Answering Queries Using Views over Description Logics Knowledge Bases

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

Answering queries using views amounts to computing the answer to a query having information only on the extension of a set of precomputed queries (views). This problem is relevant in several fields, such as information integration, query optimization, and data warehousing, and has been studied recently in different settings. In this paper we address answering queries using views in a setting where intensional knowledge about the domain is represented using a very expressive Description Logic equipped with n-ary relations, and queries are nonrecursive datalog queries whose predicates are the concepts and relations that appear in the Description Logic knowledge base. We study the problem under different assumptions, namely, closed and open domain, and sound, complete, and exact information on view extensions. We show that under the closed domain assumption, in which the set of all objects in the knowledge base coincides with the set of objects stored in the views, answering queries using views is already intractable. We show also that under the open domain assumption the problem is decidable in double exponential time.

Introduction

Answering queries using views amounts to computing the answer to a query having information only on the extension of a set of views (Abiteboul & Duschka 1998; Grahne & Mendelzon 1999). This problem is relevant in several fields, such as information integration (Ullman 1997), data warehousing (Widom 1995), query optimization (Chaudhuri *et al.* 1995), etc. Data integration is perhaps the obvious setting where query answering using views is important: a typical integration process results in a set of precomputed views, and the query evaluation mechanism can only rely on such views in order to derive correct answers to queries. Previous work on answering queries using views does not allow for the possibility to take into account intensional knowledge about the domain, or considers only restricted ways to express such knowledge (Rousset 1999).

In this paper we address the problem of answering queries using views for non-recursive datalog queries embedded in a knowledge base expressed in an expressive Description Logic (DL). Our goal is to study the computational complexity of the problem, under different assumptions, namely, closed and open domain, and sound, complete, and exact information on view extensions. Such assumptions have been used in data integration with the following meaning. The *closed domain assumption* states that the set of all objects in the domain coincides with the set of objects explicitly mentioned in the knowledge base (in databases, typically the set of objects in the view extensions). On the contrary, the *open domain assumption* leaves the possibility open that other objects besides those mentioned in the knowledge base exist in the domain. With regard to the assumptions on views, a *sound view* corresponds to an information source which is known to produce only, but not necessarily all, the answers to the associated query. A *complete view* models a source which is known to produce all answers to the associated query, and maybe more. Finally, an *exact view* is known to produce exactly the answers to the associated query.

In this paper we consider query answering using views in the following framework:

- Domain knowledge is expressed in terms of a knowledge base constituted by general inclusion assertions and membership assertions, formulated in an expressive DL which includes n-ary relations and qualified number restrictions (Calvanese, De Giacomo, & Lenzerini 1998).
- Queries and views are expressed as non-recursive datalog programs, whose predicates in the body are concepts or relations that appear in the knowledge base.
- It can be specified whether the domain is open or closed, and for each view, whether the provided extension is sound, complete, or exact.

Observe that the DL considered is capable to capture formally Conceptual Data Models typically used in databases (Hull & King 1987; ElMasri & Navathe 1988), such as the Entity-Relationship Model (Chen 1976). Hence, in our setting, query answering using views is done under the constraints imposed by the conceptual data model.

We present the following results for the described setting: Query answering using views is decidable in all cases. Moreover, under the closed domain assumption, the problem is in Δ_3^P and is coNP-complete in data complexity, whereas under the open domain assumption it is in 2EXPTIME.

Answering queries using views is tightly related to query rewriting (Levy *et al.* 1995; Duschka & Genesereth 1997; Ullman 1997). In particular, (Beeri, Levy, & Rousset 1997) studies rewriting of conjunctive queries using conjunctive

Copyright © 2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

views whose atoms are DL concepts or roles. In general, a *rewriting* of a query with respect to a set of views is a function that, given the extensions of the views, returns a set of tuples that is contained in the answer set of the query with respect to the views. Usually, one fixes a priori the language in which to express rewritings (e.g., unions of conjunctive queries), and then looks for the best possible rewriting expressible in such a language. On the other hand, we may call *perfect* a rewriting that returns exactly the answer set of the query with respect to the views, independently of the language in which it is expressed. Hence, if an algorithm for answering queries using views exists, it can be viewed as a perfect rewriting. The results in this paper show the existence of perfect, and hence maximal, rewritings in a setting generalizing that in (Beeri, Levy, & Rousset 1997), which was a question left open in that paper.

