Identification Constraints and Functional Dependencies 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. 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 identification constraints and functional dependencies. In this paper we introduce a DL which extends $\mathcal{DLR}$ and fully captures the semantics of such constraints, and we address the problem of reasoning in such a logic. We show that, verifying knowledge base satisfiability and logical implication in the presence of identification constraints and nonunary functional dependencies can be done in EXPTIME, thus with the same worst-case computational complexity as for plain $\mathcal{DLR}$. We also show that adding just unary functional dependencies to $\mathcal{DLR}$ leads to undecidability.

1 Introduction

In the last years, Description Logics (DLs) have been successfully applied to data management [Borgida, 1995; Kirk *et al.*, 1995; Calvanese *et al.*, 1998b; 1999]. 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 [Calvanese *et al.*, 1998a; 1998b], a very expressive DL with n-ary relations, called $\mathcal{DLR}$, is introduced, and it is shown how database schemas can be captured by this logic. Also, suitable mechanisms for expressing queries over $\mathcal{DLR}$ schemas have been added, and techniques for reasoning over queries have been designed. Notably, the investigation on $\mathcal{DLR}$ has led to the design of new DL systems effectively implementing powerful reasoning techniques [Horrocks *et al.*, 1999].

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 identification constraints and functional dependencies. Identification constraints (also called keys) are used to state that a certain set of properties uniquely identifies the instances of a concept. A functional dependency on a relation is used to impose that a combination of a given set of attributes functionally determines another attribute of the relation. It is easy to see that functional dependencies can be used to model keys of a relation, i.e., attributes that are sufficient to identify tuples of the relation. Both types of constraints are commonly used in database design and data management.

The question addressed in this paper is as follows: can we add identification constraints and *non-unary* functional dependencies 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 both types of constraints in $\mathcal{DLR}$. In particular, we adapt the $\mathcal{DLR}$ reasoning algorithm in such a way that reasoning on a $\mathcal{DLR}$ schema with both types of constraints and with ABoxes, can be done with the same worst-case computational complexity as for the case of plain $\mathcal{DLR}$. Also, the proposed technique can be incorporated into present DL systems, such as the one described in [Horrocks *et al.*, 1999]. We also show that adding to $\mathcal{DLR}$ *unary* functional dependencies leads to undecidability of reasoning. Observe, however, that the presence of such functional dependencies is typically considered as an indication of bad schema design in databases.

Both identification constraints and functional dependencies have been extensively investigated in the database literature (see [Abiteboul *et al.*, 1995], Chapters 8, 9). However, database models lack the kinds of constraints expressible in expressive DLs, and therefore none of the results developed in the context of databases can be used to solve our problem.

In the last years, there have been some attempts to add identification constraints to DLs. In [Calvanese *et al.*, 1995], these constraints 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. The limitation of this approach is that several interesting semantic properties of keys are not represented in the knowledge base. In [Borgida and Weddell, 1997; Khizder *et al.*, 2001], a general mechanism is proposed for modeling path functional dependencies, and a sound and complete inference system for reasoning on such constraints is presented. Path functional dependencies are sufficiently expressive to model both identification constraints and functional dependencies. How-

ever, the DLs considered are limited in expressiveness. In particular, union, negation, number restrictions, general inclusion axioms, inverse roles and n-ary relations are not part of the considered language, and therefore useful properties of database schemas cannot be represented. The proposal presented in this paper fully captures the semantics of both identification constraints and functional dependencies in an expressive DL with all the above features.

The paper is organized as follows. In Section 2, we recall the DL $\mathcal{DLR}$. In Section 3, we illustrate the mechanism for specifying identification constraints and functional dependencies in $\mathcal{DLR}$ knowledge bases. In Section 4, we discuss the modeling power of the resulting logic, called $\mathcal{DLR}_{ifd}$. In Section 5, we describe how we can extend the $\mathcal{DLR}$ reasoning technique in order to take the new types of constraints into account, and in Section 6 we show that minor extensions of $\mathcal{DLR}_{ifd}$ leads to undecidability of reasoning. Finally, Section 7 concludes the paper.

