Keys for Free in Description Logics

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{calvanese,degiacomo,lenzerini}@dis.uniroma1.it

Abstract

$\mathcal{DLR}$ is an expressive Description Logic (DL) with n-ary relations, particularly suited for modeling database schemas and queries. Although $\mathcal{DLR}$ has constituted one of the crucial steps for applying DL technology to data management, there is one important aspect of database schemas that DLs, including $\mathcal{DLR}$, do not capture yet, namely the notion of key. In this paper we show that keys are for free in $\mathcal{DLR}$. In particular, we address the following question: can we add keys to $\mathcal{DLR}$ and still have EXPTIME associated reasoning techniques? Somewhat surprisingly we answer positively, by showing how to adapt the $\mathcal{DLR}$ reasoning algorithm in such a way that reasoning on a $\mathcal{DLR}$ schema with keys can be done with the same worst-case computational complexity as for the case without keys.

1 Introduction

In the last years, Description Logics (DLs) have been successfully applied to data management [10, 2, 13, 8]. One of the basic ideas behind applying DLs to data management is that database schemas can be expressed as DL knowledge bases, so that DL reasoning techniques can be used in several ways to reason about the schema. In [5, 6], we introduced a very expressive DL with n-ary relations, called $\mathcal{DLR}$, and showed how database schemas can be captured by this logic (see also [7]). Also, we defined suitable mechanisms for expressing queries over $\mathcal{DLR}$ schemas, and designed techniques for reasoning over queries [5]. Notably, the investigation on $\mathcal{DLR}$ has led to the design of new DL systems effectively implementing powerful reasoning techniques [11].

Although the above mentioned work has been the crucial step for applying DL technology to data management, there is one important aspect of database schemas that DLs, including $\mathcal{DLR}$, do not capture yet, namely the notion of key. Keys are used to state that a certain set of properties uniquely identifies the instances of either a concept or a relation, and are commonly used in both database design, and data management.

The question addressed in this paper is as follows: can we add keys to $\mathcal{DLR}$ and still have EXPTIME associated reasoning techniques? Somewhat surprisingly, we answer positively to the question, by illustrating an approach that allows us to incorporate keys in $\mathcal{DLR}$ (almost) for free. In particular, we adapt the $\mathcal{DLR}$ reasoning algorithm in such a way that reasoning on a $\mathcal{DLR}$ schema with keys can be done with the same worst-case computational complexity as for the case without keys. Also, the proposed technique can be directly incorporated into present DL systems, such as the one described in [12].

In the last years, there have been some attempts to add keys to DLs. In [4], keys are modeled by means of special primitive concepts in an expressive DL, and it is shown that this mechanism allows some inference on keys to be carried on. However, the drawback of this approach is that several interesting semantic properties of keys are not represented in the knowledge base. In [3], a more expressive mechanism is proposed for modeling keys and functional dependencies, and a sound and complete inference system for reasoning on such constraints is presented. However, the DL considered in [3] is limited in expressiveness. In particular, neither number restrictions, nor general inclusion axioms, nor inverse roles are taken into account, and therefore useful properties of database schemas cannot be represented. The proposal presented in this paper fully captures the semantics of keys in an expressive DL, and, therefore, it overcomes all the limitations of the previous approaches.

The paper is organized as follows. In Section 2, we recall the DL $\mathcal{DLR}$. In Section 3, we illustrate the mechanism for specifying key constraints in $\mathcal{DLR}$ knowledge bases. In Section 4, we describe how we can extend the $\mathcal{DLR}$ reasoning technique in order to take key constraints into account. Finally, Section 5 concludes the paper by pointing out future work aiming at extending the results presented here.

