Optimizing TBox and ABox Reasoning with Pseudo Models

Volker Haarslev and Ralf Möller
University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
http://kogs-www.informatik.uni-hamburg.de/~<name>/

Abstract

This paper investigates optimization techniques and data structures exploiting the use of so-called *pseudo models*. These techniques are applied to speed-up TBox and ABox reasoning for the description logic $\mathcal{ALCNH}_{R^+}$. The advances are demonstrated by an empirical analysis using the description logic system RACE that implements TBox and ABox reasoning for $\mathcal{ALCNH}_{R^+}$.

1 Introduction

We introduce and analyze optimization techniques for reasoning in expressive description logics exploiting so-called pseudo models. The new techniques being investigated are called *deep model merging* and *individual model merging*. The presented algorithms are empirically evaluated using TBoxes and ABoxes derived from actual applications.

We briefly introduce the DL $\mathcal{ALCNH}_{R^+}$ [2]. We assume a set of concept names C, a set of role names R, and a set of individual names O. The mutually disjoint subsets F, P, T of R denote features, non-transitive, and transitive roles, respectively ($R = F \cup P \cup T$). $\mathcal{ALCNH}_{R^+}$ is introduced in Figure 1 using a standard Tarski-style semantics. This a slightly extended definition of $\mathcal{ALCNH}_{R^+}$ compared to the one given in [2] since we additionally support the declaration of "native" features (elements of F). This allows additional optimizations, e.g. an efficient treatment of features by the model merging technique (see below). The concept name $\top$ ($\bot$) is used as an abbreviation for $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$).

If $\mathsf{R}, \mathsf{S} \in R$ are role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ is called a *role inclusion* axiom. A *role hierarchy* $\mathcal{R}$ is a finite set of role inclusion axioms. Then, we define $\sqsubseteq^*$ as the reflexive transitive closure of $\sqsubseteq$ over such a role hierarchy $\mathcal{R}$. Given

Syntax	**Semantics**
Concepts	
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\neg\mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
$\exists\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists\,\mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}},\, \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\forall\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall\,\mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \Rightarrow \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\exists_{\geq n}\,\mathsf{S}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \,\|\{\mathsf{b} \in \Delta^{\mathcal{I}} \mid (\mathsf{a},\mathsf{b}) \in \mathsf{S}^{\mathcal{I}}\}\| \geq \mathsf{n}\}$
$\exists_{\leq n}\,\mathsf{S}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \,\|\{\mathsf{b} \in \Delta^{\mathcal{I}} \mid (\mathsf{a},\mathsf{b}) \in \mathsf{S}^{\mathcal{I}}\}\| \leq \mathsf{n}\}$
Roles	
R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

$n \in \mathbb{N}$, $n \geq 0$, $\|\cdot\|$ denotes set cardinality, and $\mathsf{S} \in S$.

Axioms	
Syntax	Satisfied if
$\mathsf{R} \in T$	$\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$
$\mathsf{F} \in F$	$\Delta^{\mathcal{I}} \subseteq (\exists_{\leq 1}\,\mathsf{F})^{\mathcal{I}}$
$\mathsf{R} \sqsubseteq \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$
$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$

Assertions	
Syntax	Satisfied if
$\mathsf{a} : \mathsf{C}$	$\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{b}) : \mathsf{R}$	$(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$

Figure 1: Syntax and Semantics of $\mathcal{ALCNH}_{R^+}$.

$\sqsubseteq^*$, the set of roles $\mathsf{R}^{\downarrow} = \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^* \mathsf{R}\}$ defines the *descendants* of a role R. $\mathsf{R}^{\uparrow} = \{\mathsf{S} \in R \mid \mathsf{R} \sqsubseteq^* \mathsf{S}\}$ is the set of *ancestors* of a role R. We also define the set $S := \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$ of *simple* roles that are neither transitive nor have a transitive role as descendant. Every descendant G of a feature F must be a feature as well ($\mathsf{G} \in F$).

A syntactic restriction holds for the combinability of number restrictions and transitive roles in $\mathcal{ALCNH}_{R^+}$. Number restrictions are only allowed for *simple* roles. This restriction is motivated by a known undecidability result in case of an unrestricted combinability [7].

