Answering Queries Using Views 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
lastname@dis.uniroma1.it
http://www.dis.uniroma1.it/∼*lastname*

Abstract

Answering queries using views amounts to computing the answer to a query having information only on the extension of a set of views. This problem is relevant in several fields, such as information integration, data warehousing, query optimization, etc. In this paper we address the problem of query answering using views for nonrecursive datalog queries embedded in a Description Logics (equipped with n-ary relations) knowledge base. We present the following results. Query answering using views is decidable in all cases. Specifically, if the set of all objects in the knowledge base coincides with the set of objects stored in the views (closed domain assumption), the problem is coNP complete, whereas if the knowledge base may contain additional objects (open domain assumption) it is solvable in double exponential time.

1 Introduction

Answering queries using views amounts to computing the answer to a query having information only on the extension of a set of views. 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.

In this paper we address the problem of query answering using views for nonrecursive datalog queries embedded in a Description Logics (equipped with n-ary relations) knowledge base. 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 knowledge base coincides with the set of objects stored in the views. On the contrary, the open domain assumption leaves the possibility open that other objects besides those stored in the views exist in the knowledge base. 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 a framework where we have a knowledge base formulated in an expressive Description Logic [Calvanese *et al.*, 1998a], and a set of views defined as non-recursive datalog programs, and we want to answer a query, again expressed as a non-recursive datalog program. Moreover, the framework allows the specification of which assumption to adopt for the domain, and of which one to adopt for each of the available views.

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 coNP complete, whereas under the open domain assumption it is solvable in double exponential time.

Our investigation is similar in spirit to the one presented in [Abiteboul and Duschka, 1998; Calvanese *et al.*, 1999a], where the decidability and the complexity of the problem is studied when the views and the queries are expressed in terms of various languages (conjunctive queries, datalog, first-order queries, regular path queries, etc.). It is worth noticing that ABox reasoning in DLs [Donini *et al.*, 1994] can be considered as a special form of query answering using views, in particular for the case where all views are assumed to be sound.

2 The Description Logic $\mathcal{DLR}$

We use the DL $\mathcal{DLR}$ [Calvanese *et al.*, 1998a] to specify knowledge bases and queries. 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:

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

where i and j denote components of relations, i.e., integers 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. Observe that, the "$\neg$" constructor on relations is used to express difference of relations, and not the complement [Calvanese *et al.*, 1998a].

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 $(\Delta^\mathcal{I})^n$. 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 $R_1^\mathcal{I}$ is included in $R_2^\mathcal{I}$ (resp. $C_1^\mathcal{I}$ is included in $C_2^\mathcal{I}$), 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, written in the form:

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

where each $body_i(\vec{\mathbf{x}},\vec{\mathbf{y}}_i,\vec{\mathbf{c}}_i)$ is a conjunction of *atoms*, and $\vec{\mathbf{x}},\vec{\mathbf{y}}_i$ (resp. $\vec{\mathbf{c}}_i$) are all the variables (resp. constants) appearing in the conjunct. Each atom has one of the forms $R(\vec{\mathbf{t}})$, or $C(t)$ where (i) $\vec{\mathbf{t}}$, t, and t' are constants or variables in $\vec{\mathbf{x}},\vec{\mathbf{y}}_i,\vec{\mathbf{c}}_i$, and ($ii$) 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 and 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

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

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 *et al.*, 1998b; 1998a]. Also, we note that logical implication (checking whether a given assertion logically follows from a KB) in $\mathcal{DLR}$ is EXPTIME-complete, and both query containment (checking whether one query is contained in another one in every model of a KB) and query answering (checking whether a tuple of constants satisfies the query in every model of a KB) are EXPTIME-hard and solvable in 2EXPTIME [Calvanese *et al.*, 1998a].

3 Answering queries using views

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

- a definition $def(V_i)$ in terms of a query over $\mathcal{K}$,
- a set $ext(V_i)$ of tuples of constants (whose arity is the same as that of V_i) which provides the information about the extension of V_i,
- 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 set of tuples that satisfy the view V_i in $\mathcal{K}$. We describe below the various possibilities that we consider for $as(V_i)$.

As pointed out in several papers [Abiteboul and Duschka, 1998; Grahne and Mendelzon, 1999; Levy, 1996], the above problem comes in different forms, 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 views. We say that an interpretation $\mathcal{I}$ of a KB is *a model of* $ext(V_1),\ldots,ext(V_n)$ *under CDA* if $\Delta^\mathcal{I}$ coincides with the set of objects appearing in $ext(V_1) \cup \cdots \cup ext(V_n)$.
- *Open Domain Assumption.* Only a subset of the objects in the domain of interpretation is known. We say that an interpretation $\mathcal{I}$ of a KB is *a model of* $ext(V_1),\ldots,ext(V_n)$ *under ODA* if $\Delta^\mathcal{I}$ is a superset of the set of constants appearing in $ext(V_1) \cup \cdots \cup ext(V_n)$.