2 Description Logic $\mathcal{DLR}$

We focus on the Description Logic $\mathcal{DLR}$, which is able to capture a great variety of data models with many forms of constraints [Calvanese *et al.*, 1998a; 1999]. 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 (having arity between 2 and n_{max}) and atomic concepts, denoted by P and A, respectively. We use R to denote arbitrary relations and C to denote arbitrary concepts, respectively built according to the following syntax:

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

where n denotes the *arity* of the relations P, R, R_1, and R_2, i denotes a component of a relation, i.e., an integer between 1 and n, and k denotes a non-negative integer. Observe that 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); (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

We introduce the following abbreviations: $\bot$ for $\neg\top_1$; $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$; $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$; $(\geq k\,[i]R)$ for $\neg(\leq k{-}1\,[i]R)$; $\exists[i]R$ for $(\geq 1\,[i]R)$; $\forall[i]R$ for $\neg\exists[i]\neg R$. Moreover, we abbreviate $(i/n:C)$ with $(i:C)$ when n is clear from the context.

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

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

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 [Calvanese *et al.*, 1998a]. An interpretation $\mathcal{I}$ *satisfies* an assertion $C_1 \sqsubseteq C_2$ (resp., $R_1 \sqsubseteq R_2$) if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ (resp., $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$).

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

Figure 1: Semantic rules for $\mathcal{DLR}$ (P, R, R_1, and R_2 have arity n, and $\sharp\sigma$ denotes the cardinality of the set σ)

We introduce a generalized form of $\mathcal{DLR}$ ABox. We consider an alphabet of new symbols, called *Skolem constants (sk-constants)*. Intuitively, an 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* (or simply ABox in the following) is constituted by a finite set of assertions, called ABox assertions, of the following types:

$$C(x) \qquad R(x_1, \ldots, x_n) \qquad x \neq y \qquad x = y$$

where R is a relation of arity n, and x, y, $x_1, \ldots, x_n$ are sk-constants.

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 $x^{\mathcal{I}} \in C^{\mathcal{I}}$;
- $R(x_1, \ldots, x_n)$ if $(x_1^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}) \in R^{\mathcal{I}}$;
- $x \neq y$ if $x^{\mathcal{I}} \neq y^{\mathcal{I}}$;
- $x = y$ if $x^{\mathcal{I}} = y^{\mathcal{I}}$.

If $\mathcal{T}$ is a $\mathcal{DLR}$ TBox, and $\mathcal{A}$ is a $\mathcal{DLR}$ ABox of the above form, then $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ is called a *$\mathcal{DLR}$ knowledge base*. An interpretation is a *model* of $\mathcal{K}$ if it satisfies every assertion in $\mathcal{K}$. A knowledge base $\mathcal{K}$ is *satisfiable* if it has a model. An assertion α (either an inclusion, or an ABox assertion) is *logically implied* by $\mathcal{K}$ if all models of $\mathcal{K}$ satisfy α.

Logical implication and knowledge base satisfiability are mutually reducible to each other. For one direction, $\mathcal{K}$ is unsatisfiable iff $\mathcal{K} \models \top_1 \sqsubseteq \bot$. For the converse direction, it is possible to show that $\mathcal{K} \models C_1 \sqsubseteq C_2$ iff $\mathcal{K} \cup \{\top_1 \sqsubseteq \exists[1](P \sqcap (2:C_1 \sqcap \neg C_2))\}$ is unsatisfiable, where P is a new binary relation. Similarly, $\mathcal{K} \models R_1 \sqsubseteq R_2$ iff $\mathcal{K} \cup \{\top_1 \sqsubseteq \exists[1](P \sqcap (2:\exists[1](C_1 \sqcap \neg C_2)))\}$ is unsatisfiable, where again P is a new binary relation. Finally, $\mathcal{K} \models C(\alpha)$ iff $\mathcal{K} \cup \{\neg C(\alpha)\}$ is unsatisfiable.