If C and D are concept terms, then $\mathsf{C} \sqsubseteq \mathsf{D}$ *(generalized concept inclusion* or *GCI*) is a terminological axiom as well. A finite set of terminological axioms $\mathcal{T}_{\mathcal{R}}$ is called a *terminology* or *TBox* w.r.t. to a given role hierarchy $\mathcal{R}$.[1]

An *ABox* $\mathcal{A}$ is a finite set of assertional axioms as defined in Figure 1. For an initial ABox $\mathcal{A}$ the set O_O is defined as $O_O = \{\mathsf{a} \mid \mathsf{a}$ mentioned in $\mathcal{A}\}$. Every individual name from O is mapped to a single element of $\Delta^{\mathcal{I}}$ in a way such that for $\mathsf{a}, \mathsf{b} \in O_O$, $\mathsf{a}^{\mathcal{I}} \neq \mathsf{b}^{\mathcal{I}}$ if $\mathsf{a} \neq \mathsf{b}$ *(unique name assumption)*. The ABox consistency problem is to decide whether a given ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$. Satisfiability of concept terms can be reduced to ABox consistency as follows: A concept term C is satisfiable iff the ABox $\{\mathsf{a} : \mathsf{C}\}$ is consistent. *Instance checking* tests whether an individual a is an instance of a concept term C w.r.t. an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, i.e. whether $\mathcal{A}$ entails $\mathsf{a} : \mathsf{C}$ w.r.t. $\mathcal{T}$. This problem is reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{a} : \neg\mathsf{C}\}$ is inconsistent. Computing the *direct types* of individuals (i.e. the set of the most specific concepts from C of which an individual is an instance) is called *realization*.

[1]The reference to $\mathcal{R}$ is omitted in the following if we use $\mathcal{T}$.

An ABox $\mathcal{A}$ is said to contain a clash if one of the following *clash triggers* is applicable. Otherwise $\mathcal{A}$ is called *clash-free*.

- *Primitive clash*: $\mathsf{a}:\bot \in \mathcal{A}$ or $\{\mathsf{a}:\mathsf{A}, \mathsf{a}:\neg\mathsf{A}\} \subseteq \mathcal{A}$, where A is a concept name.
- *Number restriction clash*: $\exists\,\mathsf{S}_1, \ldots, \mathsf{S}_\mathsf{m} \in \mathsf{R}^{\downarrow} : \{\mathsf{a}:\exists_{\leq n}\,\mathsf{R}\} \cup \{(\mathsf{a}, \mathsf{b}_\mathsf{i}):\mathsf{S}_\mathsf{i} \,|\, \mathsf{i} \in 1..\mathsf{m}\} \cup \{\mathsf{b}_\mathsf{i} \neq \mathsf{b}_\mathsf{j} \,|\, \mathsf{i}, \mathsf{j} \in 1..\mathsf{m}, \mathsf{i} \neq \mathsf{j}\} \subseteq \mathcal{A}$ with $\mathsf{m} > \mathsf{n}$.

A clash-free ABox $\mathcal{A}$ is called *complete* if no completion rule (omitted due to lack of space, see [2]) is applicable to $\mathcal{A}$. A complete ABox $\mathcal{A}'$ derived from an ABox $\mathcal{A}$ is also called a *completion* of $\mathcal{A}$.

Based on these notions we introduce and evaluate in the next sections the new optimization techniques. The evaluation clearly demonstrates a speed gain using deep model merging for classification of concepts and a dramatic gain for the realization of ABoxes if individual model merging is applied.

2 Exploiting Deep Models for TBox Reasoning

Given a set of concepts representing a conjunction whose consistency is to be checked, the *model merging strategy* tries to avoid a consistency test which relies on the "expensive" tableaux technique. This idea was first introduced in [4] for the logic $\mathcal{ALCH}f_{R^+}$. A model merging test is designed to be a "cheap" test operating on cached "concept models." It is a *sound* but incomplete consistency tester for a set of concepts. The achievement of minimal computational overhead and the avoidance of any nondeterminism are important characteristics of such a test. If the test returns false, a sound and complete tableaux calculus is applied. In order to be more precise, we use the term *pseudo model* instead of "concept model." A model is understood in the sense of an interpretation and a pseudo model as a data structure containing recorded information.

In the following we present and analyze a technique called *deep model merging* that generalizes the original model merging approach [4] in two ways. (1) We extend the model merging technique to the logic $\mathcal{ALCNH}_{R^+}$. (2) We introduce *deep* pseudo models which are *recursively* traversed and checked for possible clashes. To the best of our knowledge this is the first formal treatment showing the soundness of model merging.

A pseudo model for a concept term C is defined as follows. Let $\mathsf{A} \in C$ be a concept name, $\mathsf{R} \in R$ a role name, $\mathsf{F} \in F$ a feature name. The consistency of $\mathcal{A} = \{\mathsf{a}:\mathsf{C}\}$ is tested. If $\mathcal{A}$ is inconsistent, the *pseudo model*[2] of C is defined as $\bot$. If $\mathcal{A}$ is consistent, then there exists a set of completions $\mathcal{C}$. A completion $\mathcal{A}' \in \mathcal{C}$ is selected and a pmodel M for a concept C is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall}\rangle$ of concept sets using the following definitions.

