmarking OWL Reasoners. In: ARea Workshop at ESWC 2008. CEUR-WS, Vol 350 (2008)
6. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: ISWC '02. pp. 54–68 (2002)
7. Cadoli, M., Donini, F.M.: A survey on knowledge compilation. AI Commun. 10(3-4), 137–150 (1997)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data Complexity of Query Answering in Description Logics. In: KR-06. pp. 260–270 (2006)
9. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J. of Automated Reasoning 39(3), 385–429 (2007)
10. Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., Tinelli, E.: A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces. Int. J. on Electronic Commerce 12(2), 127–154 (2007)
11. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach. J. of Artificial Intelligence Research 29, 269–307 (2007)
12. Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: A System for Principled Matchmaking in an Electronic Marketplace. Int. J. on Electronic Commerce 8(4), 9–37 (2004)
13. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Efficient Reasoning on Large SHIN Aboxes in Relational Databases. In: SSWS '09. pp. 110–124 (2009)
14. Doyle, J., Patil, R.S.: Two Theses of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the Utility of Representation Services. Artificial Intelligence 48(3), 261–297 (1991)
15. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM - A Pragmatic Semantic Repository for OWL. In: WISE. vol. 3807, pp. 182–192. Springer (2005)
16. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. Int. J. on Electronic Commerce 8(4) (2004)
17. Lu, J., Ma, L., Zhang, L., Brunner, J.S., Wang, C., Pan, Y., Yu, Y.: SOR: a Practical System for Ontology Storage, Reasoning and Search. In: VLDB '07. pp. 1402–1405. VLDB Endowment (2007)
18. del Mar Roldan-Garcia, M., Aldana-Montes, J.F.: A Survey on Disk Oriented Querying and Reasoning on the Semantic Web. In: ICDEW'06. pp. 58–65. IEEE Computer Society (2006)
19. Pan, Z., Heflin, J.: DLDB: Extending Relational Databases to Support Semantic Web Queries. In: PSSS1. vol. 89, pp. 109–113. CEUR-WS.org (2003)
20. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. 74(2), 249–310 (1995)
21. Thakker, D., Osman, T., Gohil, S., Lakin, P.: A pragmatic approach to semantic repositories benchmarking. In: The Semantic Web: Research and Applications, vol. 6088, pp. 379–393. Springer (2010)
22. Weithner, T., Liebig, T., Luther, M., Bhm, S.: Whats Wrong with OWL Benchmarks. In: SSWS 2006. pp. 101–114 (2006)

Paraconsistent Rough Description Logic

Henrique Viana[*], João Alcântara[**] and Ana Teresa Martins[***]

Departamento de Computação, Universidade Federal do Ceará, P.O.Box 12166,
Fortaleza, CE, Brasil 60455-760

Abstract. In this paper, we introduce a paraconsistent extension of Rough Description Logics which allows the representation of incomplete and contradictory concepts, as well as the lower and upper approximations of these kinds of concepts. Furthermore, we use the notions of approximations, which can be applied successively, to make restrictions or relaxations in the concept queries with the objective of decreasing or increasing the number of results of the queries, respectively.

1 Introduction

In many applications of Artificial Intelligence not rarely query tasks result in empty answers. Similarly, it may happen that a query has too many answers as a result, and it is not always that all these answers are important. Some approaches, known as query refinement used to deal with uncertain knowledge, develop methods to settle these problems. One of these approaches is the Rough Set theory introduced by Z. Pawlak [13]. Rough Set theory may be applied for query refinement by resorting to query restriction or query relaxation. A query can be restricted in order to obtain only the necessary results, or it can be relaxed, aiming at increasing the number of its results. In this paper, we will focus on the definition of a rough set framework in Description Logics (DLs) fine-tuned to deal with query refinement and incomplete and contradictory information.

Rough Description Logics (RDLs) [5,9,10,14] have introduced a mechanism that allows modeling uncertain reasoning by means of approximations of concepts. RDLs extend classical DLs with two operations, the lower and the upper approximation. Both approximations are based on capturing uncertainty as an indiscernibility relation R, an equivalence relation (i.e., reflexive, symmetric and transitive). We can formally define the upper approximation of a concept as the set of individuals that are indiscernible from at least one that is known to belong to the concept:

$$\overline{C} = \{a \mid \exists b\ (a,b) \in R \wedge b \in C\}.$$

Similarly, one can define the lower approximation of a concept as the set of individuals which for all ones that are indiscernible from, it is known that these ones belong to the concept:

$$\underline{C} = \{a \mid \forall b\ (a,b) \in R \rightarrow b \in C\}.$$

[*] This research is partially supported by CAPES (PROPAG).
[**] This research is partially supported by CAPES (PROCAD).
[***] This research is partially supported by CAPES (PROCAD) and CNPq (PQ, Universal 2010).

Extending the ideas of rough set theory, in order to represent incomplete and contradictory information, the work presented in [16] introduced the notion of paraconsistent rough sets. Instead of employing approximations to classical sets, it was taken into consideration four-valued sets, which are intended to represent incomplete and contradictory information. Furthermore, a similarity relation (i.e., a sort of "indiscernibility" relation that is at least reflexive) is used instead of an equivalence relation, with the aim of modeling different levels of uncertainty.