It follows from the results in [Calvanese *et al.*, 1998a], that checking a $\mathcal{DLR}$ knowledge base for satisfiability is EXPTIME-complete.

3 Identification and Functional Dependency Assertions

We extend $\mathcal{DLR}$ with identification constraints and functional dependencies. The resulting DL, called $\mathcal{DLR}_{ifd}$, allows one to express these constraints through new kinds of assertions in the TBox.

An *identification assertion* on a concept has the form:

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

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 two instances of C cannot agree on the participation to $R_1, \ldots, R_h$ via components $i_1, \ldots, i_h$, respectively.

A *functional dependency assertion* on a relation has the form:

$$(\mathbf{fd}\ R\ i_1, \ldots, i_h \rightarrow j)$$

where R is a relation, $h \geq 2$, and $i_1, \ldots, i_h, j$ denote components of R. The assertion imposes that two tuples of R that agree on the components $i_1, \ldots, i_h$, agree also on the component j.

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

- An interpretation $\mathcal{I}$ *satisfies* the assertion $(\mathbf{id}\ C\ [i_1]R_1, \ldots, [i_h]R_h)$ 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:

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

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

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

A $\mathcal{DLR}_{ifd}$ knowledge base is a set $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ of assertions, where $\mathcal{T} \cup \mathcal{A}$ is a $\mathcal{DLR}$ knowledge base and $\mathcal{F}$ is a set of identification and functional dependency assertions.

Note that unary functional dependencies (i.e., functional dependencies with $h = 1$) are ruled out in $\mathcal{DLR}_{ifd}$. We will come to this in Section 6. Note also that the right hand side of a functional dependency contains a single element. However, this is not a limitation, because any functional dependency with more than one element in the right hand side can always be split into several dependencies of the above form. Also, to verify whether a functional dependency with more than one element in the right hand side is logically implied by a $\mathcal{DLR}_{ifd}$ knowledge base, it suffices to verify whether each of the functional dependencies in which it can be split, is logically implied by the knowledge base.

4 Modeling in $\mathcal{DLR}_{ifd}$

$\mathcal{DLR}_{ifd}$ captures database schemas expressed in several data models . For example, Entity-Relationship schemas can be represented already in $\mathcal{DLR}$, by modeling each entity as a concept, and each relationship as a relation [Calvanese *et al.*, 1998b; 1999]. Attributes of entities are modeled by means of binary relations, and single-valued or mandatory attributes are expressible through the use of number restrictions. 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. Also, integrity constraints such as is-a, cardinality, existence, and typing constraints are expressible by means of inclusion assertions. Finally, unary keys (keys constituted by a single attribute) can be modeled through number restrictions. Non-unary keys cannot be represented in $\mathcal{DLR}$, while they are obviously expressible in $\mathcal{DLR}_{ifd}$.

Example 1 Suppose that Person and University are concepts, EnrolledIn is a binary relation between Person and University, and StudentCode is an (optional) attribute (modeled as a binary relation) of Person associating to each student (a person who is enrolled in a university) a code that is unique in the context of the university in which she is enrolled. Such a situation can be represented by the following $\mathcal{DLR}_{ifd}$ TBox:

$$\begin{array}{rcl} \mathsf{EnrolledIn} & \sqsubseteq & (1:\mathsf{Person}) \sqcap (2:\mathsf{University}) \\ \mathsf{StudentCode} & \sqsubseteq & (1:\mathsf{Person}) \sqcap (2:\mathsf{String}) \\ \mathsf{Person} & \sqsubseteq & (\leq 1\ [1]\mathsf{StudentCode}) \end{array}$$

$$(\mathbf{id}\ \mathsf{Person}\ [1]\mathsf{StudentCode}, [1]\mathsf{EnrolledIn})$$

Note that, the notion of student is modeled by the concept $\mathsf{Person} \sqcap \exists[1]\mathsf{EnrolledIn}$ and, in the conceptual modeling terminology, this concept is a weak entity, i.e., part of its identifier is external through the relationship EnrolledIn.