2 Description Logic $\mathcal{DLR}$

We focus on the Description Logic $\mathcal{DLR}$ introduced in [6]. Such a DL is able to capture a great variety of data models with many forms of constraints [9, 6]. The basic elements of $\mathcal{DLR}$ are *concepts* (unary relations), and *n-ary relations*. We assume to deal with a finite set of atomic relations and atomic concepts, denoted by P and A, respectively. We use R to denote arbitrary relations (of given arity between 2 and n_{max}), and C to denote arbitrary concepts, respectively built

$$
\begin{array}{rcl}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
P^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} \\
(\$i/n\!:\!C)^{\mathcal{I}} & = & \{t \in \top_n^{\mathcal{I}} \mid t[i] \in C^{\mathcal{I}}\} \\
(\neg R)^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus R^{\mathcal{I}} \\
(R_1 \sqcap R_2)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} \\
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists t \in R^{\mathcal{I}}.\, t[i] = d\} \\
(\leq k\, [\$i]R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \sharp\{t \in R_1^{\mathcal{I}} \mid t[i] = d\} \leq k\}
\end{array}
$$

Figure 1: Semantic rules for $\mathcal{DLR}$ (P, R, R_1, and R_2 have arity n)

according to the following syntax:

$$
\begin{array}{rcl}
R & ::= & \top_n \mid P \mid \$i/n\!:\!C \mid \neg R \mid R_1 \sqcap R_2 \\
C & ::= & \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]R \mid (\leq k\, [\$i]R)
\end{array}
$$

where i denotes a component of a relation, i.e., an integer between 1 and n_{max}, n denotes the *arity* of a relation, i.e., an integer between 2 and n_{max}, and k denotes a nonnegative integer.

We consider only concepts and relations that are *well-typed*, which means that (i) only relations of the same arity n are combined to form expressions of type $R_1 \sqcap R_2$ (which inherit the arity n), and (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

A $\mathcal{DLR}$ *TBox* (or *schema*) is constituted by a finite set of *inclusion assertions*, where each assertion has one of the forms:

$$
R_1 \sqsubseteq R_2 \qquad\qquad C_1 \sqsubseteq C_2
$$

with R_1 and R_2 of the same arity.

The semantics of $\mathcal{DLR}$ is specified as follows. An *interpretation* $\mathcal{I}$ is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each relation R of arity n a subset $R^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$ such that the conditions in Figure 1 are satisfied. In the figure, $t[i]$ denotes the i-th component of tuple t. Observe that, the "$\neg$" constructor on relations is used to express difference of relations, and not the complement [6].

An interpretation $\mathcal{I}$ *satisfies* an assertion $R_1 \sqsubseteq R_2$ (resp., $C_1 \sqsubseteq C_2$) if $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$ (resp., $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$). An interpretation that satisfies all assertions in a TBox $\mathcal{T}$ is called a *model* of $\mathcal{T}$. A TBox is *satisfiable* if it has a model. A relation

R (resp., concept C) is *satisfiable* in a TBox $\mathcal{T}$ if $\mathcal{T}$ admits a model $\mathcal{I}$ such that $R^{\mathcal{I}}$ (resp., $C^{\mathcal{I}}$) is nonempty. An assertion α is *logically implied* by a TBox $\mathcal{T}$ if all models of $\mathcal{T}$ satisfy α. Satisfiability and logical implication in $\mathcal{DLR}$ are EXPTIME-complete [6]. In fact, relation (resp., concept) satisfiability and logical implication are polynomially reducible to each other.

$\mathcal{DLR}$ TBoxes can capture database schemas expressed in several data models [9, 7]. For example, Entity-Relationship schemas can be represented in $\mathcal{DLR}$ by modeling each entity as a concept, and each relationship as a relation. Attributes of entities are binary relations, and single-valued attributes are modeled through the use of number restrictions. Also, single keys can be modeled through number restrictions, while multiple keys (keys constituted by more than one attribute) cannot be represented in $\mathcal{DLR}$. Attributes of relationships can be modeled in several ways, for instance through special $(n+1)$-ary relations, where n is the arity of the relationship. Finally, integrity constraints such as is-a, cardinality, existence, and typing constraints are modeled by means of inclusion assertions.