In this work, we will adapt the notions of paraconsistent rough sets to DL, introducing a paraconsistent RDL, called $PR_{\mathcal{ALC}}$. This paraconsistent RDL is slightly different from the well-known paraconsistent DL presented in [11]. Furthermore, we introduce two similarity relations to deal with approximation of concepts and apply the notion of contextual approximation [5] in our approach as an alternative form of approximation of concepts. Finally, we present some reasoning tasks, related to query refinement, which can be applied with these introduced operations.

The paper is structured as follows. In Section 2, we present the notions of the four-valued calculus, sets and approximations defined in [16]. In Section 3, we introduce $PR_{\mathcal{ALC}}$, and we apply the contextual approximation to $PR_{\mathcal{ALC}}$ together with the similarity relations. In Section 4, we present some query tasks that can be used in $PR_{\mathcal{ALC}}$; in particular, the tight and loose approximations. Finally, in Section 5 we conclude the paper.

2 Paraconsistent Rough Sets

In order to represent incomplete and contradictory information, it was introduced a rough set framework taking into account four-valued sets, instead of elementary sets [16]. In these four-valued sets, an element may belong to a given set, or it may not belong to the set, or its membership in the set may be unknown due to incomplete information, or even inconsistent due to a contradictory evidence. Under this view, membership functions, set containment and set operations are also four-valued, where logical values are **t** (*true*), **f** (*false*), **i** (*inconsistent*) and **u** (*unknown*). Moreover, since the knowledge at hand is incomplete, instead of indiscernibility relations, the authors resort to similarity relations to group together elements that are close to each other. Consequently, the notions of upper and lower approximations extend the usual definitions of approximations in the rough set theory.

2.1 Four-valued Semantics

In [16], the language for the four truth values was adapted from Belnap's Logic [1], which is grounded on bilattices [6]. As it is known, in bilattices, two orderings of truth values are considered: *truth ordering* $(\leq_t)$ and *knowledge ordering* $(\leq_k)$. However, in [16], the construction of the language is slightly different from that employed by Belnap. The change is motivated by some results accounted in [12] as counterintuitive in Belnap's *truth ordering*. In order to give more details, let us regard the following example involving test of cars:

Example 1 Suppose that we have two cars a and b, belonging to the same man, where *Safe*$(a) = \mathbf{u}$ and *Safe*$(b) = \mathbf{i}$. We want to know if all of his cars are safe, i.e., *Safe*$(a) \wedge$ *Safe*(b), where $\wedge$ stands for the meet operation with respect to $\leq_t$ in Belnap's Logic. We have that $(\mathbf{u} \wedge \mathbf{i}) = \mathbf{f}$. However, in this case we do not know whether both cars are safe because we do not have any information about the safety of car a. In contrast to the answer obtained in Belnap's logic, **u** seems to be a more intuitive answer to this case. Similarly, if we want to check if the man has a safe car, i.e., *Safe*$(a) \vee$ *Safe*(b), Belnap's Logic results in the value **t**. This differs from our intuition because we know that the safety of car b is unclear, since the results of the tests are contradictory, and we know nothing about the safety of car a. In such a case, **i** seems to be a more intuitive answer.

In [16], in order to overcome these unexpected results, they redefined $\leq_t$ as the smallest reflexive and transitive relation satisfying $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{i} \leq_t \mathbf{t}$. Consequently, the operations of meet ($\wedge$) and join ($\vee$) in the *truth ordering* are defined as the greatest lower bound and as the least upper bound in this new ordering, respectively, i.e., $(x \wedge y) = \text{GLB}^t\{x, y\}$ and $(x \vee y) = \text{LUB}^t\{x, y\}$. The truth table for the meet and join operations in $\leq_t$ as well as to the negation and implication can be seen in Table 1.

$\wedge$	**f**	**u**	**i**	**t**
f	**f**	**f**	**f**	**f**
u	**f**	**u**	**u**	**u**
i	**f**	**u**	**i**	**i**
t	**f**	**u**	**i**	**t**

$\vee$	**f**	**u**	**i**	**t**
f	**f**	**u**	**i**	**t**
u	**u**	**u**	**i**	**t**
i	**i**	**i**	**i**	**t**
t	**t**	**t**	**t**	**t**

$\hookrightarrow$	**f**	**u**	**i**	**t**
f	**t**	**t**	**t**	**t**
u	**u**	**u**	**i**	**t**
i	**i**	**i**	**i**	**t**
t	**f**	**u**	**i**	**t**

$\neg$	
f	**t**
u	**u**
i	**i**
t	**f**

Table 1. Truth tables for $\wedge, \vee, \hookrightarrow$ and $\neg$

The implication $\hookrightarrow$ naturally extends the usual two-valued implication, which can be defined as $(\neg x \vee y)$. Consequently, the implication has the following property: $(x \hookrightarrow y) \equiv (\neg y \hookrightarrow \neg x)$, but it does not satisfy the Modus Ponens if we assume that $\{\mathbf{t}, \mathbf{i}\}$ is the set of designated values. For instance, $(\mathbf{i} \hookrightarrow \mathbf{f}) \wedge \mathbf{i}$ does not result in **f**. A more detailed explanation of the definition of $\hookrightarrow$ can be found in [16]. Finally, the semantics of $\forall$ and $\exists$ is given by

$$\forall x P(x) = \underset{x \in U}{\text{GLB}^t}\{P(x)\} \text{ and } \exists x P(x) = \underset{x \in U}{\text{LUB}^t}\{P(x)\},$$

where U is the universe set and $P(x)$ denotes that x has the property P, which is evaluated to one of the four truth values.