We additionally want to model the notion of exam in our application. An exam is a relationship involving a student, a course, a professor, and a grade. In an exam, the combination of student and course functionally determines both the professor, and the grade. This can be represented by adding the following assertions to the TBox:

$$\begin{array}{l} \mathsf{Exam} \sqsubseteq (1:(\mathsf{Person} \sqcap \exists[1]\mathsf{EnrolledIn})) \sqcap \\ \qquad (2:\mathsf{Course}) \sqcap (3:\mathsf{Person}) \sqcap (4:\mathsf{Grade}) \end{array}$$

$$(\mathbf{fd}\ \mathsf{Exam}\ 1, 2 \rightarrow 3) \qquad (\mathbf{fd}\ \mathsf{Exam}\ 1, 2 \rightarrow 4)$$ ∎

Observe that generally, in conceptual data models, if an attribute (or a relationship) L is part of a key for an entity E, then in the database schema it must be the case that E has a single and mandatory participation in L, i.e., each instance of E has exactly one associated value for L [Abiteboul *et al.*, 1995]. This is not required in $\mathcal{DLR}_{ifd}$ (but can be asserted when needed), where one can define an attribute as part of a key of an entity, even if the attribute is multi-valued or optional.

We have mentioned that unary functional dependencies are not allowed in $\mathcal{DLR}_{ifd}$. However, this limitation does not prevent one from defining unary keys for relations. Indeed, the fact that component i is a key for the relation R can already be expressed in $\mathcal{DLR}$ by means of the assertion:

$$\top_1 \sqsubseteq (\leq 1\ [i]R)$$

The above observation also implies that functional dependencies in the context of binary relations, which are by definition unary, are expressible in $\mathcal{DLR}_{ifd}$. Indeed, such functional dependencies correspond to key constraints, which are

expressible as specified above. For example, the functional dependency $1 \rightarrow 2$ in the context of the binary relation R can be expressed by specifying that component 1 is a key for R. Thus, the only functional dependencies that are not admitted in $\mathcal{DLR}_{ifd}$ are unary functional dependencies in the context of non-binary relations. This is because they lead to undecidability of reasoning, as shown in Section 6. Note also, that the presence of such functional dependencies is considered as an indication of bad design in the framework of the relational data model (see [Abiteboul *et al.*, 1995], Chapter 11). In fact, a unary functional dependency in the context of an n-ary relation (with $n > 2$) represents a hidden relationship between the arguments of the relation, which may cause several modeling problems.

The possibility of defining identification constraints substantially enriches the modeling power of DLs. In particular, it is possible to show that, even if only binary relations are allowed in a DL, then the use of identification constraints permits simulating the presence of n-ary relations in such a logic. For example, a relation with arity 3 can be modeled by means of a concept and 3 binary relations. Number restrictions are used to state that every instance of the concept participates in exactly one instance of the binary relation, and a suitable identification assertion states that the combination of the three binary relations form a key for the concept. Obviously, $\mathcal{DLR}_{ifd}$ further increases the modeling power by allowing the explicit use of n-ary relations, and the possibility of imposing functional dependencies in the context of relations.

5 Reasoning on $\mathcal{DLR}_{ifd}$

First of all we observe that, when reasoning in $\mathcal{DLR}_{ifd}$, identification assertions of the form $(\mathbf{id}\ C\ [i]R)$, where R is a *binary* relation, are equivalent to $\mathcal{DLR}$ assertions $\top \sqsubseteq (\leq 1\,[j](R \sqcap (i\!:\!C)))$, where $j = 2$ if $i = 1$, and $j = 1$ if $i = 2$. Hence, in the following, without loss of generality, we will not consider such identification assertions.

Next we show that we can reduce logical implication in $\mathcal{DLR}_{ifd}$ to knowledge base satisfiability. As already observed in Section 2, logical implication of inclusion and ABox assertions can be reduced to knowledge base satisfiability. We show that the same can be done also for identification and functional dependency assertions.