3 Key Assertions

In this section we extend $\mathcal{DLR}$ with key constraints. The resulting DL, called $\mathcal{DLR}_{key}$, allows one to express key constraints through a new kind of assertions in the TBox.

A *key assertion on a concept* has the form:

$$(\mathbf{key}\ C\ [\$i_1]R_1, \ldots, [\$i_h]R_h)$$

where C is a concept, each R_j is a relation, and each $\$i_j$ denotes one component of R_j. Intuitively, such an assertion states that no two different instances of C agree on the participation to $R_1, \ldots, R_h$. In other words, if a is an instance of C that is the i_j-th component of a tuple t_j of R_j, for $j \in \{1, \ldots, h\}$, and b is an instance of C that is the i_j-th component of a tuple s_j of R_j, for $j \in \{1, \ldots, h\}$, and for each j, t_j agrees with s_j in all components different from i_j, then a and b coincide.

A *key assertion on a relation* has the form:

$$(\mathbf{key}\ R\ \$i_1, \ldots, \$i_h)$$

where R is a relation and $\$i_1, \ldots, \$i_h$ denote the components of R that identify the whole tuple. In other words, there cannot be two different tuples in R that agree on the components $\$i_1, \ldots, \$i_h$.

We assign semantics to these assertions by defining when an interpretation satisfies them. In particular:

- An interpretation $\mathcal{I}$ *satisfies* the assertion $(\mathbf{key}\ C\ [\$i_1]R_1, \ldots, [\$i_h]R_h)$ if and only if for all $a, b \in C^{\mathcal{I}}$ and for all $t_1, s_1 \in R_1^{\mathcal{I}}, \ldots, t_h, s_h \in R_h^{\mathcal{I}}$ we have that:

$$\left.\begin{array}{l} a = t_1[i_1] = \cdots = t_h[i_h], \\ b = s_1[i_1] = \cdots = s_h[i_h], \\ t_j[i_k] = s_j[i_k], \text{ for each } j \in \{1, \ldots, h\}, \text{ and for each } k \neq i_j \end{array}\right\} \text{ implies } a = b$$

- An interpretation $\mathcal{I}$ *satisfies* the assertion $(\mathbf{key}\ R\ \$i_1, \ldots, \$i_h)$ if and only if for all $t, s \in R^{\mathcal{I}}$, we have that:

$$t[i_1] = s[i_1], \ \ldots, \ t[i_h] = s[i_h] \quad \text{implies} \quad t = s$$

Generally, in conceptual data models, if an attribute (or a relationship) A is part of a key for an entity E, then in the database schema it must be the case that E has a single-mandatory participation in A, i.e., each instance of E has exactly one associated value for A (see [1]). This limitation is not present in $\mathcal{DLR}_{key}$. Indeed, one can define an attribute (or a set of attributes) as a key of an entity even if the attribute is multi-valued, or optional. Obviously, single-valued or mandatory attributes can be modeled in $\mathcal{DLR}_{key}$ by means of number restrictions.

Example 1 Suppose that $\mathsf{Student}$ and $\mathsf{University}$ are concepts, $\mathsf{EnrolledIn}$ is a binary relation between $\mathsf{Student}$ and $\mathsf{University}$, and Code is an attribute of $\mathsf{Student}$ (a binary relation) associating to each student a unique code. Suppose that each student has a unique code within the university in which she is enrolled. Such a situation can be represented by the following $\mathcal{DLR}_{key}$ TBox:

$$\begin{array}{l} \mathsf{EnrolledIn} \ \sqsubseteq \ (\$1 : \mathsf{Student}) \sqcap (\$2 : \mathsf{University}) \\ \mathsf{Code} \ \sqsubseteq \ (\$1 : \mathsf{Student}) \sqcap (\$2 : \mathsf{String}) \\ \mathsf{Student} \ \sqsubseteq \ (\leq 1\, [\$1]\mathsf{Code}) \\ (\mathbf{key}\ \ \mathsf{Student}\ \ [\$1]\mathsf{Code}, [\$1]\mathsf{EnrolledIn}) \end{array}$$