[2]For brevity a pseudo model is called a *pmodel*.

$$M^{\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a} : \mathsf{A} \in \mathcal{A}'\}, \quad M^{\neg\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a} : \neg\mathsf{A} \in \mathcal{A}'\}$$

$$M^{\exists} = \{\exists\, \mathsf{R} \,.\, \mathsf{C} \,|\, \mathsf{a} : \exists\, \mathsf{R} \,.\, \mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\, \mathsf{R} \,|\, \mathsf{a} : \exists_{\geq n}\, \mathsf{R} \in \mathcal{A}'\}$$

$$M^{\forall} = \{\forall\, \mathsf{R} \,.\, \mathsf{C} \,|\, \mathsf{a} : \forall\, \mathsf{R} \,.\, \mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\leq n}\, \mathsf{R} \,|\, \mathsf{a} : \exists_{\leq n}\, \mathsf{R} \in \mathcal{A}'\} \cup \{\exists\, \mathsf{F} \,.\, \mathsf{C} \,|\, \mathsf{a} : \exists\, \mathsf{F} \,.\, \mathsf{C} \in \mathcal{A}'\}$$

Note that the set $M^{\exists}$ contains all exists- and at-least concepts while $M^{\forall}$ contains all at-most- and all-concepts as well as all exists-concepts for features. This guarantees the correct treatment of features.

The procedure **mergable** shown in Procedure 1 implements the flat and deep model merging test. In case of deep merging it has to test for a blocking situation, i.e. whether the actual pmodel set *MS* is a member of the set *VM* of visited pmodel sets. The initial call of mergable has the empty set as value for *VM*. The third parameter *D*? controls whether the deep or flat mode (see below) of mergable will be used.

We assume a procedure **get_pmodel** that retrieves for a concept C its cached pmodel. In case the pmodel does not yet exist, it is computed.

Procedure 1 mergable(*MS*, *VM*, *D*?)

```
if MS = ∅ ∨ MS ∈ VM then
  return true
else if ⊥ ∈ MS ∨ ¬atoms_mergable(MS) then
  return false
else
  for all M ∈ MS do
    for all C ∈ M^∃ do
      if critical_at_most(C, M, MS) then
        return false
      else
        MS' ← collect_pmodels(C, MS)
        if (¬D? ∧ MS' ≠ ∅) ∨ ¬mergable(MS', VM ∪ {MS}, D?) then
          return false
return true
```

The procedure **atoms_mergable** tests for a possible primitive clash between pairs of pmodels. It is applied to a set of pmodels *MS* and returns *false* if there exist $\{M_1, M_2\} \subseteq MS$ with $(M_1^{\mathsf{A}} \cap M_2^{\neg\mathsf{A}}) \neq \emptyset$ or $(M_1^{\neg\mathsf{A}} \cap M_2^{\mathsf{A}}) \neq \emptyset$. Otherwise it returns *true*.

The procedure **critical_at_most** tests for a potential number restriction clash in a set of pmodels and tries to avoid *true* answers which are too conservative. It is applied to a concept C of the form $\exists\, \mathsf{S} \,.\, \mathsf{D}$ or $\exists_{\geq n}\, \mathsf{S}$, a pmodel M (the current model) and a set of pmodels $MS = \{M_1, \ldots, M_k\}$ and returns *true* if there exists a pmodel $M' \in (MS \setminus M)$ and a role $\mathsf{R} \in \mathsf{S}^{\uparrow}$ with $\exists_{\leq m} \mathsf{R} \in M'^{\forall}$ such that $\sum_{\mathsf{E} \in N} \mathrm{num}_{RS}(\mathsf{E}) > m$, $N = \cup_{i \in 1..k} M_i^{\exists}$, $RS = \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow}$. In all other cases critical_at_most returns *false*. The procedure $\mathrm{num}_{RS}(\mathsf{E})$ returns 1 for concepts of the form $\mathsf{E} = \exists \mathsf{R}' . \mathsf{D}$ and n for $\mathsf{E} = \exists_{\geq n} \mathsf{R}'$, provided $\mathsf{R}' \in RS$.

The procedure **collect_pmodels** is applied to a concept C of the form $\exists \mathsf{S} . \mathsf{D}$ or $\exists_{\geq n} \mathsf{S}$ and a set of pseudo models MS. It computes the pmodels of the set Q of "qualifications." We define $Q' = \{\mathsf{D}\}$ if $\mathsf{C} = \exists \mathsf{S} . \mathsf{D}$ and $Q' = \emptyset$ otherwise.

$$Q = Q' \cup \{\mathsf{E} \mid \exists M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow} : (\forall \mathsf{R} . \mathsf{E} \in M^{\forall} \vee \exists \mathsf{R} . \mathsf{E} \in M^{\forall})\} \cup$$
$$\{\forall \mathsf{T} . \mathsf{E} \mid \exists M \in MS, \mathsf{R} \in \mathsf{S}^{\uparrow}, \mathsf{T} \in T \cap \mathsf{S}^{\uparrow} \cap \mathsf{R}^{\downarrow} : \forall \mathsf{R} . \mathsf{E} \in M^{\forall}\}$$

The procedure collect_pmodels returns the set $\{\text{get\_pmodel}(\mathsf{C}) \mid \mathsf{C} \in Q\}$. Observe that $\exists \mathsf{R} . \mathsf{E} \in M^{\forall}$ implies that R is a feature.