Given an identification assertion

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

we define the ABox $\mathcal{A}_\kappa$ constituted by the following assertions:

- $C(x)$, $C(y)$, and $x \neq y$, where x and y are new sk-constants;
- $R_j(t_j)$ and $R_j(s_j)$, with $j \in \{1, \ldots, h\}$, where t_j and s_j are tuples of new sk-constants with $t_j[i_j] = x$, $s_j[i_j] = y$, and $t_j[i] = s_j[i]$ for $i \neq i_j$.

Similarly, for a functional dependency assertion

$$\kappa \;=\; (\mathbf{fd}\ R\ i_1, \ldots, i_h \rightarrow j)$$

we define $\mathcal{A}_\kappa$ constituted by the following assertions:

- $R(t)$, $R(s)$, and $t[j] \neq s[j]$, where t and s are tuples of new sk-constants with $t[i_j] = s[i_j]$, for $j \in \{1, \ldots, h\}$.

From the semantics of identification and functional dependency assertions it is immediate to see that A_κ provides a concrete counterexample to κ. Hence it follows that, given a $\mathcal{DLR}_{ifd}$ knowledge base $\mathcal{K}$, $\mathcal{K} \models \kappa$ if and only if $\mathcal{K} \cup \mathcal{A}_\kappa$ is unsatisfiable.

Theorem 2 *Logical implication in $\mathcal{DLR}_{ifd}$ can be reduced to knowledge base satisfiability.*

We now present a reasoning procedure for knowledge base satisfiability in $\mathcal{DLR}_{ifd}$.

$\mathcal{DLR}_{ifd}$ TBoxes (which in fact are $\mathcal{DLR}$ TBoxes since they do not include identification and functional dependency assertions) have the tree-model property [Calvanese *et al.*, 1998a], which is true for most DLs. In particular, if a $\mathcal{DLR}$ TBox admits a model, it also admits a model which has the structure of a tree, where nodes are either objects or (reified) tuples, and edges connect tuples to their components. Observe that in such models identification and functional dependency assertions (which in $\mathcal{DLR}_{ifd}$ are non-unary) are trivially satisfied, since there cannot be two tuples agreeing on more than one component. As an immediate consequence we have that, given a $\mathcal{DLR}_{ifd}$ TBox $\mathcal{T}$ and a set of identification and functional dependency assertions $\mathcal{F}$, $\mathcal{T} \cup \mathcal{F}$ is satisfiable iff $\mathcal{T}$ is so. This implies that, in absence of an ABox, logical implication of inclusion assertions can be verified without considering identification and functional dependency assertions at all.

When we add an ABox, then we may still restrict the attention to models that have the structure of a tree, except for a cluster of objects representing the sk-constants in the ABox (see again [Calvanese *et al.*, 1998a][1]). We call such models *clustered tree models*. On such models, identification and functional dependency assertions are always satisfied, except possibly for the cluster of sk-constants. Hence we can concentrate on verifying such assertions on the objects and tuples appearing in the ABox only.

Given a $\mathcal{DLR}_{ifd}$ knowledge base $\mathcal{K}$, we define a *saturation* of $\mathcal{K}$ as an ABox A_s constructed as follows:

- for each sk-constant x occurring in $\mathcal{K}$, and for each identification assertion $(\mathbf{id}\ C\ [i_1]R_1, \ldots, [i_h]R_h)$ in $\mathcal{K}$, A_s contains either $C(x)$ or $\neg C(x)$;
- for each tuple t of sk-constants occurring in an assertion $R(t)$ of $\mathcal{K}$,
  - for each identification assertion $(\mathbf{id}\ C\ [i_1]R_1, \ldots, [i_h]R_h)$ in $\mathcal{K}$, for each R_j ($j \in \{1, \ldots, h\}$) having the arity of R, A_s contains either $R_j(t)$ or $\neg R_j(t)$,
  - for each functional dependency assertion $(\mathbf{fd}\ R'\ i_1, \ldots, i_h \rightarrow j)$ in $\mathcal{K}$, such that R' has the arity of R, A_s contains either $R'(t)$ or $\neg R'(t)$;

[1] In fact, [Calvanese *et al.*, 1998a] makes use of inclusion assertions involving nominals, i.e., concepts having a single instance. It is immediate to verify that such inclusion assertions correspond to the kind of generalized ABoxes we adopt here.