2.2 Four-valued Sets

Now, we present the notion of a four-valued set. Given the disjoint sets U and $\neg U$, where $\neg U = \{\neg x \mid x \in U\}$, a four-valued set A on U is defined as any subset of $U \cup \neg U$. Intuitively, $x \in A$ represents that there is an evidence that x is in A, and $(\neg x) \in A$ represents that there is an evidence that x is not in A. We assume that $\neg(\neg x)$ is equal to x. In this framework, set membership is four-valued and it extends the usual two-valued membership.

Set membership, denoted as $\bar{\in} : U \times 2^{U \cup \neg U} \rightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{i}, \mathbf{u}\}$, is defined by

$$x\bar{\in}A = \begin{cases} \textbf{t}, \text{ if } x \in A, (\neg x) \notin A, \\ \textbf{f}, \text{ if } x \notin A, (\neg x) \in A, \\ \textbf{i}, \text{ if } x \in A, (\neg x) \in A, \\ \textbf{u}, \text{ if } x \notin A, (\neg x) \notin A. \end{cases}$$

The complement $\neg A$ of a four-valued set A is defined by $\neg A = \{\neg x \mid x \in A\}$, and the four-valued set inclusion is defined by $X \Subset Y = \forall x \in U(x\bar{\in}X \hookrightarrow x\bar{\in}Y)$. The four-valued intersection and union are defined as $x\bar{\in}(X \Cap Y) = (x\bar{\in}X) \wedge (x\bar{\in}Y)$ and $x\bar{\in}(X \Cup Y) = (x\bar{\in}X) \vee (x\bar{\in}Y)$, respectively.

A four-valued extension of rough sets is, then, defined in [16] by four-valued set approximations as follows. First, the equivalence relation is replaced by a similarity relation, which is also four-valued. A similarity relation extends the indiscernibility relation by gathering in the same class objects which are not necessarily indiscernible, but sufficiently close or similar. In other words, a similarity relation is constructed from the indiscernibility relation by relaxing the original conditions for indiscernibility. This relaxation can be performed in many ways, thus giving many possible definitions for similarity.

Definition 1 (Four-valued Similarity Relation) *[16] By a four-valued similarity relation σ we mean any four-valued binary relation on a universe set U satisfying at least the reflexivity condition, i.e., for any element x of the universe $(x, x)\bar{\in}\sigma =$ **t**. By the neighbourhood of an element $x \in U$ w.r.t. σ, we understand the four-valued set $\sigma(x)$ such that $y\bar{\in}\sigma(x) = (x, y)\bar{\in}\sigma$.*

The membership of an element x in the lower approximation of a four-valued set A is determined by verifying the set inclusion of its neighbourhood $\sigma(x)$ in A. The membership of an element x in the upper approximation is determined by computing the greatest membership value that an element of the universe may have in the intersection of $\sigma(x)$ and A.

Definition 2 (Lower/Upper Approximation) *[16] Let A be a four-valued set. Then, the lower and upper approximations of A w.r.t. the similarity relation σ, denoted by A^{+}_{σ} and $A^{\oplus}_{\sigma}$, respectively, are defined by*

$$x\bar{\in}A^{+}_{\sigma} = \sigma(x) \Subset A \textit{ and } x\bar{\in}A^{\oplus}_{\sigma} = \exists y \in U[y\bar{\in}(\sigma(x) \Cap A)].$$

3 Paraconsistent Rough Description Logic

Rough Description Logics (RDLs) [5,9,10,14] have introduced a complementary mechanism that allows modelling uncertain knowledge by means of approximations of concepts. RDLs extend classical DLs with two modal-like operations, the lower and the upper approximation. In the spirit of rough set theory, two concepts approximate an underspecified (uncertain) concept C as particular sub- and super-concepts, describing which elements are definitely and possibly, respectively, elements of C.

In this section, taking into consideration the approach presented in [16], we extend the RDLs formalisms to paraconsistent rough sets by introducing a four-valued DL general enough to encompass two kinds of similarity relations: those explicitly defined and those defined in terms of a context. One distinctive aspect of our formalism is that it permits successive refinements of a query (see Section 4). In the sequel, we introduce the syntax, the semantics and reasoning tasks for this DL.

3.1 Paraconsistent Rough Description Logic $\mathcal{ALC}$

We introduce a paraconsistent rough extension of the description logic $\mathcal{ALC}$, called $PR_{\mathcal{ALC}}$.