In the following we prove the soundness of the procedure mergable. Note that mergable depends on the clash triggers of the particular tableaux calculus chosen since it has to detect potential clashes in a set of pmodels. The structure and composition of the completion rules might vary as long as the clash triggers do not change and the calculus remains sound and complete.

Proposition 1 (Soundness of mergable) *Let $D?$ have either the value* true *or* false*, $CS = \{\mathsf{C_1}, \ldots, \mathsf{C_n}\}$, $M_{C_i} = \text{get\_pmodel}(\mathsf{C_i})$, and $PM = \{M_{C_i} \mid \mathsf{i} \in 1..\mathsf{n}\}$. If the procedure call* mergable$(PM, \emptyset, D?)$ *returns* true*, the concept $\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n}$ is consistent.*

Proof. This is proven by contradiction and induction. Let us assume that the call mergable$(PM, \emptyset, D?)$ returns *true* but the ABox $\mathcal{A} = \{\mathsf{a} : (\mathsf{C_1} \sqcap \ldots \sqcap \mathsf{C_n})\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}$. Every concept $\mathsf{C_i}$ must be satisfiable, otherwise we would have $\perp \in PM$ and mergable would return *false* due to line 3 in Procedure 1. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}$. Without loss of generality we can select an arbitrary $\mathcal{A}' \in \mathcal{C}$ and make a case analysis of its possible clash culprits.

1. We have a primitive clash for the "root" individual a, i.e. $\{\mathsf{a} : \mathsf{D}, \mathsf{a} : \neg \mathsf{D}\} \subseteq \mathcal{A}'$. Thus, $\mathsf{a} : \mathsf{D}$ and $\mathsf{a} : \neg \mathsf{D}$ have not been propagated to a via role assertions and there have to exist $\mathsf{C_i}, \mathsf{C_j} \in CS$, $\mathsf{i} \neq \mathsf{j}$ such that $\mathsf{a} : \mathsf{D}$ ($\mathsf{a} : \neg \mathsf{D}$) is derived from $\mathsf{a} : \mathsf{C_i}$ ($\mathsf{a} : \mathsf{C_j}$) due to the satisfiability of the concepts $\mathsf{C_i}$, $\mathsf{i} \in 1..\mathsf{n}$. It holds for the associated pmodels $M_{C_i}, M_{C_j} \in PM$ that $\mathsf{D} \in M_{C_i}^{\mathsf{A}} \cap M_{C_j}^{\neg \mathsf{A}}$. However, due to our assumption the call of mergable$(PM, \emptyset, D?)$ returned *true*. This is a contradiction since mergable called atoms_mergable with PM (line 3 in Procedure 1) which returned *false* since $\mathsf{D} \in M_{C_i}^{\mathsf{A}} \cap M_{C_j}^{\neg \mathsf{A}}$.
2. A number restriction clash in $\mathcal{A}'$ is detected for a, i.e. $\mathsf{a} : \exists_{\leq m} \mathsf{R} \in \mathcal{A}'$ and there exist $l > m$ distinct R-successors of a.[3] These successors can only

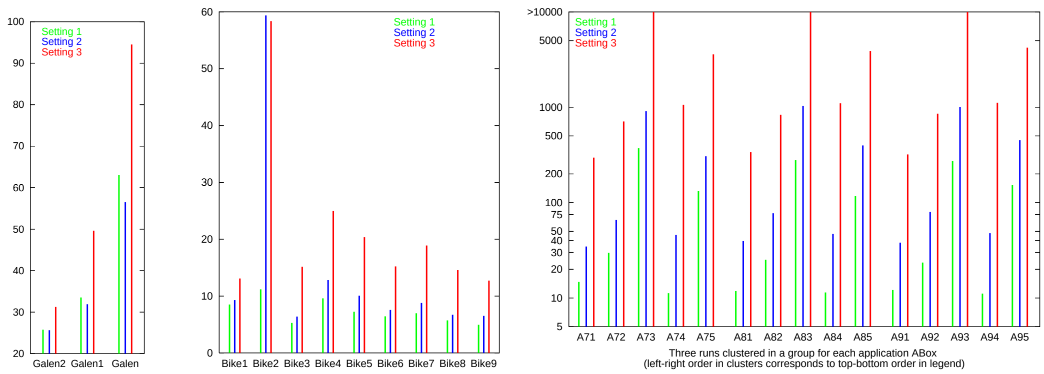
[3]Due to our syntax restriction all elements of $\mathsf{R}^{\downarrow}$ are not transitive.