- for each pair of sk-constants x and y occurring in $\mathcal{K}$, A_s contains either $x = y$ or $x \neq y$.

Note that the size of a saturation is polynomial in the size of $\mathcal{K}$. Note also that there are many (actually, an exponential number) of different saturations of $\mathcal{K}$, one for each possible set of choices in the items above.

On a saturation one can immediately verify whether an identification or functional dependency assertion of $\mathcal{K}$ is *violated*. Indeed, for all sk-constants and tuples of sk-constants appearing in the saturation, membership or non-membership in the relevant relations and concepts appearing in the assertions that could be violated is explicitly asserted (after substituting each sk-constant with a representative of its equivalence class according to the equalities). Hence, it suffices to verify whether the semantic condition of the assertion is violated, considering relations and concepts appearing in the assertion as primitives. Once such a check on the identification and functional dependencies is done, it remains to verify whether $\mathcal{A}_s$ is consistent with the other assertions of $\mathcal{K}$.

Theorem 3 *A $\mathcal{DLR}_{ifd}$ knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ is satisfiable if and only if there exists a saturation $\mathcal{A}_s$ of $\mathcal{K}$ that does not violate the identification and functional dependency assertions in $\mathcal{K}$ and such that the $\mathcal{DLR}$ knowledge base $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ is satisfiable.*

Proof (sketch). "$\Leftarrow$" Assume that there exists a saturation $\mathcal{A}_s$ that does not violate the identification and functional dependency assertions in $\mathcal{K}$ and such that $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ is satisfiable. Then there exists a clustered tree model of $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$, and such interpretation is also a model of $\mathcal{K}$.

"$\Rightarrow$" Assume that $\mathcal{K}$ is satisfiable. Then from a model $\mathcal{I}$ of $\mathcal{K}$ one can directly construct a saturation for which $\mathcal{I}$ satisfies all assertions. □

The above result provides us with an upper bound for reasoning in $\mathcal{DLR}_{ifd}$, matching the lower bound holding already for $\mathcal{DLR}$.

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

Proof (sketch). By Theorem 2, logical implication in $\mathcal{DLR}_{ifd}$ reduces to knowledge base satisfiability in $\mathcal{DLR}_{ifd}$. By Theorem 3, satisfiability of a $\mathcal{DLR}_{ifd}$ knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A} \cup \mathcal{F}$ reduces to solving a (possibly exponential) number of tests, where each test involves one saturation A_s and consists in directly verifying all identification and functional dependency assertions in $\mathcal{A}_s$ (a polynomial step) and checking the satisfiability of the $\mathcal{DLR}$ knowledge base $\mathcal{T} \cup \mathcal{A} \cup \mathcal{A}_s$ (an exponential step). □

6 Unary functional dependencies in $\mathcal{DLR}_{ifd}$

One might wonder whether the method described in the previous works also for unary functional dependencies. Actually, this is not the case since unary functional dependencies are *not trivially satisfied* in tree structured interpretations. For example, it may happen that two tuples of a relation R agree on say component 1 and not component 2, and therefore violate the functional dependency (**fd** $R\ 1 \rightarrow 2$).