Description Logic $\mathcal{DLR}$

To specify knowledge bases and queries we use the Description Logic $\mathcal{DLR}$ (Calvanese, De Giacomo, & Lenzerini 1998). 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, atomic concepts, and *constants*, denoted by P, A 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 according to the following syntax:

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

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 use the usual abbreviations, in particular $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$, $C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$, and $C_1 \equiv C_2$ for $(C_1 \Rightarrow C_2) \sqcap (C_2 \Rightarrow C_1)$.

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 $\mathcal{DLR}$ *knowledge base* (KB) is constituted by a finite set of *assertions*, where each assertion has one of the forms:

$$R_1 \sqsubseteq R_2, \quad C_1 \sqsubseteq C_2, \quad C(a), \quad R(a_1, \ldots, a_n)$$

where R_1 and R_2 are of the same arity, and R has arity n.

The semantics of $\mathcal{DLR}$ is specified as follows. An *interpretation* $\mathcal{I}$ of a KB is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each constant an element of $\Delta^{\mathcal{I}}$ under the unique name assumption, 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. Observe that, the "$\neg$" constructor on relations is used to express difference of relations, and not the complement (Calvanese, De Giacomo, & Lenzerini 1998). We assume that $\Delta^{\mathcal{I}}$ is a subset of a fixed infinitely countable domain Δ. To simplify the notation we do not distinguish between constants and their interpretations.

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. $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$), and satisfies an assertion $C(a)$ (resp., $R(a_1, \ldots, a_n)$) if $a \in C^{\mathcal{I}}$ (resp., $(a_1, \ldots, a_n) \in R^{\mathcal{I}}$). An interpretation that satisfies all assertions in a KB $\mathcal{K}$ is called a *model* of $\mathcal{K}$.

A *query* Q is a non-recursive datalog query of the form

$$Q(\vec{\mathbf{x}}) \leftarrow body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

where each $body_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of *atoms*, and $\vec{\mathbf{x}}$, $\vec{\mathbf{y}}_i$ are all the variables appearing in the conjunct. Each atom has one of the forms $R(\vec{\mathbf{t}})$ or $C(t)$, where $\vec{\mathbf{t}}$ and t are variables in $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}_i$ or objects of the knowledge base, and R, C are relations and concepts, respectively. The number of variables of $\vec{\mathbf{x}}$ is called the *arity* of Q, and is the arity of the relation denoted by the query Q.

We observe that the atoms in the queries are arbitrary $\mathcal{DLR}$ relations and concepts, freely used in the assertions of the KB. This distinguishes our approach with respect to (Donini *et al.* 1998; Levy & Rousset 1996), where no constraints on the relations that appear in the queries can be expressed in the KB.

Given an interpretation $\mathcal{I}$ of a KB, a query Q of arity n is interpreted as the set $Q^{\mathcal{I}}$ of n-tuples $(o_1, \ldots, o_n)$, with each $o_i \in \Delta^{\mathcal{I}}$, such that, when substituting each o_i for x_i, the formula

$$\exists \vec{\mathbf{y}}_1.body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee \exists \vec{\mathbf{y}}_m.body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

evaluates to true in $\mathcal{I}$.

We observe that $\mathcal{DLR}$ is able to capture a great variety of data models with many forms of constraints (Calvanese, Lenzerini, & Nardi 1998; Calvanese, De Giacomo, & Lenzerini 1998). Logical implication (checking whether a given assertion logically follows from a KB) in $\mathcal{DLR}$ is EXPTIME-complete, and query containment (checking whether one query is contained in another one in every model of a KB) is EXPTIME-hard and solvable in 2EXPTIME (Calvanese, De Giacomo, & Lenzerini 1998).