Note that, in the conceptual modeling terminology, $\mathsf{Student}$ is a weak entity, i.e., part of its identifier is external through the relationship $\mathsf{EnrolledIn}$. ■

4 Reasoning on $\mathcal{DLR}$ with Key Assertions

We deal now with the problem of verifying logical implication in $\mathcal{DLR}_{key}$. To this end we first observe that a key assertion on a relation of the form

$$(\mathbf{key}\ R\ \$i)$$

is equivalent to the $\mathcal{DLR}$ assertion

$$\top \sqsubseteq (\leq 1\,[\$i]R)$$

Similarly, the key assertion on a concept

$$(\mathbf{key}\ C\ [\$i]R)$$

where R is a binary relation, is equivalent to the $\mathcal{DLR}$ assertion

$$\top \sqsubseteq (\leq 1\,[\$j](R \sqcap \$i\!:\!C))$$

where $j = 2$ if $i = 1$, and $j = 1$ if $i = 2$. Hence, in the following, without loss of generality, we will not consider key assertions of the above form.

We first focus on verifying whether an inclusion assertion (which does not involve keys) is logically implied by a $\mathcal{DLR}_{key}$ TBox. Let $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ be a $\mathcal{DLR}_{key}$ TBox, where $\mathcal{L}$ is the set of inclusion assertions in $\mathcal{T}$ and $\mathcal{K}$ is the set of key assertions in $\mathcal{T}$.

Theorem 2 *A $\mathcal{DLR}_{key}$ TBox $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ logically implies an inclusion assertion $L_1 \sqsubseteq L_2$ if and only if $\mathcal{L}$ logically implies $L_1 \sqsubseteq L_2$.*

Proof (sketch). We show that the concept (resp., relation) $L_1 \sqcap \neg L_2$ is satisfiable in $\mathcal{L}$ iff it is satisfiable in $\mathcal{L} \cup \mathcal{K}$.

"$\Leftarrow$" Trivial.

"$\Rightarrow$" It is possible to show that $\mathcal{DLR}$ has the tree-model property [6], i.e., if a TBox admits a model satisfying a concept, it also admits a model satisfying the concept which has the structure of a tree, considering reified tuples as nodes. On such models a key assertion is always satisfied. Observe that key assertions that are equivalent to number restrictions are dealt with directly in $\mathcal{DLR}$. $\square$

Theorem 2 shows that key assertions do not interact with inclusion assertions. We consider now logical implication of key assertions. To this end we introduce a generalized form of $\mathcal{DLR}$ ABox. We make use of *Skolem constants (sk-constants)*. Intuitively, a sk-constant denotes an individual in an interpretation, in such a way that different sk-constants may denote the same individual.

A (generalized) $\mathcal{DLR}$ ABox is constituted by a finite set of assertions of the following types:

$$C(x), \qquad R(t), \qquad x \neq y, \qquad r \neq s$$

where x and y are sk-constants, t is a tuple of sk-constants of the same arity as that of R, and r and s are two tuples of sk-constants of the same arity. The notion of interpretation is extended so as to assign to each sk-constant x an individual $x^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$

- satisfies $C(x)$ if $x^{\mathcal{I}} \in C^{\mathcal{I}}$,

- it satisfies $R(x_1, \ldots, x_n)$ if $(x_1^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}) \in R^{\mathcal{I}}$,
- it satisfies $x \neq y$ if $x^{\mathcal{I}} \neq y^{\mathcal{I}}$, and
- it satisfies $r \neq s$ if for at least one component i, $(r[i])^{\mathcal{I}} \neq (s[i])^{\mathcal{I}}$.