Definition 3 (Alphabet) *The $PR_{\mathcal{ALC}}$ alphabet is composed of the symbols $\neg, \sqcap, \sqcup, \exists$, $\forall, \top, \bot, \underline{\cdot}, \overline{\cdot}$ and three disjoint sets: the set of individuals N_I, the set of concept names N_C, the set of role names N_R and the set of similarity relations N_S.*

Definition 4 (Concepts) *Concepts in $PR_{\mathcal{ALC}}$ are defined by the syntax rules below, where C and D are concepts, A is an atomic concept, R is an atomic role and S is a similarity relation:*

$$C, D \longrightarrow\ A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \overline{C}^S \mid \underline{C}_S$$

Definition 5 (Semantics) *The semantics of $PR_{\mathcal{ALC}}$ individuals, atomic concepts, atomic roles and similarity relations is given by an interpretation $I = (\Delta^I, \cdot^I)$, where the domain Δ^I is a nonempty set of elements and $\cdot^I$ is a mapping function defined as follows: each individual $a \in N_I$ is mapped to $a^I \in \Delta^I$; each atomic concept name $A \in N_C$ is mapped to $A^I : \Delta^I \to \{\boldsymbol{t}, \boldsymbol{f}, \boldsymbol{i}, \boldsymbol{u}\}$; each atomic role name $R \in N_R$ is mapped to $R^I : \Delta^I \times \Delta^I \to \{\boldsymbol{t}, \boldsymbol{f}, \boldsymbol{i}, \boldsymbol{u}\}$; each similarity relation $S \in N_S$ is mapped to $S^I : \Delta^I \times \Delta^I \to \{\boldsymbol{t}, \boldsymbol{f}, \boldsymbol{i}, \boldsymbol{u}\}$ satisfying at least the reflexivity condition, i.e., $S^I(x,x) = \boldsymbol{t}$ for any $x \in \Delta^I$.*

Concepts can be interpreted inductively as follows where, for all $x \in \Delta^I$,

Syntax	Semantics
$\top$	$\top^I(x) = \mathbf{t}$
$\bot$	$\bot^I(x) = \mathbf{f}$
$\neg C$	$(\neg C)^I(x) = \neg(C^I(x))$
$C \sqcap D$	$(C \sqcap D)^I(x) = (C^I(x) \wedge D^I(x))$
$C \sqcup D$	$(C \sqcup D)^I(x) = (C^I(x) \vee D^I(x))$
$\exists R.C$	$(\exists R.C)^I(x) = \underset{y \in \Delta^I}{\text{LUB}^t}(R^I(x,y) \wedge C^I(y))$
$\forall R.C$	$(\forall R.C)^I(x) = \underset{y \in \Delta^I}{\text{GLB}^t}(R^I(x,y) \hookrightarrow C^I(y))$
$\overline{C}^S$	$(\overline{C}^S)^I(x) = \underset{y \in \Delta^I}{\text{LUB}^t}(S^I(x,y) \wedge C^I(y))$
$\underline{C}_S$	$(\underline{C}_S)^I(x) = \underset{y \in \Delta^I}{\text{GLB}^t}(S^I(x,y) \hookrightarrow C^I(y))$

The semantics of concepts is based on semantics of Fuzzy DLs [2], as well as the lower/upper approximations of a concept, which are related with fuzzy rough sets [4]. The main difference from our approach to others based on fuzzy sets is that we consider t-norms, t-conorms, implication functions, and negation functions as the operations of conjunction ($\wedge$), disjunction ($\vee$), implication ($\hookrightarrow$) and negation ($\neg$), respectively, described for the four-valued calculus in [16] (see Table 1). Now, we define the notions of Terminological Box (TBox), Assertional Box (ABox) and ontology in *PR*$_{\mathcal{ALC}}$.

Definition 6 (TBox) *A TBox is a finite set of expressions of the form $C \sqsubseteq D$, the so-called general concept inclusion, where C and D are concepts in PR*$_{\mathcal{ALC}}$.

Definition 7 (ABox) *An ABox consists of a finite set of assertion axioms of the form $C(a)$ and $R(a, b)$, where C is a concept, R is a role, and a and b are individuals in PR*$_{\mathcal{ALC}}$.

When we want to refer to inclusion and assertion axioms indistinctly, we will just call them axioms. The semantics of inclusion and assertion axioms is given by the following table:

Syntax	**Semantics**
$C \sqsubseteq D$	$\mathbf{i} \leq_t \underset{x \in \Delta^I}{\text{GLB}^t}(C^I(x) \hookrightarrow D^I(x))$
$C(a)$	$\mathbf{i} \leq_t C^I(a^I)$
$R(a,b)$	$\mathbf{i} \leq_t R^I(a^I, b^I)$

Note that the semantics of concept inclusion $C \sqsubseteq D$ is derived from the four-valued calculus presented in [16], i.e., it is described w.r.t. implication $\hookrightarrow$, which is defined as $(a \hookrightarrow b) = \neg a \vee b$. We assume that $\{\mathbf{i}, \mathbf{t}\}$ is the set of designated values, therefore $C \sqsubseteq D$ holds iff the result of the implication is **i** or **t**. For assertion axioms the idea is similar: $C(a)$ (resp. $R(a, b)$) holds iff $C(a)$ (resp. $R(a, b)$) is evaluated to one of the designated values.