be derived from assertions of the form $\mathsf{a} : \exists\, \mathsf{S_j} \,.\, \mathsf{E_j}$ or $\mathsf{a} : \exists_{\geq n_j} \mathsf{S_j}$ with $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in 1..\mathsf{k_1}$. The concepts $\mathsf{C_i} \in CS$, $\mathsf{i} \in 1..\mathsf{n}$ are satisfiable and there has to exist a subset $CS' = \{\mathsf{C_{i_1}}, \ldots, \mathsf{C_{i_{k_2}}}\} \subseteq CS$ such that $\exists_{\leq m} \mathsf{R} \in \cup_{C_i \in CS'} M^{\forall}_{C_i}$ and $\sum_{\mathsf{E'} \in N} \mathrm{num}_{RS}(\mathsf{E'}) \geq l$, $N = \cup_{C_i \in CS'} M^{\exists}_{C_i}$, $RS = (\cup_{\mathsf{j} \in 1..\mathsf{k_1}} \mathsf{S_j}^{\uparrow}) \cap \mathsf{R}^{\downarrow}$. However, due to our assumption the call of mergable($PM, \emptyset, D?$) returned *true*. This is a contradiction since there exists an $\mathsf{i'} \in 1..\mathsf{k_2}$ and a concept $\mathsf{E'} \in M^{\exists}_{C_{i'}}$ such that mergable called critical_at_most($\mathsf{E'}, M_{C_{i'}}, PM$) (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E'} \in N} \mathrm{num}_{RS}(\mathsf{E'}) \geq l > m$.

3. Let the individual $\mathsf{a_n}$ be a successor of $\mathsf{a_0}$ via a chain of role assertions $(\mathsf{a_0}, \mathsf{a_1}) : \mathsf{R_1}, \ldots, (\mathsf{a_{n-1}}, \mathsf{a_n}) : \mathsf{R_n}$, $\mathsf{n} > 0$ and we now assume that a clash for $\mathsf{a_n}$ is discovered.
   (a) In case of a primitive clash we have $\{\mathsf{a_n} : \mathsf{D}, \mathsf{a_n} : \neg\mathsf{D}\} \subseteq \mathcal{A}'$. These clash culprits are derived from assertions for $\mathsf{a_{n-1}}$ of the form $\mathsf{a_{n-1}} : \exists_{\geq m} \mathsf{R_n}$ or $\mathsf{a_{n-1}} : \exists\, \mathsf{R_n} \,.\, \mathsf{E_1}$, and $\mathsf{a_{n-1}} : \forall\, \mathsf{S} \,.\, \mathsf{E_2}$ and/or $\mathsf{a_{n-1}} : \forall\, \mathsf{S'} \,.\, \mathsf{E_3}$ with $\mathsf{S}, \mathsf{S'} \in \mathsf{R_n}^{\uparrow}$. Due to the clash there exists a pair $\mathsf{E_i}, \mathsf{E_j}$ with $\mathsf{D} \in M^{\mathsf{A}}_{E_i} \cap M^{\neg\mathsf{A}}_{E_j}$ for some $\mathsf{i}, \mathsf{j} \in 1..3$, $\mathsf{i} \neq \mathsf{j}$. Each role assertion in the chain between $\mathsf{a_0}$ and $\mathsf{a_{n-1}}$ can only be derived from an assertion of the form $\mathsf{a_{k-1}} : \exists\, \mathsf{R_k} \,.\, \mathsf{E_k}$ or $\mathsf{a_{k-1}} : \exists_{\geq m_k} \mathsf{R_k}$ with $\mathsf{k} \in 1..\mathsf{n} - 1$. The call graph of mergable($PM, \emptyset, D?$) contains a chain of calls resembling the chain of role assertions. By induction on the call graph we know that the node resembling $\mathsf{a_{n-1}}$ of this call graph chain contains the call mergable(PM', VM', true) such that $\{M_{E_i}, M_{E_j}\} \subseteq PM'$ and atoms_mergable has been called with a set MS' and $\{M_{E_i}, M_{E_j}\} \subseteq MS'$. The call of atoms_mergable has returned *false* since $\mathsf{D} \in M^{\mathsf{A}}_{E_i} \cap M^{\neg\mathsf{A}}_{E_j}$. This contradicts our assumption that mergable($PM, \emptyset, D?$) returned *true*.
   (b) In case of a number restriction clash we can argue in an analogous way. Again, we have a chain of role assertions where a number restriction clash is detected for the last individual of the chain. It exists a corresponding call graph chain where by induction the last call of mergable called critical_at_most with a set of pmodels for which critical_at_most returned *true*. This contradicts the assumption that mergable($PM, \emptyset, D?$) returned *true*.