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

- *Sound View Assumption.* When a view V_i is *sound* (satisfies the SVA), written $as(V_i) = \text{SVA}$, 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. More formally, if $as(V_i) = \text{SVA}$, then an interpretation $\mathcal{I}$ of a KB is *a model of* V_i if $ext(V_i) \subseteq def(V_i)^{\mathcal{I}}$.
- *Complete View Assumption.* When a view V_i is *complete* (satisfies the CVA), written $as(V_i) = \text{CVA}$, 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. More formally, if $as(V_i) = \text{CVA}$, then an interpretation $\mathcal{I}$ of a KB is *a model of* V_i if $ext(V_i) \supseteq def(V_i)^{\mathcal{I}}$.
- *Exact View Assumption.* For each view V_i is exact (satisfies the EVA), written $as(V_i) = \text{EVA}$, the extension of the view is exactly the set of tuples of objects that satisfy the view. More formally, if $as(V_i) = \text{EVA}$, then an interpretation $\mathcal{I}$ of a KB is *a model of* V_i if $ext(V_i) = def(V_i)^{\mathcal{I}}$.

The problem of *answering queries using views under the domain assumption α in $\mathcal{DLR}$* is the following: Given

- a KB $\mathcal{K}$,
- a set of views $\mathcal{V} = \{V_1, \ldots, V_n\}$ with $def(V_i)$, $ext(V_i)$, and $as(V_i)$, for each V_i,
- a query Q of arity n, and a tuple $\vec{\mathbf{d}} = (d_1, \ldots, d_n)$ of constants,

decide whether $\vec{\mathbf{d}} \in ans(Q, \mathcal{K}, \mathcal{V})$ under α, i.e., decide whether $(a_1, \ldots, a_n) \in Q^{\mathcal{I}}$, for each $\mathcal{I}$ such that: (i) $\mathcal{I}$ is a model of $\mathcal{K}$; (ii) $\mathcal{I}$ is a model of $ext(V_1), \ldots, ext(V_n)$ under α; (iii) $\mathcal{I}$ is a model of every V_i.

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

Our goal is to characterize the complexity of query answering using views in $\mathcal{DLR}$.

4.1 Under closed domain assumption

We start our investigation by considering the closed domain assumption. By exploiting one of the results in [Calvanese *et al.*, 1999a], it is easy to see that the problem is coNP-hard. Moreover, the number of possible interpretations of the KB is finite, and therefore, we can guess one of them, check if it is a model which is a model of the views, and evaluate the query. This yields an NP algorithm that checks whether the answer to the query is no.

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

4.2 Under 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 constants is in $ans(Q, \mathcal{K}, \mathcal{V})$ to the problem of checking the satisfiability of a concept in the DL $\mathcal{CIQ}$ [De Giacomo and Lenzerini, 1996]. The reduction is done in three steps.

- First we add to the KB $\mathcal{K}$ special assertions as follows.
  - For each *sound view* V, with $def(V) = body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1, \vec{\mathbf{c}}_1) \vee \cdots \vee body_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m, \vec{\mathbf{c}}_m)$, for each tuple $\vec{\mathbf{a}}$ in $ext(V)$, we include an existentially quantified formula:
  $$\exists \vec{\mathbf{y}}_1 \cdots \exists \vec{\mathbf{y}}_m . body_1(\vec{\mathbf{a}}, \vec{\mathbf{y}}_1, \vec{\mathbf{c}}_1) \vee \cdots \vee body_m(\vec{\mathbf{a}}, \vec{\mathbf{y}}_m, \vec{\mathbf{c}}_m)$$
  - For each *complete view* V, we include a universally quantified formula:
  $$\forall \vec{\mathbf{x}} . \forall \vec{\mathbf{y}} . ((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \vee \cdots \vee \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_k) \supset \neg q(\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{c}}))$$
  where $\{\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_k\} = ext(V)$ and $q(\vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{c}})$ is the right hand part of $def(V)$.
  - According to the definition, we treat each *exact view* as a view that is both sound and complete.
  - Finally, since we are checking whether $\vec{\mathbf{d}}$ is an answer of Q, we consider the negation of the query Q, and we include a universally quantified formula:
  $$\forall \vec{\mathbf{y}} . \neg q(\vec{\mathbf{d}}, \vec{\mathbf{y}}, \vec{\mathbf{c}})$$
  where $q(\vec{\mathbf{d}}, \vec{\mathbf{y}}, \vec{\mathbf{c}})$ is obtained by instantiating $\vec{\mathbf{x}}$ to $\vec{\mathbf{d}}$ in the right hand part of Q.
- Then we translate $\mathcal{K}$ and each of the formulas introduced in the previous step into a single concept in $\mathcal{CIQ}$ plus *object names* (which are concepts that are satisfied by a single object in each model). In particular following [Calvanese *et al.*, 1998a]:
  - We eliminate n-ary relations by means of reification.
  - We internalize the assertions in $\mathcal{K}$.
  - We translate each existentially quantified formula into a concept, treating every existentially quantified variable as a new object name (skolem constant).
  - We translate each universally quantified formula into a conjunction of concepts, one for each possible instantiation of the universally quantified variables with the object names introduced so far.
- Finally, we eliminate object names along the lines of [Calvanese *et al.*, 1998a], thus obtaining a concept in $\mathcal{CIQ}$.