Definition 8 (Ontology) *An ontology or knowledge base is a set composed by a TBox and an ABox.*

Definition 9 (Satisfiability) *The notion of satisfaction of a set of axioms ε by an interpretation $I = (\Delta^I, \cdot^I)$, denoted $I \models \varepsilon$, is defined as follows: $I \models \varepsilon$ iff I satisfies each element in ε. For an axiom α, if $I \models \alpha$ we say that I is a model of α. I satisfies an ontology O, denoted by $I \models O$, iff I is a model of each axiom of ontology O.*

Definition 10 (Logical Consequence) *An axiom α is a logical consequence of an ontology O, denoted by $O \models \alpha$, iff every model of O satisfies α.*

3.2 Contextual Approximation

In [5], it was introduced the notion of contextual indiscernibility relations in RDLs as a way of defining an equivalence relation based on the indiscernibility criteria. In particular, the notion of context is introduced, allowing the definition of specific equivalence relationships to be used for approximations. The main advantage of this approach is that the reasoning with equivalence classes becomes optimized, since the equivalence relations are discovered in the process of reasoning, differently from the traditional RDLs, where the equivalence relation must be explicitly defined.

In this subsection, we apply the notions of contextual approximation to $\mathit{PR}_{\mathcal{ALC}}$, extending the definitions of lower and upper approximations of a concept. We first present the notion of a context via a collection of $\mathit{PR}_{\mathcal{ALC}}$ concepts. We, then, introduce two specific similarity relations based on such contexts, and finally we define upper and lower approximations of concepts using these new similarities.

Definition 11 (Context) *A context is a finite nonempty set of DL concepts* $\mathcal{C} = \{C_1, \ldots, C_n\}$.

Two individuals a and b are indiscernible with respect to the context $\mathcal{C} = \{C_1, \ldots, C_n\}$ iff for all C_i, where $i \in \{1, \ldots, n\}, C_i^I(a) = C_i^I(b)$. This easily induces an equivalence relation:

Definition 12 (Indiscernibility Relation) *Let* $\mathcal{C} = \{C_1, \ldots, C_n\}$ *be a context. The indiscernibility relation* $R_{\mathcal{C}}$ *induced by* $\mathcal{C}$ *is defined as follows:*

$$R_{\mathcal{C}} = \{(a,b) \in \Delta^I \times \Delta^I \mid \textit{for all } C_i \textit{ where } i \in \{1,\ldots,n\}, C_i^I(a) = C_i^I(b)\}.$$

Since we are dealing with incomplete information, a similarity relation should be more adequate to model relationships between individuals, because it allows to group together individuals that are close to each other, but not necessarily indiscernible. Now, we introduce a specific similarity relation, which is based on the work presented in [8]:

Definition 13 (Similarity Relation - unknown concepts) *Let* $\mathcal{C} = \{C_1, \ldots, C_n\}$ *be a context. The similarity relation* $S_{\mathcal{C}}$ *induced by* $\mathcal{C}$ *is defined as follows:*

$$S_{\mathcal{C}} = \{(a,b) \in \Delta^I \times \Delta^I \mid \textit{for all } C_i \textit{ where } i \in \{1,\ldots,n\}, C_i^I(a) = C_i^I(b) \textit{ or } C_i^I(a) = \boldsymbol{u} \textit{ or } C_i^I(b) = \boldsymbol{u}\}.$$

The purpose of the similarity relation $S_{\mathcal{C}}$ is to approximate incomplete information by considering the truth value **u** as similar to **t**, **f** or **i** and vice versa. In $S_{\mathcal{C}}$ (the *do not care* relation), the relevant information is the only one which is asserted, i.e., to assure an individual has been evaluated to **u** in a concept is regarded as non-relevant. Therefore, an individual a is similar to b in a context $\mathcal{C}$ if for all concepts in $\mathcal{C}$, the interpretation of a and b are equal, or the interpretation of a or b is equal to **u**.

In order to approximate both contradictory and incomplete information, we introduce a second similarity relation, dubbed $P_{\mathcal{C}}$. In this new similarity relation, the truth values **t** and **f** are similar to **i**.

Definition 14 (Similarity Relation - unknown and inconsistent concepts) *Let* $\mathcal{C} = \{C_1, \ldots, C_n\}$ *be a context. The similarity relation* $P_{\mathcal{C}}$ *induced by* $\mathcal{C}$ *is defined as follows:*

$$P_{\mathcal{C}} = \{(a,b) \in \Delta^I \times \Delta^I \mid \textit{for all } C_i \textit{ where } i \in \{1,\ldots,n\}, C_i^I(a) = C_i^I(b) \textit{ or } C_i^I(a) = \boldsymbol{u} \textit{ or } C_i^I(b) = \boldsymbol{u} \textit{ or if } C_i^I(a) = \boldsymbol{t} \textit{ then } C_i^I(b) = \boldsymbol{i} \textit{ or if } C_i^I(a) = \boldsymbol{f} \textit{ then } C_i^I(b) = \boldsymbol{i}\}.$$

In $P_{\mathcal{C}}$, whose definition was borrowed from [8], it is assumed that the information may be partially described because of our incomplete or contradictory knowledge. From this point of view, an individual a can be considered similar to b if the information obtained in a is also obtained in b. Hence, for a concept C, where $C^I(a) = \mathbf{t}$ and $C^I(b) = \mathbf{i}$, the individual a is similar to b since the truth value **t** is contained in **i**. Note the converse is not true: b is not similar to a according to $P_{\mathcal{C}}$.