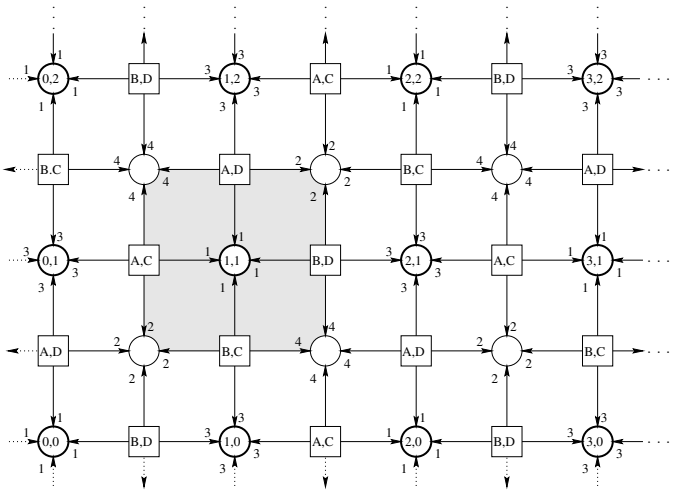

Figure 2: Grid structure enforced by $\mathcal{K}_T$

Indeed, we show that if we allow for unary functional dependencies, then reasoning in $\mathcal{DLR}_{ifd}$ becomes undecidable. To do so we exhibit a reduction from the unconstrained quadrant *tiling problem* [van Emde Boas, 1997], which consists in deciding whether the first quadrant of the integer grid can be tiled using a finite set of square tile types in such a way that adjacent tiles respect adjacency conditions. Tiling problems are well suited to show undecidability of variants of description and dynamic logics [van Emde Boas, 1997; Baader and Sattler, 1999]. The crux of the undecidability proof consists in enforcing that the tiles lie on an integer grid. Once the grid structure is enforced, it is typically easy to impose the adjacency conditions on the tiles. In our case we exploit unary functional dependencies to construct the grid.

Formally, a *tiling system* is a triple $T = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ where $\mathcal{D}$ is a finite set of elements representing tile types and $\mathcal{H}$ and $\mathcal{V}$ are two binary relations over $\mathcal{D}$. The *unconstrained quadrant tiling problem* consists in verifying the existence of a tiling *consistent with* T, i.e., a mapping τ from $\mathbb{N} \times \mathbb{N}$ to $\mathcal{D}$ such that $(\tau(i,j), \tau(i+1,j)) \in \mathcal{H}$ and $(\tau(i,j), \tau(i,j+1)) \in \mathcal{V}$, for $i, j \in \mathbb{N}$. Such a problem is undecidable, more precisely Π_0^0-complete [Berger, 1966; van Emde Boas, 1997].

From a tiling system $T = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ we construct a $\mathcal{DLR}_{ifd}$ knowledge base $\mathcal{K}_T$ as follows. The basic idea is to enforce the grid structure shown in Figure 2, where squares represent tuples of arity 4 and circles represent objects. Each object depicted using a bold circle and labeled i, j represents the node (i, j) of the grid. Numbers labeling arrows represent components of tuples. A pair of letters X, Y inside a tuple represents the fact that the tuple is an instance of the relations X and Y. In particular we use four relations of arity 4: A, B, C, and D. The gray box in the figure represents how a tile would be placed in such a grid.

We enforce the grid structure by means of the following assertions in $\mathcal{K}_T$:

$$\exists[i]\top_4 \sqsubseteq \exists[i](A \sqcap C) \sqcap \exists[i](A \sqcap D) \sqcap \exists[i](B \sqcap C) \sqcap \exists[i](B \sqcap D) \quad \text{for } i \in \{1, \ldots, 4\}$$

$(\mathbf{fd}\ A\ 2 \rightarrow 3)\ (\mathbf{fd}\ B\ 2 \rightarrow 3)\ (\mathbf{fd}\ C\ 2 \rightarrow 1)\ (\mathbf{fd}\ D\ 1 \rightarrow 2)$
$(\mathbf{fd}\ A\ 4 \rightarrow 1)\ (\mathbf{fd}\ B\ 4 \rightarrow 1)\ (\mathbf{fd}\ C\ 3 \rightarrow 4)\ (\mathbf{fd}\ D\ 4 \rightarrow 3)$

We enforce the adjacency conditions on the tiles of the first quadrant by using one concept for each tile type in $\mathcal{D}$ and introducing in $\mathcal{K}_T$ the following assertions: for each $D_i \in \mathcal{D}$

$$\begin{aligned} D_i \sqsubseteq\ & \forall[1](B \sqcap D \Rightarrow (3 : \bigsqcup_{(D_i,D_j)\in\mathcal{H}} D_j)) \sqcap \\ & \forall[3](A \sqcap C \Rightarrow (1 : \bigsqcup_{(D_i,D_j)\in\mathcal{H}} D_j)) \sqcap \\ & \forall[1](A \sqcap D \Rightarrow (3 : \bigsqcup_{(D_i,D_j)\in\mathcal{V}} D_j)) \sqcap \\ & \forall[3](B \sqcap C \Rightarrow (1 : \bigsqcup_{(D_i,D_j)\in\mathcal{V}} D_j)) \end{aligned}$$