The reduction can be shown to be correct along the line of [Calvanese *et al.*, 1998a]. From the reduction we get the following result.

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

Interestingly, when all views are sound and the query is *simple*, i.e., is an atom of the form $R(\vec{\mathbf{x}})$ or $C(x)$, we obtain a setting that corresponds closely to the typical TBox and ABox reasoning in DLs. In this case the following result holds.

Theorem 3 *Answering simple queries using sound views under the open domain assumption in $\mathcal{DLR}$ is EXPTIME-complete.*

5 Comparison with query rewriting

The problem of query answering using views has also been dealt with techniques based on rewriting queries using views [Levy *et al.*, 1995; Duschka and Genesereth, 1997; Ullman, 1997; Beeri *et al.*, 1997]: Given a query Q and views $V_1, \ldots, V_n$ with associated definitions $def(V_1), \ldots, def(V_n)$, generate a new query Q' over the alphabet $V_1, \ldots, V_n$ such that for every database (model in our setting), first computing the extension of each V_i on the database, and then evaluating Q' on the basis of such extensions, provides the answer to Q. Although methods for query rewriting can be adapted to the problem of query answering using views [Levy *et al.*, 1995], the two problems are different. Query rewriting has as inputs only the view definitions and the query and uses the view definitions to re-express the query in terms of the views. Then, to compute the answer to the original query, the rewritten query is evaluated on the extensions of the views. On the other hand, query answering takes as inputs the view definitions, the view extensions, the view assumptions, and the query, and computes directly the answer to the query.

Hence, in the general case one cannot exploit query rewriting using views for query answering. In particular, when the rewriting is not exact (i.e., it is not equivalent to the query), it may miss some tuples that are in the answer to a query. Even if there exists an exact rewriting, it may still miss some tuples of the answer to a query in the case where the views are sound (but not exact). Only if the rewriting is exact and the views are exact, one can use such rewriting to solve the query answering problem [Grahne and Mendelzon, 1999; Calvanese *et al.*, 1999b].

Note that query rewriting in our setting remains an open problem.

6 Conclusions

We have studied query answering using views for nonrecursive 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 established decidability and established upper and lower bounds for the computational complexity of the problem.

We conclude by stressing that *query answering using views* 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).

References

[Abiteboul and Duschka, 1998] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[Beeri *et al.*, 1997] Catriel Beeri, Alon Y. Levy, and Marie-Christine Rousset. Rewriting queries using views in description logics. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[Calvanese *et al.*, 1998a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[Calvanese *et al.*, 1998b] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publisher, 1998.

[Calvanese *et al.*, 1999a] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Vardi. Answering regular path queries using views. Technical report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1999.

[Calvanese *et al.*, 1999b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Vardi. Query answering using views for data integration over the web. In *2nd Int. Workshop on the Web and Databases (WebDB'99)*, 1999.

[Chaudhuri *et al.*, 1995] S. Chaudhuri, S. Krishnamurthy, S. Potarnianos, and K. Shim. Optimizing queries with materialized views. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95)*, Taipei, Taiwan, 1995.

[De Giacomo and Lenzerini, 1996] Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.

[Donini *et al.*, 1994] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[Donini *et al.*, 1998] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating datalog and description logics. *J. of Intelligent Information Systems*, 10(3):227–252, 1998.

[Duschka and Genesereth, 1997] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997.

[Grahne and Mendelzon, 1999] Gösta Grahne and Alberto O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer-Verlag, 1999.

[Levy and Rousset, 1996] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI'96)*, pages 323–327, 1996.

[Levy *et al.*, 1995] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[Levy, 1996] Alon Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 402–412, 1996.

[Ullman, 1997] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1997.

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