The contextual lower and upper approximation of a DL concept C w.r.t the context $\mathcal{C}$ is denoted by $\underline{C}_{\mathcal{R}_{\mathcal{C}}}$ and $\overline{C}^{\mathcal{R}_{\mathcal{C}}}$, respectively, where $\mathcal{R}$ is one of those similarity relations presented above. It is important to emphasize that, for approximations with indiscernibility and similarity relations, we have the following result:

Proposition 1 *[15] Given the concept C and the context $\mathcal{C}$, it holds that:*

$$\underline{C}_{P_{\mathcal{C}}} \sqsubseteq \underline{C}_{S_{\mathcal{C}}} \sqsubseteq \underline{C}_{R_{\mathcal{C}}} \sqsubseteq C \sqsubseteq \overline{C}^{R_{\mathcal{C}}} \sqsubseteq \overline{C}^{S_{\mathcal{C}}} \sqsubseteq \overline{C}^{P_{\mathcal{C}}}.$$

This holds because $P_{\mathcal{C}}$ is more general than $S_{\mathcal{C}}$, whilst, $S_{\mathcal{C}}$ is more general than $R_{\mathcal{C}}$.

4 Query Refinement

Rough set theory is an interesting candidate as a framework to be employed in query refinement. By definition, the upper approximation will add an individual to the result of the query as soon as it is related to one of the concepts already in the query, while the lower approximation will only retain an individual in the result if all related concepts are in the query. We can imagine a situation in which a query results in an empty answer; in this case, its upper approximation could be applied to possibly produce at least individuals related to the query. On the other hand, a query may result in many individuals, thus a lower approximation could be applied to possibly restrict the number of individuals in order to obtain those most related to the query.

However, the lower approximation may result in the empty query, being in this case too strict for query refinement. As for the upper approximation, it corresponds to a well known approach to query expansion. Nevertheless, it can be too flexible as a query expansion technique, resulting not only in an explosion of the query, but possibly even worse, in the addition of non-relevant individuals due to the ambiguous nature of some of the query concepts.

In this section, we combine the flexibility of the upper approximation with the strictness of the lower approximation by applying them alternately [3,4]. For instance, first we can expand the query by adding all the individuals that are known to be related to at least one of the query concepts. Next, we can reduce the expanded query by taking its lower approximation, thereby pruning away all previously added individuals suspected to be irrelevant for the query. The pruning strategy targets those individuals that are strongly related to the query concepts, but that do not belong to the expanded query. All these strategies of approximations are defined in *PR*$_{\mathcal{ALC}}$ in the sequel:

Definition 15 (Tight and Loose Upper/Lower Approximations) *Tight and loose upper/lower approximations are denoted by $\underline{\overline{C}^{S}}_{S}$, $\overline{\overline{C}^{S}}^{S}$, $\underline{\underline{C}_{S}}_{S}$, and $\overline{\underline{C}_{S}}^{S}$, and are defined as*

Syntax	**Semantics**
$\underline{\overline{C}^S}_S$	$(\underline{\overline{C}^S}_S)^I(x) = \underset{y \in \Delta^I}{\mathrm{GLB}^t}(S^I(x,y) \hookrightarrow \underset{z \in \Delta^I}{\mathrm{LUB}^t}(S^I(y,z) \wedge C^I(z)))$
$\overline{\overline{C}^S}^S$	$(\overline{\overline{C}^S}^S)^I(x) = \underset{y \in \Delta^I}{\mathrm{LUB}^t}(S^I(x,y) \wedge \underset{z \in \Delta^I}{\mathrm{LUB}^t}(S^I(y,z) \wedge C^I(z)))$
$\underline{\underline{C}_S}_S$	$(\underline{\underline{C}_S}_S)^I(x) = \underset{y \in \Delta^I}{\mathrm{GLB}^t}(S^I(x,y) \hookrightarrow \underset{z \in \Delta^I}{\mathrm{GLB}^t}(S^I(y,z) \hookrightarrow C^I(z)))$
$\overline{\underline{C}_S}^S$	$(\overline{\underline{C}_S}^S)^I(x) = \underset{y \in \Delta^I}{\mathrm{LUB}^t}(S^I(x,y) \wedge \underset{z \in \Delta^I}{\mathrm{GLB}^t}(S^I(y,z) \hookrightarrow C^I(z)))$

Proposition 2 *[4] Given the concept C, a similarity relation S, and the indiscernibility relation $R_{\mathcal{C}}$, it holds that:*

- $\underline{\underline{C}_S}_S \sqsubseteq \underline{C}_S \sqsubseteq C \sqsubseteq \overline{C}^S \sqsubseteq \overline{\overline{C}^S}^S$
- $\underline{C}_S \sqsubseteq \overline{\underline{C}_S}^S \sqsubseteq \overline{C}^S$ *and* $\underline{C}_S \sqsubseteq \underline{\overline{C}^S}_S \sqsubseteq \overline{C}^S$
- $\underline{\underline{C}_{R_{\mathcal{C}}}}_{R_{\mathcal{C}}} \equiv \underline{C}_{R_{\mathcal{C}}} \sqsubseteq C \sqsubseteq \overline{C}^{R_{\mathcal{C}}} \equiv \overline{\overline{C}^{R_{\mathcal{C}}}}^{R_{\mathcal{C}}}$
- $\underline{C}_{R_{\mathcal{C}}} \equiv \overline{\underline{C}_{R_{\mathcal{C}}}}^{R_{\mathcal{C}}} \sqsubseteq \overline{C}^{R_{\mathcal{C}}}$ *and* $\underline{C}_{R_{\mathcal{C}}} \sqsubseteq \underline{\overline{C}^{R_{\mathcal{C}}}}_{R_{\mathcal{C}}} \equiv \overline{C}^{R_{\mathcal{C}}}$

Note that the application of tight and loose approximations w.r.t. an indiscernibility relation does not result in new answers to a query. Otherwise Proposition 2 suggests that if we resort to similarity relations, successive applications of approximations may result in different answers, making a similarity relation an interesting alternative to query refinement. For an application of *$PR_{\mathcal{ALC}}$* to query refinement, let us consider an example considering similarity relations for incomplete and contradictory information.