Finally, to represent the origin of the tiling we use the concept $C_0 = (\exists[1]\top_4) \sqcap \bigsqcup_{D_i\in\mathcal{D}} D_i$ Then the tiling problem associated to T admits a solution if and only if $\mathcal{K}_T \not\models C_0 \sqsubseteq \bot$. Indeed, from a tiling consistent with T one obtains immediately a model of $\mathcal{K}_T$ with an object satisfying C_0. Conversely, from a model $\mathcal{I}$ of $\mathcal{K}_T$ with $C_0^{\mathcal{I}} \neq \emptyset$, we can construct a tiling consistent with T. The first set of assertions impose that a portion of $\mathcal{I}$ has exactly the structure depicted in Figure 2 (observe that not the whole model necessarily has a grid structure, but only a portion corresponding to the first quadrant). The second set of assertions impose on such a portion only that instances of concepts representing tile types respect the adjacency conditions. As a consequence of such a reduction we obtain the following result.

Theorem 5 *Knowledge base satisfiability (and thus logical implication) in $\mathcal{DLR}_{ifd}$ extended with unary functional dependencies is undecidable.*

The reduction above can be easily modified to show that, if we allow for *nominals* [Tobies, 2000], then even n-ary functional dependencies lead to undecidability. For example, we may use one nominal o and relations of arity 5 instead of 4. Then we can force the fifth component of all tuples to be the object o by means of the assertion $\top_5 \sqsubseteq (5 : o)$ and we can enforce the grid structure as above, by adding component 5 to the antecedent of the functional dependencies above (thus getting binary functional dependencies). Observe that, since all tuples agree on component 5, such binary functional dependencies are actually mimicking the unary dependencies in the previous reduction.

7 Conclusions

$\mathcal{DLR}_{ifd}$ extends $\mathcal{DLR}$ by fully capturing identification constraints on concepts and functional dependencies on relations. We have shown that reasoning in the presence of such constraints remains EXPTIME decidable. We have also shown that adding to $\mathcal{DLR}$ just unary functional dependencies on non-binary relations, usually considered an indication of bad design in data modeling, leads to undecidability.

The approach presented in this paper can be extended in several ways. For example, our technique can be directly applied to reasoning in $\mathcal{DLR}_{reg}$ extended with identification and functional dependency constraints. Moreover, we are working on the following extensions: (i) using chaining in specifying identification constraints, in the spirit of [Borgida and Weddell, 1997]; (ii) introducing a notion of functional dependency between properties of concepts; (iii) query containment and query answering using views in the presence of identification constraints and functional dependencies.

References

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[Baader and Sattler, 1999] F. Baader and U. Sattler. Expressive number restrictions in description logics. *J. of Log. and Comp.*, 9(3):319–350, 1999.

[Berger, 1966] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[Borgida and Weddell, 1997] A. Borgida and G. E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proc. of DOOD'97*, pages 85–102, 1997.

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

[Calvanese *et al.*, 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of DOOD'95*, volume 1013 of *LNCS*, pages 229–246. Springer-Verlag, 1995.

[Calvanese *et al.*, 1998a] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.

[Calvanese *et al.*, 1998b] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*, pages 2–13, 1998.

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

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer-Verlag, 1999.

[Khizder *et al.*, 2001] V. L. Khizder, D. Toman, and G. E. Weddell. On decidability and complexity of description logics with uniqueness constraints. In *Proc. of ICDT 2001*, 2001.

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

[Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.

[van Emde Boas, 1997] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture notes in pure and applied mathematics*, pages 331–363. Marcel Dekker Inc., 1997.