It is easy to see that this proof also holds in the case the value of $D?$ is *false* since the "flat mode" is more conservative than the "deep" one, i.e. it will always return *false* instead of possibly *true* if the set of collected pmodels M' is not empty (line 12 in Procedure 1) □

The advantage of the deep vs. the flat mode of the model merging technique is demonstrated by empirical tests using a set of "quasi-standard" application TBoxes/Aboxes [6, 5, 1]. Figure 2a-b shows the runtimes for computing the subsumption lattice of these KBs. Each KB is iteratively classified using 3 different



(a) Galen TBoxes (b) Bike TBoxes (c) Bike ABoxes

Figure 2: Evaluation of model merging techniques (3 runs for each TBox/ABox, left-right order corresponds to top-bottom order in the legend).

parameter settings. The first setting has all optimization techniques enabled, in the second one the subtableaux caching technique [5, 1] is disabled. The third setting has both subtableaux caching and the deep mode of model merging disabled but the flat mode of model merging is still enabled. The 3 different settings are justified by the order in which these optimization techniques are applied if a "subtableaux" is tested for consistency in RACE. First, subtableaux caching is applied. If no cache entry exists, (deep) model merging is tried. If it returns *false* the standard tableaux test is invoked. Thus, tableaux caching might reduce the number of encountered model merging tests and the advantage of the deep against the flat mode of model merging can only be accurately evaluated if one compares the runtimes between the second and third setting. The comparison between these settings indicates a speed-up in runtimes of a factor $1.5 - 2$ for almost all TBoxes/ABoxes if the deep mode is enabled. The comparison between the first and second setting clearly demonstrates that the deep mode can sometimes compensate the disabled subtableaux caching technique. However, the BCS TBox introduced in [1] can be classified within less than 10 seconds of runtime if subtableaux caching is enabled but cannot be classified within 10,000 seconds of runtime if subtableaux caching is disabled and the flat or deep mode of model merging is enabled.

3 Exploiting Flat Models for ABox Reasoning

An ABox is realized through a sequence of instance checking tests. The *realization* of an individual a occurring in an ABox $\mathcal{A}$ w.r.t to a TBox $\mathcal{T}$ computes the direct types of a (w.r.t. $\mathcal{A}$ and $\mathcal{T}$). For instance, in order to compute the direct types of a for a given subsumption lattice of the concepts $\mathsf{D}_1, \ldots, \mathsf{D}_n$, a sequence of ABox consistency tests for $\mathcal{A}_{\mathsf{D}_i} = \mathcal{A} \cup \{\mathsf{a} : \neg \mathsf{D}_i\}$ might be required. However, individuals are usually members of only a small number of concepts and the ABoxes $\mathcal{A}_{\mathsf{D_i}}$ are proven as consistent in most cases. The basic idea is to design a cheap but *sound* model merging test for the focused individual a and the concept terms $\neg\mathsf{D_i}$ without explicitly considering role assertions and concept assertions for the other individuals mentioned in $\mathcal{A}$ since these interactions are reflected in the "individual pseudo model" of a. This is the motivation for devising the novel *individual model merging* technique.

A pseudo model for an individual a mentioned in a consistent ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is defined as follows. Since $\mathcal{A}$ is consistent, there exists a set of completions $\mathcal{C}$ of $\mathcal{A}$. Let $\mathcal{A}' \in \mathcal{C}$. An *individual pseudo model* M for an individual a in $\mathcal{A}$ is defined as the tuple $\langle M^{\mathsf{A}}, M^{\neg\mathsf{A}}, M^{\exists}, M^{\forall}\rangle$ w.r.t. $\mathcal{A}'$ and $\mathcal{A}$ using the following definitions.

$$M^{\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a} : \mathsf{A} \in \mathcal{A}'\}, \quad M^{\neg\mathsf{A}} = \{\mathsf{A} \,|\, \mathsf{a} : \neg\mathsf{A} \in \mathcal{A}'\}$$
$$M^{\exists} = \{\exists\, \mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a} : \exists\, \mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\geq n}\, \mathsf{R} \,|\, \mathsf{a} : \exists_{\geq n}\, \mathsf{R} \in \mathcal{A}'\} \cup \{\exists_{\geq 1}\, \mathsf{R} \,|\, (\mathsf{a},\mathsf{b}) : \mathsf{R} \in \mathcal{A}\}$$
$$M^{\forall} = \{\forall\, \mathsf{R}\,.\,\mathsf{C} \,|\, \mathsf{a} : \forall\, \mathsf{R}\,.\,\mathsf{C} \in \mathcal{A}'\} \cup \{\exists_{\leq n}\, \mathsf{R} \,|\, \mathsf{a} : \exists_{\leq n}\, \mathsf{R} \in \mathcal{A}'\} \cup \{\exists\, \mathsf{F}\,.\,\mathsf{C} \,|\, \mathsf{a} : \exists\, \mathsf{F}\,.\,\mathsf{C} \in \mathcal{A}'\}$$