Example 2 (Query Relaxation/Restriction) Let $x_1, x_2, x_3, x_4, x_5, x_6$ and x_7 be a set of individuals representing houses; *GoodLocation*, *Basement*, *Fireplace*, *Expensive*, *Cheap* and *Medium* be DL concepts; $\mathcal{C} = \{\mathit{GoodLocation}, \mathit{Basement}, \mathit{Fireplace}\}$ be a context and $I = (\Delta^I, \cdot^I)$ be an interpretation such that

$\Delta^I = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$, $\mathit{GoodLocation}^I = \{x_1, \neg x_2, x_3, \neg x_4, x_6, \neg x_7\}$,
$\mathit{Basement}^I = \{x_1, x_2, \neg x_2, \neg x_3, x_4, x_6, \neg x_6, x_7\}$,
$\mathit{Fireplace}^I = \{x_1, \neg x_2, \neg x_4, x_5, x_6, x_7\}$,
$\mathit{Medium}^I = \{\neg x_1, \neg x_2, x_3, x_4, x_5, \neg x_6, x_7\}$,
$\mathit{Expensive}^I = \{x_1, \neg x_2, \neg x_3, \neg x_4, \neg x_5, x_6, \neg x_7\}$,
$\mathit{Cheap}^I = \{\neg x_1, x_2, \neg x_3, \neg x_4, \neg x_5, \neg x_6, \neg x_7\}$.

Now, let us show our first example using query relaxation: suppose that we want to know what houses are expensive. Making a query with each house, we have that

$$I \models \mathit{Expensive}(x_1),\ I \not\models \mathit{Expensive}(x_2),\ I \not\models \mathit{Expensive}(x_3),\ I \not\models \mathit{Expensive}(x_4) \text{ and } I \not\models \mathit{Expensive}(x_5).$$

That is, x_1 is the only expensive house. But suppose that we want to know which houses are possibly expensive. Using query relaxation we will have that

$$I \models \overline{\mathit{Expensive}}^{S_{\mathcal{C}}}(x_1) \text{ and } I \models \overline{\mathit{Expensive}}^{S_{\mathcal{C}}}(x_5).$$

Thus, x_1 and x_5 are possibly expensive. We can conclude that x_5 is similar to x_1, because x_1 is the only object which is expensive. If we use query relaxation again (loose upper approximation) we conclude that

$$I \models \overline{\overline{\mathit{Expensive}}^{S_{\mathcal{C}}}}^{S_{\mathcal{C}}}(x_1),\ I \models \overline{\overline{\mathit{Expensive}}^{S_{\mathcal{C}}}}^{S_{\mathcal{C}}}(x_3) \text{ and } I \models \overline{\overline{\mathit{Expensive}}^{S_{\mathcal{C}}}}^{S_{\mathcal{C}}}(x_5).$$

We have now that x_3 is loosely possibly an expensive object: because x_3 is similar to x_5, an object possibly expensive. Another example related to query refinement, but using query restriction: suppose that we want to know what houses are neither expensive nor cheap, i.e., medium value. Making a query with each house, we have that

$$I \not\models \mathit{Medium}(x_1),\ I \not\models \mathit{Medium}(x_2),\ I \models \mathit{Medium}(x_3),\ I \models \mathit{Medium}(x_4) \text{ and}$$
$$I \models \mathit{Medium}(x_5).$$

Using query restriction, we can conclude that

$$I \models \underline{\mathit{Medium}}_{S_{\mathcal{C}}}(x_3), \text{ but } I \not\models \underline{\mathit{Medium}}_{S_{\mathcal{C}}}(x_4) \text{ and } I \not\models \underline{\mathit{Medium}}_{S_{\mathcal{C}}}(x_5).$$

That is, x_4 and x_5 do not have necessarily medium value. If we use query restriction again (tight lower approximation) we can conclude that

$$I \not\models \underline{\underline{\mathit{Medium}}_{S_{\mathcal{C}}}}_{S_{\mathcal{C}}}(x_3).$$

Thus x_3 surely does not necessarily have medium value (i.e., it is similar to an object that necessarily does not have medium value). Instead of using tight lower approximation, if we want to know what houses possibly have necessarily medium value, we may use loose lower approximation and conclude that

$$I \models \overline{\underline{\mathit{Medium}}_{S_{\mathcal{C}}}}^{S_{\mathcal{C}}}(x_3) \text{ and } I \models \overline{\underline{\mathit{Medium}}_{S_{\mathcal{C}}}}^{S_{\mathcal{C}}}(x_5).$$

With this example, we obtain that x_5 necessarily does not have medium value, but possibly necessarily it has medium value (because x_5 is similar to x_3, which has necessarily medium value).