Answering queries using views in $\mathcal{DLR}$

Consider a KB $\mathcal{K}$, and suppose we want to answer a query Q only on the basis of our knowledge about the extension of a set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$. Associated to each view V_i we have:

- A definition $def(V_i)$ in terms of a query $V_i(\vec{\mathbf{x}}) \leftarrow v_i(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ over $\mathcal{K}$.
- A set $ext(V_i)$ of tuples of objects (whose arity is the same as that of V_i) which provides the information about the extension of V_i. We assume without loss of generality that such objects already appear in the KB.
- A specification $as(V_i)$ of which *assumption* to adopt for the view V_i, i.e., how to interpret $ext(V_i)$ with respect to the actual set of tuples that satisfy V_i. We describe below the various possibilities that we consider for $as(V_i)$.

As pointed out in several papers (Abiteboul & Duschka 1998; Grahne & Mendelzon 1999; Levy 1996; Calvanese *et al.* 2000), the above problem comes in different forms,

$$
\begin{array}{rcl}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n \\
P^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} \\
\$i/n : C^{\mathcal{I}} & = & \{(d_1, \ldots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
(\neg R)^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus R^{\mathcal{I}} \\
(R_1 \sqcap R_2)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}
\end{array}
\qquad
\begin{array}{rcl}
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} & = & C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists [\$i] R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists (d_1, \ldots, d_n) \in R^{\mathcal{I}}. d_i = d\} \\
(\leq k\, [\$i] R)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \sharp\{(d_1, \ldots, d_n) \in R_1^{\mathcal{I}} \mid d_i = d\} \leq k\}
\end{array}
$$

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

depending on various assumptions about how accurate is the knowledge on both the objects of the KB, and the pairs satisfying the views. With respect to the knowledge about the objects, we distinguish between:

- *Closed Domain Assumption*. The exact set of objects in the domain of interpretation is known, and coincides with the set of objects that appear in the KB. Formally, an interpretation $\mathcal{I}$ of a KB *satisfies the closed domain assumption*, if $\Delta^{\mathcal{I}}$ coincides with the set of objects in the KB.
- *Open Domain Assumption*. Only a subset of the objects in the domain of interpretation is known. Formally, an interpretation $\mathcal{I}$ of a KB *satisfies the open domain assumption*, if $\Delta^{\mathcal{I}}$ includes the set of objects in the KB. Notice that this is the usual assumption in DLs.

With regard to the knowledge about the views, we consider the following three assumptions:

- *Sound Views*. When a view V_i is *sound*, from the fact that a tuple is in $ext(V_i)$ one can conclude that it satisfies the view, while from the fact that a tuple is not in $ext(V_i)$ one cannot conclude that it does not satisfy the view. Formally, an interpretation $\mathcal{I}$ of a KB is *a model of a sound view* V_i if $ext(V_i) \subseteq def(V_i)^{\mathcal{I}}$.
- *Complete Views*. When a view V_i is *complete*, from the fact that a tuple is in $ext(V_i)$ one cannot conclude that such a tuple satisfies the view. On the other hand, from the fact that a tuple is not in $ext(V_i)$ one can conclude that such a tuple does not satisfy the view. Formally, an interpretation $\mathcal{I}$ of a KB is *a model of a complete view* V_i if $ext(V_i) \supseteq def(V_i)^{\mathcal{I}}$.
- *Exact Views*. When a view V_i is *exact*, the extension of the view is exactly the set of tuples of objects that satisfy the view. Formally, an interpretation $\mathcal{I}$ of a KB is *a model of an exact view* V_i if $ext(V_i) = def(V_i)^{\mathcal{I}}$.