The procedure **get_ind_pmodel** called with an individual a mentioned in a consistent ABox $\mathcal{A}$ (w.r.t. a TBox $\mathcal{T}$) either appropriately creates a pmodel for a or retrieves the cached pmodel of a.

Proposition 2 (Soundness of individual_model_merging) *Let* a *be an individual mentioned in a consistent ABox* $\mathcal{A}$ *w.r.t. a TBox* $\mathcal{T}$*,* $\neg\mathsf{C}$ *be a satisfiable concept,* M_a *(*$M_{\neg C}$*) denote the pmodel returned by* get_ind_pmodel(a) *(*get_pmodel($\neg\mathsf{C}$)*), and the set* PM *be defined as* $\{M_a, M_{\neg C}\}$*. If the procedure call* mergable($PM, \emptyset, \mathit{false}$) *returns* true*, the ABox* $\mathcal{A} \cup \{\mathsf{a} : \neg\mathsf{C}\}$ *is consistent, i.e.* a *is not an instance of* C*.*

Proof. This is proven by contradiction. Let us assume that the procedure call mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returns *true* but the ABox $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{a} : \neg\mathsf{C}\}$ is inconsistent, i.e. there exists no completion of $\mathcal{A}'$. Let us assume a finite set $\mathcal{C}$ containing all contradictory ABoxes encountered during the consistency test of $\mathcal{A}'$. Without loss of generality we can select an arbitrary $\mathcal{A}'' \in \mathcal{C}$ and make a case analysis of its possible clash culprits.

1. A clash is detected for an individual b in $\mathcal{A}''$ that is distinct to a. Since $\mathcal{A}$ is consistent the individual b must be a successor of a via a chain of role assertions $(\mathsf{a}, \mathsf{b_1}) : \mathsf{R_1}, \ldots, (\mathsf{b_n}, \mathsf{b}) : \mathsf{R_{n+1}}, \mathsf{n} \geq 0$ and one of the clash culprits must be derived from the newly added assertion $\mathsf{a} : \neg\mathsf{C}$ and propagated to b via the role assertion chain originating from a with $(\mathsf{a}, \mathsf{b_1}) : \mathsf{R_1}$. Since $\neg\mathsf{C}$ is satisfiable and $\mathcal{A}$ is consistent we have an "interaction" via the role or feature $\mathsf{R_1}$. This implies for the associated pmodels $M_a, M_{\neg C}$ that $(M_a^{\exists} \cap M_{\neg C}^{\forall}) \cup (M_a^{\forall} \cap M_{\neg C}^{\exists}) \neq \emptyset$. This contradicts the assumption

that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable eventually called collect_pmodels for $M_a, M_{\neg C}$ which returned a non-empty set (line 11 in Procedure 1).

2. In case of a primitive clash for a we have $\{\mathsf{a}:\mathsf{D}, \mathsf{a}:\neg\mathsf{D}\} \subseteq \mathcal{A}''$. Since $\mathsf{a}:\neg\mathsf{C}$ is a concept assertion we know that $\mathsf{a}:\mathsf{D}$ and $\mathsf{a}:\neg\mathsf{D}$ cannot be propagated to a via role assertions. Thus, either $\mathsf{a}:\mathsf{D}$ or $\mathsf{a}:\neg\mathsf{D}$ must be derived from $\mathsf{a}:\neg\mathsf{C}$ and we have $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$. This contradicts the assumption that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable called atoms_mergable($\{M_a, M_{\neg C}\}$) which returned *false* (line 3 in Procedure 1) since $\mathsf{D} \in (M_a^{\mathsf{A}} \cap M_{\neg C}^{\neg\mathsf{A}}) \cup (M_a^{\neg\mathsf{A}} \cap M_{\neg C}^{\mathsf{A}})$.
3. A number restriction clash in $\mathcal{A}''$ is detected for a, i.e. $\mathsf{a}:\exists_{\leq m}\mathsf{R} \in \mathcal{A}''$ and there exist $l > m$ distinct R-successors of a in $\mathcal{A}''$. This implies that the set $N = M_a^{\exists} \cup M_{\neg C}^{\exists}$ contains concepts of the form $\exists\,\mathsf{S_j}\,.\,\mathsf{E_j}$ or $\exists_{\geq n_j}\mathsf{S_j}$,[4] $\mathsf{S_j} \in \mathsf{R}^{\downarrow}$, $\mathsf{j} \in 1..\mathsf{k}$, such that $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l$, $RS = (\cup_{\mathsf{j} \in 1..\mathsf{k}} \mathsf{S_j}) \cap \mathsf{R}^{\downarrow}$. This contradicts the assumption that mergable($\{M_a, M_{\neg C}\}, \emptyset, \mathit{false}$) returned *true* since mergable called critical_at_most (lines 6-8 in Procedure 1) which returned *true* since $\sum_{\mathsf{E}' \in N} \mathrm{num}_{RS}(\mathsf{E}') \geq l > m$. □