If $\mathcal{T}$ is a $\mathcal{DLR}$ TBox, and $\mathcal{A}$ is a $\mathcal{DLR}$ ABox of the above form, then $\langle \mathcal{T}, \mathcal{A} \rangle$ is called a $\mathcal{DLR}$ knowledge base. An interpretation satisfies $\langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies every assertion in $\mathcal{T} \cup \mathcal{A}$. It follows from the results in [6], that checking a $\mathcal{DLR}$ knowledge base for satisfiability is EXPTIME-complete.

Given a $\mathcal{DLR}_{key}$ TBox $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ and a key assertion

$$\kappa \ = \ (\mathbf{key}\, C\ [\$i_1]R_1, \ldots, [\$i_h]R_h)$$

we define $\mathcal{A}(\mathcal{K}, \kappa)$ to be a $\mathcal{DLR}$ ABox that contains:

- $C(x)$, $C(y)$, $x \neq y$;
- $R_j(t_j)$, $R_j(s_j)$, with $j \in \{1, \ldots, h\}$, where t_j and s_j are tuples whose arity is that of R_j with $t_j[i_j] = x$, $s_j[i_j] = y$, and $t_j[i] = s_j[i]$ for $i \neq i_j$;
- for each key assertion in $\mathcal{K}$ of the form $(\mathbf{key}\, C'\ [\$i'_1]R'_1, \ldots, [\$i'_h]R'_{h'})$
  - either $C'(x)$ or $\neg C'(x)$,
  - either $C'(y)$ or $\neg C'(y)$,
  - either $R'_{j'}(t_j)$ or $\neg R'_{j'}(t_j)$, for each $j \in \{1, \ldots, h\}$ and each $j' \in \{1, \ldots, h'\}$ such that the arity of $R'_{j'}$ equals that of t_j,
  - either $R'_{j'}(s_j)$ or $\neg R'_{j'}(s_j)$, for each $j \in \{1, \ldots, h\}$ and each $j' \in \{1, \ldots, h'\}$ such that the arity of $R'_{j'}$ equals that of s_j;
- for each key assertion in $\mathcal{K}$ of the form $(\mathbf{key}\, R\ \$i'_1, \ldots, \$i'_{h'})$
  - either $R(t_j)$ or $\neg R(t_j)$, for each $j \in \{1, \ldots, h\}$ such that the arity of R equals that of t_j,
  - either $R(s_j)$ or $\neg R(s_j)$, for each $j \in \{1, \ldots, h\}$ such that the arity of R equals that of s_j.

Similarly, given $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ and a key assertion

$$\kappa \ = \ (\mathbf{key}\, R\ \$i_1, \ldots, \$i_h)$$

we define $\mathcal{A}(\mathcal{K}, \kappa)$ to be a $\mathcal{DLR}$ ABox that contains:

- $R(t)$, $R(s)$, $t \neq s$, where t and s are tuples whose arity is that of R and such that $t[i_j] = s[i_j]$, for $j \in \{1, \ldots, h\}$;

- for each key assertion of the form (**key** R' $\$i'_1, \ldots, \$i'_{h'}$) where the arity of R' equals that of R
  - either $R'(t)$ or $\neg R'(t)$, and
  - either $R'(s)$ or $\neg R'(s)$.

Note that, given $\mathcal{K}$ and κ, there are many (actually, an exponential number) different ABoxes $\mathcal{A}(\mathcal{K}, \kappa)$, one for each possible choice in the items above. Intuitively, an ABox $\mathcal{A}(\mathcal{K}, \kappa)$ has the following features:

- it violates the key assertion κ,
- it allows to immediately verify whether it *violates* a key assertion in $\mathcal{K}$. Indeed, for all objects and tuples appearing in $\mathcal{A}(\mathcal{K}, \kappa)$, membership or non-membership in the relevant relations and concepts appearing in key assertions that could be violated is explicitly asserted. Hence, it suffices to verify whether the semantic condition of the key assertion is violated, considering relations and concepts appearing in the key assertion as primitives.