The problem of *answering queries using views under the open (resp., closded) domain assumption in* $\mathcal{DLR}$ is the following: Given

- a KB $\mathcal{K}$,
- a set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$, with, for each V_i, its definition $def(V_i)$, its extension $ext(V_i)$, and the specification of whether it is sound, complete, or exact,
- a query Q of arity n, and a tuple $\vec{\mathbf{d}} = (d_1, \ldots, d_n)$ of objects in the KB,

decide whether $\vec{\mathbf{d}} \in ans(Q, \mathcal{K}, \mathcal{V})$, i.e., decide whether $(d_1, \ldots, d_n) \in Q^{\mathcal{I}}$, for each $\mathcal{I}$ such that: (i) $\mathcal{I}$ satisfies the open (resp., closed) domain assumption; (ii) $\mathcal{I}$ is a model of $\mathcal{K}$; (iii) $\mathcal{I}$ is a model of $V_1, \ldots, V_m$.

Next we study answering queries using views in $\mathcal{DLR}$, both under the closed and under the open domain assumption.

Closed domain assumption

We start our investigation by considering the closed domain assumption.

We first focus on *data complexity*, i.e., we consider as the only parameter in input the number of objects in the KB (while considering the query and view definitions fixed). By a reduction from graph-3-colorability (known to be NP-complete) analogous to the one in (Calvanese *et al.* 2000), we can show that answering queries using views under the closed domain assumption is coNP-hard in data complexity. Moreover, considering that under the closed domain assumption the number of possible interpretations of the KB is finite, to solve answering queries using views, we can guess an interpretation, check if it is a model of the KB and the views, and evaluate the query. This yields an algorithm, NP in data complexity, that checks whether a tuple is not in the answer to the query.

Theorem 1 *Answering queries using views under the closed domain assumption in* $\mathcal{DLR}$ *is coNP-complete in data complexity.*

Next we consider queries and views that are *simple*, i.e., are an atom of the form $R(\vec{\mathbf{x}})$ or $C(x)$, where R (resp., C) is a (possibly complex) $\mathcal{DLR}$ relation (resp., $\mathcal{DLR}$ concept). Under the assumption that the maximal arity of relations is fixed, and exploiting the results in (Schild 1995), it is possible to show that evaluating a $\mathcal{DLR}$ relation (or concept) over an interpretation is polynomial both in the size of the interpretation and the size of the relation (or concept) itself . Hence, checking whether an interpretation is a model of the KB and a model of the simple views, and checking whether a tuple of objects is in the query, can be done in polynomial time. We obtain the following result.

Theorem 2 *Answering queries using views under the closed domain assumption in* $\mathcal{DLR}$ *is coNP-complete, in the case where the query and all views are simple, and under the assumption that the maximal arity of relations is fixed.*

Finally, in the general case, where queries and views are non-recursive datalog queries, once the relations and concepts appearing as atoms in queries and views are evaluated, checking whether an interpretation is a model of all views,

and whether a tuple is in the answer to the query, requires $O(m)$ (where m is the number of views) calls to an NP or coNP oracle (Cosmadakis 1983). Hence, considering the need to first guess the interpretation, we obtain the following result.

Theorem 3 *Answering queries using views under the closed domain assumption in $\mathcal{DLR}$ is in Δ_3^P, under the assumption that the maximal arity of relations is fixed.*

Open domain assumption

Let us now consider the case of the open domain assumption. In this case we reduce the problem of checking whether a tuple $\vec{\mathbf{d}}$ of objects is in $ans(Q, \mathcal{K}, \mathcal{V})$ to the problem of checking the unsatisfiability of a concept in the DL $\mathcal{CIQ}$ (De Giacomo & Lenzerini 1996). $\mathcal{CIQ}$ can be seen as the DL obtained from $\mathcal{DLR}$ by restricting relations to be binary (*roles* and *inverse roles*) and allowing for complex roles corresponding to regular expressions. The reduction is done in three steps.

Encoding of view extensions by means of special assertions.

For each *view* $V \in \mathcal{V}$, with $def(V) = V(\vec{\mathbf{x}}) \leftarrow v(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ and $ext(V) = \{\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_k\}$, we introduce special assertions as follows.