The performance gain by the individual model merging technique is empirically evaluated using a set of five ABoxes containing between 15 and 25 individuals. Each of these ABoxes is realized w.r.t. to the application KBs Bike7-9 derived from a bike configuration task. The KBs especially vary on the degree of explicit disjointness declarations between atomic concepts. Figure 2c shows the runtimes for the realization of the ABoxes 1-5. Each ABox is realized with three different parameter settings. The first setting has all optimization techniques enabled, in the second setting an optimization technique is disabled that considers disjointness between concepts[5], and additionally the third setting has the individual model merging technique disabled. The comparison between setting two and three reveals a speed gain of at least one order of magnitude if the individual model merging technique is used. Note the use of a logarithmic scale.

4 Conclusion and Future Work

In this paper we have analyzed optimization techniques for TBox and ABox reasoning in the expressive description logic $\mathcal{ALCNH}_{R^+}$. These techniques exploit the traversal of flat and/or deep pseudo models extracted from ABox consistency tests. A moderate speed gain using deep models for classification of concepts and a dramatic gain for realization of ABoxes is empirically demonstrated.

[4] Any role assertion of the form $(\mathsf{a}, \mathsf{b}):\mathsf{R} \in \mathcal{A}$ implies that $\exists_{\geq 1}\mathsf{R} \in M_a^{\exists}$.

[5] This technique interacts with individual model merging since it prunes the search space for realization and thus decreases the number of instance checking tests that can be solved by individual model merging. The technique dealing with disjoint concepts is discussed in [3].

The advantage of the deep vs. the flat model merging mode is apparent. If the flat model merging test is (recursively) applied during tableaux expansion and repeatedly returns *false* because of interacting all- and some-concepts, this test might be too conservative. This is illustrated with a small example ($\mathsf{C}, \mathsf{D} \in C$, $\mathsf{R}, \mathsf{S} \in R$). For instance, the deep model merging test starts with the pmodels $\langle \emptyset, \emptyset, \{\exists\, \mathsf{R}\,.\, \exists\, \mathsf{S}\,.\, \mathsf{C}\}, \emptyset \rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\, \mathsf{R}\,.\, \forall\, \mathsf{S}\,.\, \mathsf{D}\} \rangle$. Due to the interaction on the role R, the test is recursively applied to the pmodels $\langle \emptyset, \emptyset, \{\exists\, \mathsf{S}\,.\, \mathsf{C}\}, \emptyset \rangle$ and $\langle \emptyset, \emptyset, \emptyset, \{\forall\, \mathsf{S}\,.\, \mathsf{D}\} \rangle$. Eventually, the deep model merging test succeeds with the pmodels $\langle \{\mathsf{C}\}, \emptyset, \emptyset, \emptyset \rangle$ and $\langle \{\mathsf{D}\}, \emptyset, \emptyset, \emptyset \rangle$ and returns true. This is in contrast to the flat mode where in this example no tableaux tests are avoided and the runtime for the model merging tests is wasted.

It is easy to see that an enhanced version of the individual model merging technique can be developed, which additionally exploits the use of deep models. This is immediately possible if only ABoxes containing no joins for role assertions are encountered. In case an ABox $\mathcal{A}$ contains a join (e.g. $\{(\mathsf{a}, \mathsf{c}) : \mathsf{R}, (\mathsf{b}, \mathsf{c}) : \mathsf{R}\} \subseteq \mathcal{A}$), one has to consider a graph-like instead of a tree-like traversal of pseudo models reflecting the dependencies caused by joins.

References

[1] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In P. Lambrix et al., editor, *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, pages 115–119, June 1999.

[2] V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, April 11-15, 2000*, pages 273–284, April 2000.

[3] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases, August 2000. In this volume.

[4] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[5] I. Horrocks and P. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.

[6] I. Horrocks and P.F. Patel-Schneider. DL systems comparison. In *Proceedings of DL'98, International Workshop on Description Logics*, pages 55–57, Trento(Italy), 1998.

[7] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999.