Focusing on the similarity relation for inconsistent information, we have that $I \not\models \mathit{Cheap}(x_4)$ and $I \not\models \overline{\mathit{Cheap}}^{S_{\mathcal{C}}}(x_4)$, but $I \models \overline{\mathit{Cheap}}^{P_{\mathcal{C}}}(x_4)$. This means that $P_{\mathcal{C}}$ can be used to discover individuals related to contradictions within the context. Knowing $I \not\models \overline{\mathit{Cheap}}^{S_{\mathcal{C}}}(x_4)$ and $I \models \overline{\mathit{Cheap}}^{P_{\mathcal{C}}}(x_4)$, we infer that there is contradictory information in $\mathcal{C}$, and x_4 could be related to it. A similar intuition may be used for lower approximation to discover those individuals which certainly are not related to contradictions. For instance, as $I \models \underline{\mathit{Medium}}_{P_{\mathcal{C}}}(x_7)$, the individual x_7 is certainly medium and not related to contradictions. On the other hand, as $I \models \underline{\mathit{Medium}}_{S_{\mathcal{C}}}(x_3)$ and $I \not\models \underline{\mathit{Medium}}_{P_{\mathcal{C}}}(x_3)$, although x_3 is certainly medium, it is related to contradictions. As we can see, in $\mathit{PR}_{\mathcal{ALC}}$, we can represent very elaborated query refinements.

5 Conclusion

In this paper, we introduced the paraconsistent rough description logic $\mathit{PR}_{\mathcal{ALC}}$, suitable to represent and approximate incomplete and contradictory concepts. Besides including in its language indiscernibility relations, our proposal permits to reason with more relaxed notions of similarity relations in order to characterise the upper/lower approximations of a concept/query. As consequence, many sophisticated kinds of query refinements can be represented in $PR_{\mathcal{ALC}}$. Owing to the similarity relations, one of its distinctive aspects is that successive applications of approximations may result in successive refinements of a query.

As a future work, we will extend the $PR_{\mathcal{ALC}}$ to represent and approximate more sorts of incomplete knowledge, as in [8], where the incomplete information can be divided into several kinds of missing information. Consequently, new similarity relations can be introduced to model each kind of missing information. Another track to be explored is to introduce dominance relations [7], which are commonly employed to model preference between information and individuals.

References

1. N. D. Belnap. A useful four-valued logic. In J. Michael Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel, 1977.
2. F. Bobillo and U. Straccia. Supporting fuzzy rough sets in fuzzy description logics. In *ECSQARU*, pages 676–687, 2009.
3. M. De Cock and C. Cornelis. Fuzzy rough set based web query expansion. In *Proceedings of Rough Sets and Soft Computing in Intelligent Agent and Web Technology, International Workshop at WIIAT2005*, pages 9–16, 2005.
4. M. De Cock, C. Cornelis, and E. E. Kerre. Fuzzy rough sets: Beyond the obvious. In *Proceedings of FUZZ-IEEE2004*, volume 1, pages 103–108, 2004.
5. N. Fanizzi, C. d'Amato, F. Esposito, and T. Lukasiewicz. Representing uncertain concepts in rough description logics via contextual indiscernibility relations. In *URSW*, 2008.
6. M.L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
7. S. Greco, B. Matarazzo, and R. Slowinski. Rough approximation of a preference relation by dominance relations. *European Journal of Operational Research*, 117(1):63–83, 1999.
8. J.W. Grzymala-Busse. Rough set strategies to data with missing attribute values. In *Foundations and Novel Approaches in Data Mining*, pages 197–212. 2006.
9. Y. Jiang, J. Wang, S. Tang, and B. Xiao. Reasoning with rough description logics: An approximate concepts approach. *Inf. Sci.*, 179(5):600–612, 2009.
10. C. Maria Keet. On the feasibility of description logic knowledge bases with rough concepts and vague instances. In *Description Logics*, 2010.
11. Y. Ma, P. Hitzler, and Z. Lin. Algorithms for paraconsistent reasoning with owl. In *The Semantic Web: Research and Applications. Proceedings of the 4th European Semantic Web Conference, ESWC2007*, pages 399–413. Springer, 2007.
12. J. Małuszyński, A. Szalas, and A. Vitória. A four-valued logic for rough set-like approximate reasoning. *T. Rough Sets*, 6:176–190, 2007.
13. Z. Pawlak. Rough sets. *International Journal of Information and Computer Science*, 11:341–356, 1982.
14. S. Schlobach, M. Klein, and Vrije Universteit Amsterdam. L.: Description logics with approximate definitions: Precise modeling of vague concepts. In *IJCAI 07*, 2007.
15. R. Słowiński and D. Vanderpooten. Similarity relation as a basis for rough approximations. *P.P. Wang, ed., Advances in Machine Intelligence and Soft-Computing. Vol.IV, Bookwrights, Raleigh, NC*, 1997.
16. A. Vitória, A. Szałas, and J. Małuszyński. Four-valued extension of rough sets. In *Proceedings of Third International Conference on Rough Sets and Knowledge Technology, Chengdu, China (17th–19th May 2008)*, volume 5009 of *LNCS*, pages 106–114, 2008.

Author Index