- If V is *sound*, then for each tuple $\vec{\mathbf{a}}_i$, $1 \leq i \leq k$, we include the existentially quantified assertion:

$$\exists \vec{\mathbf{y}}.v(\vec{\mathbf{a}}_i, \vec{\mathbf{y}})$$

- If V is *complete*, then we include the universally quantified assertion:

$$\forall \vec{\mathbf{x}}.\forall \vec{\mathbf{y}}.((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k) \rightarrow \neg v(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

- If V is *exact*, then, according to the definition, we treat it as a view that is both sound and complete.

Finally, since we are checking whether $\vec{\mathbf{d}}$ is an answer of Q, we are interested in the negation of the query Q instantiated on $\vec{\mathbf{d}}$. Hence we include the universally quantified assertion

$$\forall \vec{\mathbf{y}}.\neg q(\vec{\mathbf{d}}, \vec{\mathbf{y}})$$

where $q(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ is the right hand part of Q.

The newly introduced assertions are not yet expressed in a DL. The next step is to translate them in a DL, namely $\mathcal{CIQ}$ extended with object-names.

Translation into a $\mathcal{CIQ}$ concept.

We translate $\mathcal{K}$ and each of the assertions introduced in the previous step into a single concept in $\mathcal{CIQ}$ plus *object-names*. Object-names are concepts that are satisfied by a single object in each model. Observe that we do not require object-names to be disjoint by default (i.e, we do not make the unique name assumption on them), but disjointness can be explicitly enforced when needed. In order to obtain the translation we proceed as follows.

- We eliminate n-ary relations by means of *reification*, i.e., we represent each n-ary relation by a concept with n functional roles $f_1, \ldots, f_n$, one for each component of the relation (De Giacomo & Lenzerini 1995).
- We reformulate inclusion and membership assertions in $\mathcal{K}$ as concepts, by exploiting reflexive-transitive closure (Schild 1991), and by reexpressing objects as object-names (De Giacomo & Lenzerini 1996). Observe that we need to explicitly enforce disjointness between the object names corresponding to different objects.
- We translate each existentially quantified assertion

$$\exists \vec{\mathbf{y}}.v(\vec{\mathbf{a}}, \vec{\mathbf{y}})$$

by treating every existentially quantified variable as a new object-name (skolem constant). Specifically:

  - An atom $C(t)$, where C is a concept and t is a term (either an object or a skolem constant), is translated to

$$\forall U.(N_t \Rightarrow \sigma(C))$$

where $\sigma(C)$ is the reified counterpart of C, N_t is an object-name corresponding to t, and U is the reflexive-transitive closure of all roles and inverse roles introduced in the reification.

  - An atom $R(\vec{\mathbf{t}})$, where R is a relation and $\vec{\mathbf{t}} = (t_1, \ldots, t_n)$ is a tuple of terms, is translated to the conjunction of the following concepts:

$$\forall U.(N_{\vec{\mathbf{t}}} \Rightarrow \sigma(R))$$

where $\sigma(R)$ is the reified counterpart of R and $N_{\vec{\mathbf{t}}}$ is an object-name corresponding to $\vec{\mathbf{t}}$,

$$\forall U.(N_{\vec{\mathbf{t}}} \equiv \exists f_1.N_{t_1} \sqcap \cdots \sqcap \exists f_n.N_{t_n})$$

and for each i, $1 \leq i \leq n$, a concept

$$\forall U.(N_{t_i} \Rightarrow (\exists f_i^-.N_{\vec{\mathbf{t}}} \sqcap (\leq 1\, f_i^-.N_{\vec{\mathbf{t}}}))$$

Then, the translations of the atoms are combined as in $v(\vec{\mathbf{a}}, \vec{\mathbf{y}})$.

- It remains to translate universally quantified assertions corresponding to the complete views and the query. We focus on complete views, i.e., on assertions of the form

$$\forall \vec{\mathbf{x}}.\forall \vec{\mathbf{y}}.((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k) \rightarrow \neg v(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

since the assertion corresponding to the query has the same form, except that the antecedent is empty.