The following theorem shows that keys are "for free" in $\mathcal{DLR}$, in the sense that reasoning in $\mathcal{DLR}_{key}$ can be reduced to reasoning in $\mathcal{DLR}$ knowledge bases.

Theorem 3 *A $\mathcal{DLR}_{key}$ TBox $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ does not logically imply a key assertion κ if and only if there exists one $\mathcal{A}(\mathcal{K}, \kappa)$ that satisfies all key assertions in $\mathcal{K}$ and such that the $\mathcal{DLR}$ knowledge base $\langle \mathcal{L}, A(\mathcal{K}, \kappa) \rangle$ is satisfiable.*

Proof (sketch). "$\Leftarrow$" Assume that there exists one $\mathcal{A}(\mathcal{K}, \kappa)$ that satisfies all key assertions in $\mathcal{K}$ and such that $\langle \mathcal{L}, A(\mathcal{K}, \kappa) \rangle$ is satisfiable. Then, by exploiting the tree model property, one can easily construct a model of $\mathcal{L} \cup \mathcal{K}$ where κ is violated.

"$\Rightarrow$" Assume that $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ does not logically imply the key assertion κ. Then there is a model of $\mathcal{T}$ where κ is violated. But then such a model would be a model of $\langle \mathcal{L}, A(\mathcal{K}, \kappa) \rangle$ for some $\mathcal{A}(\mathcal{K}, \kappa)$. □

We can now state the theorem showing that reasoning on a $\mathcal{DLR}$ TBox with key assertions can be done with the same worst-case computational complexity as for the case without keys.

Theorem 4 *Satisfiability and logical implication in $\mathcal{DLR}_{key}$ are EXPTIME-complete.*

Proof (sketch). By Theorem 2, satisfiability of a TBox, satisfiability of a concept or relation in a TBox, and logical implication of inclusion assertions in $\mathcal{DLR}_{key}$ reduce to the corresponding problems in $\mathcal{DLR}$. By Theorem 3, logical implication of a key assertion κ from a $\mathcal{DLR}_{key}$ TBox $\mathcal{T} = \mathcal{L} \cup \mathcal{K}$ reduces to solving a (possibly exponential) number of tests, where each test involves one $\mathcal{A}(\mathcal{K}, \kappa)$ and consists in directly verifying all key assertions in $\mathcal{K}$ and checking the satisfiability of the $\mathcal{DLR}$ knowledge base $\langle \mathcal{L}, A(\mathcal{K}, \kappa) \rangle$. Note that the size of each $\langle \mathcal{L}, A(\mathcal{K}, \kappa) \rangle$ is polynomial in the size of $\mathcal{T} \cup \kappa$. $\square$

5 Conclusions

We have shown how to add keys to $\mathcal{DLR}$, while still retaining the possibility of reasoning over schemas in EXPTIME. We believe that the approach presented in this paper can be extended in several ways. For example, our technique can be directly applied to the case where the $\mathcal{DLR}$ knowledge base contains an ABox, or to the case where $\mathcal{DLR}_{reg}$ is used instead of $\mathcal{DLR}$. Moreover, we are working on the following extensions: (i) using chaining in specifying keys, in the spirit of [3], (ii) functional dependencies on relations, (iii) query containment and query answering using views in the presence of both key constraints, and functional dependencies.

References

[1] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California, 1992.

[2] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[3] A. Borgida and G. E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proc. of the 5th Int. Conf. on Deductive and Object-Oriented Databases (DOOD'97)*, pages 85–102, 1997.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD'95)*, volume 1013 of *Lecture Notes in Computer Science*, pages 229–246. Springer-Verlag, 1995.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive query containment in Description Logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL'97)*, pages 5–9, 1997.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT

SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), pages 149–158, 1998.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[9] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publisher, 1998.

[10] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[11] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[12] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. Query containment using a DLR ABox. Technical Report LTCS-Report 99-15, RWTH Aachen, 1999.

[13] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.