In fact, it is easy to see that it is sufficient to deal with assertions of the form

$$\forall \vec{\mathbf{x}}.\forall \vec{\mathbf{y}}.((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k) \rightarrow \neg body(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

Following (Calvanese, De Giacomo, & Lenzerini 1998) we construct for $body(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ a special graph, called *tuple-graph*, which reflects the dependencies between variables. Specifically, the tuple-graph is used to detect cyclic dependencies. In general, the tuple-graph is composed of $\ell \geq 1$ connected components. For the i-th connected component we build a $\mathcal{CIQ}$ concept $\delta_i(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ as in (Calvanese, De Giacomo, & Lenzerini 1998). Such a concept

contains newly introduced concepts A_x and A_y, one for each x in $\vec{\mathbf{x}}$ and y in $\vec{\mathbf{y}}$. We have to treat variables in $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ that occur in a cycle in the tuple-graph differently from those outside of cycles. Let $\vec{\mathbf{x}}_c$ (resp., $\vec{\mathbf{y}}_c$) denote the variables in $\vec{\mathbf{x}}$ (resp., $\vec{\mathbf{y}}$) that occur in a cycle, and $\vec{\mathbf{x}}_l$ (resp., $\vec{\mathbf{y}}_l$) those that do not occur in cycles. Consider the concept

$$C[\vec{\mathbf{x}}_c/\vec{\mathbf{s}}, \vec{\mathbf{y}}_c/\vec{\mathbf{t}}]$$

obtained from

$$(\forall U.\neg\delta_1(\vec{\mathbf{x}}, \vec{\mathbf{y}})) \sqcup \cdots \sqcup (\forall U.\neg\delta_\ell(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

as follows:

- for each variable x_i in $\vec{\mathbf{x}}_c$ (resp., y_i in $\vec{\mathbf{y}}_c$), the concept A_{x_i} (resp., A_{y_i}) is replaced by N_{s_i} (resp., N_{t_i});
- for each variable y in $\vec{\mathbf{y}}_l$, the concept A_y is replaced by $\top$.

The concept corresponding to the universally quantified assertion is the conjunction of:

- $\forall U.C_{\vec{\mathbf{x}}_l}$, where $C_{\vec{\mathbf{x}}_l}$ is obtained from $\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k$ by replacing each $x \neq a$ with $X \equiv \neg N_a$. Observe that $(x_1, \ldots, x_n) \neq (a_1, \ldots, a_n)$ is an abbreviation for $x_1 \neq a_1 \vee \cdots \vee x_n \neq a_n$.
- One concept $C[\vec{\mathbf{x}}_c/\vec{\mathbf{s}}, \vec{\mathbf{y}}_c/\vec{\mathbf{t}}]$ for each possible instantiation of $\vec{\mathbf{s}}$ and $\vec{\mathbf{t}}$ with the object-names corresponding to the objects in $\mathcal{K}$, with the proviso that $\vec{\mathbf{s}}$ cannot coincide with any of the $\vec{\mathbf{a}}_i$, for $1 \leq i \leq k$ (notice that the proviso applies only in the case where all variables in $\vec{\mathbf{x}}$ occur in a cycle in the tuple-graph).

The critical point in the above construction is how to express a universally quantified assertion

$$\forall\vec{\mathbf{x}}.\forall\vec{\mathbf{y}}.((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k) \rightarrow \neg body(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$$

If there are no cycles in the corresponding tuple-graph, then we can directly translate the assertion into a $\mathcal{CIQ}$ concept. As shown in the construction above, dealing with a nonempty antecedent requires some special care to correctly encode the exceptions to the universal rule. Instead, if there is a cycle, due to the fundamental inability of $\mathcal{CIQ}$ to express that two role sequences meet in the same object, no $\mathcal{CIQ}$ concept can directly express the universal assertion. The same inability, however, is shared by $\mathcal{DLR}$. Hence we can assume that the only cycles present in a model are those formed by the objects in the KB. And these are taken care of by the explicit instantiation.

Encoding of object-names.

As the last step to obtain a $\mathcal{CIQ}$ concept, we need to encode object-names in $\mathcal{CIQ}$. To do so we can exploit the construction used in (De Giacomo & Lenzerini 1996) to encode $\mathcal{CIQ}$-ABoxes as concepts. Such a construction applies to the current case without any need of major adaptation. To this point it is crucial to observe that the translation above uses object-names in order to form a sort of disjunction of ABoxes (cfr. (Horrocks *et al.* 1999)).

Theorem 4 *Let C_{qa} be the $\mathcal{CIQ}$ concept obtained by the construction above. Then $\vec{\mathbf{d}} \in ans(Q, \mathcal{K}, \mathcal{V})$ if and only if C_{qa} is unsatisfiable.*

The size of C_{qa} is polynomial in the size of the query, of the view definitions, and of the inclusion assertions in $\mathcal{K}$, and is at most exponential in the number of objects of $\mathcal{K}$. The exponential blow-up is due to the number of instantiations of $C[\vec{\mathbf{x}}_c/\vec{\mathbf{s}}, \vec{\mathbf{y}}_c/\vec{\mathbf{t}}]$ with objects of $\mathcal{K}$ that are needed to capture universally quantified assertions. Hence, considering EXPTIME-completeness of satisfiability in $\mathcal{DLR}$ and in $\mathcal{CIQ}$, we get the following result.

Theorem 5 *Answering queries using views under the open domain assumption in $\mathcal{DLR}$ is EXPTIME-hard and can be done in 2EXPTIME.*

Interestingly, when the query, all complete views, and all exact views are simple, the construction above does not require the instantiation that gives rise to the exponential blow-up.

Theorem 6 *Answering queries using views under the open domain assumption in $\mathcal{DLR}$ is EXPTIME-complete, in the case where the query, all complete views, and all exact views are simple.*

Discussion

We stress that answering queries using views under the open domain assumption is essentially an extended form of a familiar reasoning service for DLs, namely *instance checking*, where from a partial knowledge about the extensions of concepts and relations, i.e., the ABox, one wants to establish if a given individual (tuple of individuals) is in the extension of a concept (relation). The first additional aspect introduced by answering queries using views is due to the fact that the query is of a more general form than a single concept or relation. In particular, it contains existentially quantified variables, which introduce universal quantification when the problem is reduced to satisfiability. The second additional aspect is the presence of the views, which introduce additional incomplete information.

Dealing only with the first aspect gives rise to the problem of *query answering (over a KB)*, i.e., given a KB (constituted by a TBox and an ABox), a (non-recursive datalog) query, and a tuple of objects, check whether the tuple satisfies the query in every model of the KB. If we apply the construction presented for the open domain assumption to the case where no views are present, we get a solution to query answering. The resulting algorithm has the same computational complexity as the one for answering queries using views. This is due to the fact that the essential difficulty of dealing with universal quantification is already present in this case.

Finally, we observe that, to compute the whole set $ans(Q, \mathcal{K}\,\mathcal{V})$, we need to run the algorithm presented above once for each possible tuple (of the arity of Q) of objects in $\mathcal{K}$. Since we are dealing with incomplete information in a rich language, we should not expect to do much better than considering each tuple of objects separately. Indeed, in such a setting reasoning on objects, such as query answering, requires sophisticated forms of logical inference. In particular, verifying whether a certain tuple belongs to a query gives rise to a line of reasoning which may depend on the tuple under consideration, and which may vary substantially

from one tuple to another. For simple languages we may indeed avoid considering tuples individually, as shown in (Rousset 1999) for query answering in KBs expressed using the DL $\mathcal{ALN}$ without cyclic TBox assertions. Observe, however, that for such a DL, reasoning on objects is polynomial in both data and expression complexity (Lenzerini & Schaerf 1991; Schaerf 1994), and does not require sophisticated forms of inference.

Conclusions

We have studied query answering using views for non-recursive datalog queries embedded in a $\mathcal{DLR}$ knowledge base. We have considered different assumptions on the view extensions (sound, complete, and exact) and on our knowledge of the domain (closed and open domain assumptions). We have shown decidability and established upper and lower bounds for the computational complexity of the problem under the different assumptions. It remains open to close the gap between the known upper and lower bounds.

We have seen in the introduction that an algorithm for answering queries using views is in fact a perfect rewriting. However, it remains open to find perfect rewritings expressed in a more declarative query language. Moreover it is of interest to find maximal rewritings belonging to well behaved query languages, in particular, languages with polynomial data complexity. Observe that from the coNP-hardness of data complexity already under the closed domain assumption, rewritings expressed in such a language cannot be perfect.

References

Abiteboul, S., and Duschka, O. 1998. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, 254–265.

Beeri, C.; Levy, A. Y.; and Rousset, M.-C. 1997. Rewriting queries using views in description logics. In *Proc. of PODS'97*, 99–108.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. Answering regular path queries using views. In *Proc. of ICDE 2000*, 389–398.

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

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

Chaudhuri, S.; Krishnamurthy, S.; Potarnianos, S.; and Shim, K. 1995. Optimizing queries with materialized views. In *Proc. of ICDE'95*.

Chen, P. P. 1976. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems* 1(1):9–36.

Cosmadakis, S. S. 1983. The complexity of evaluating relational queries. In *Proc. of PODS'83*, 149–155.

De Giacomo, G., and Lenzerini, M. 1995. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of IJCAI'95*, 801–807.

De Giacomo, G., and Lenzerini, M. 1996. TBox and ABox reasoning in expressive description logics. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Proc. of KR'96*, 316–327. Morgan Kaufmann, Los Altos.

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Schaerf, A. 1998. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. of Intelligent Information Systems* 10(3):227–252.

Duschka, O. M., and Genesereth, M. R. 1997. Answering recursive queries using views. In *Proc. of PODS'97*, 109–116.

ElMasri, R. A., and Navathe, S. B. 1988. *Fundamentals of Database Systems*. Benjamin and Cummings Publ. Co., Menlo Park, California.

Grahne, G., and Mendelzon, A. O. 1999. Tableau techniques for querying information sources through global schemas. In *Proc. of ICDT'99*, volume 1540 of *Lecture Notes in Computer Science*, 332–347. Springer-Verlag.

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

Hull, R. B., and King, R. 1987. Semantic database modelling: Survey, applications and research issues. *ACM Computing Surveys* 19(3):201–260.

Lenzerini, M., and Schaerf, A. 1991. Concept languages as query languages. In *Proc. of AAAI'91*, 471–476.

Levy, A. Y., and Rousset, M.-C. 1996. CARIN: A representation language combining Horn rules and description logics. In *Proc. of ECAI'96*, 323–327.

Levy, A. Y.; Mendelzon, A. O.; Sagiv, Y.; and Srivastava, D. 1995. Answering queries using views. In *Proc. of PODS'95*, 95–104.

Levy, A. Y. 1996. Obtaining complete answers from incomplete databases. In *Proc. of VLDB'96*, 402–412.

Rousset, M.-C. 1999. Backward reasoning in ABoxes for query answering. In *Proc. of DL'99*, 18–22.

Schaerf, A. 1994. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. Ph.D. Dissertation, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza".

Schild, K. 1991. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI'91*, 466–471.

Schild, K. 1995. *Querying Knowledge and Data Bases by a Universal Description Logic with Recursion*. Ph.D. Dissertation, Universität des Saarlandes, Germany.

Ullman, J. D. 1997. Information integration using logical views. In *Proc. of ICDT'97*, volume 1186 of *Lecture Notes in Computer Science*, 19–40. Springer-Verlag.

Widom, J. 1995. Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering* 18(2).