Switching Terminologies - Creating a New View of An Old World: Preliminary Report

Catriel Beeri*
Hebrew University
Institute of Computer Science,
Jerusalem, Israel
beeri@cs.huji.ac.il

Alon Y. Levy
AI Principles Research Department
AT&T Laboratories
Murray Hill, NJ
levy@research.att.com

Marie-Christine Rousset
L.R.I. U.R.A C.N.R.S
University of Paris-Sud
Orsay, France
mcr@lri.lri.fr

1 Introduction

Several advanced applications of database systems require the modeling, maintenance, and usage of large collections of views. Prime examples include mediator systems that provide access to multiple information sources, data mining and archeology, mobile databases, data warehouses, and decision support systems. Furthermore, some database vendors are considering the maintenance of materialized views also as a means for query optimization. As a result, problems concerning materialized views have recently received a lot of attention in the database community.

A view is essentially a query. If the answers to the query are physically maintained, the view is said to be materialized. Naturally, when there are many views (either materialized or not) there are complex relationships between the answers to different views. For example, one view may be guaranteed to be a superset of another, or two views may be guaranteed to be mutually disjoint. In order to perform tasks involving large collections of views, a system needs the capability to reason about the relationships between views, and about the relationship between a view and a query.

An important problem concerning materialized views is that of rewriting queries using views. Informally, the problem is the following. We are given a query over a set of database relations, but we do not have access to the actual database relations. Instead, we have access to *views* over the database relations, and we must find a way of answering the query using the views. Rewriting queries using views is a crucial problem for information integration [Levy *et al.*, 1996a; Ullman, 1997], where the information sources are modeled as views, and the mediator needs to find a way to answer queries using the information sources. The problem has been considered recently in several works [Yang and Larson, 1987; Chaudhuri *et al.*, 1995; Levy *et al.*, 1995; 1996b; Srivastava *et al.*, 1996].

Several authors have noted that description logics are a very natural formalism for modeling and reasoning about views, because they provide a tool for modeling complex relationships between sets of objects. While there is great promise in using description logics for problems involving views, there are several issues that need to be addressed in order to close the gap between the current capabilities of description logics and the needs of applications using views in relational and object oriented systems. In previous work [Levy and Rousset, 1996a; 1996b] we have developed a query language over description logics, CARIN, which provides constructs that are common to traditional database query languages (e.g., joins, arbitrary conjunctive queries, recursion). In this paper we consider the problem of rewriting queries using views, when the queries are concepts, and the views are *conjunctive queries* over a description logic, i.e., conjunctions that allow both concept and role atoms and existential variables. In particular, we show that when the queries and views denote concepts, it is possible to find a rewriting of the query using the views whenever the description logic is decidable and has the conjunction operator. When the views are conjunctive queries, we show that the rewriting of the query using the views may result in a *recursive* set of rules.

We observe that the general case of query rewriting indicates that it involves the use of inference procedures that do not rely on the *unique names assumption*. Most of the procedures in the literature use this assumption.

Finally, it should be noted that the problem of answering queries using views is also interesting from another perspective. We can think of the views as representing one terminology, while a set of queries represents a second terminology. The problem of rewriting the query using the views provides a *mapping* between the two terminologies. Constructing such mappings is important for applications in which we need to deal with multiple ontologies (and therefore, multiple terminologies).

2 Preliminaries and Examples

In this paper we consider the description logic, denoted $\mathcal{L}_1$, that includes the constructors $\sqcap$, $(\forall R.C)$, $(\leq n\, R)$, $(\geq n\, R)$, and negation on primitive concepts. We assume that our terminology contains only acyclic concept definitions (and not arbitrary inclusions), and they are

*Work partially supported by the Israeli Science Foundation, administered by the Israeli Academy of Science

assumed to be unfolded. Note that $\mathcal{L}_1$ is a subset of the language used in CLASSIC, and therefore subsumption in the language is decidable in polynomial time.[1]

2.1 Queries and Views

We consider queries and views that are datalog programs [Ullman, 1989] over description logic concepts and roles (i.e., CARIN programs [Levy and Rousset, 1996a]). *Datalog* programs are collections of horn-rules of the form:

$Q : q(\bar{X}) : -p(\bar{X}_1)$ & ... & $p(\bar{X}_n)$.

The *head* of the rule is $q(\bar{X})$, and variables in the tuple $\bar{X}$ are called the distinguished variables. We require that the rules be safe, i.e., every variable that appears in the head of the rule should appear also in the body (that is, $\bar{X} \subseteq \bar{X}_1 \cup \ldots \cup \bar{X}_n$). Variables that appear in the body and not in the head are called *existential variables.*

We distinguish between two sets of predicates in a given datalog program: the *extensional predicates* (EDB predicates), that appear only in bodies of rules, and the *intensional predicates* (IDB predicates), which are the predicates appearing in heads of rules, and may also appear in the bodies. The EDB predicates refer to the given relations while the IDB predicates are defined by the program. In our context, we assume that the EDB predicates are concept names, descriptions or roles that appear in a terminology $\mathcal{T}$ in $\mathcal{L}_1$, and the IDB predicates have arbitrary arities. A query that contains a single rule, and all the predicates in the body are EDB's is called a *conjunctive query.*

We assume one of the IDB predicates of the program is the *query predicate.* Given a terminology $\mathcal{T}$, a set of rules $\mathcal{R}$, a set of ground atomic facts $\mathcal{A}$ for the concept and role predicates, and a query predicate p of arity m, the answer to a query $p(\bar{X})$ are the set of m-tuples $\bar{a}$, formed by constants appearing in $\mathcal{A}$, such that $\mathcal{T} \cup \mathcal{R} \cup \mathcal{A} \models p(\bar{a})$. In ordinary datalog programs, (i.e., when the EDBs are ordinary relations), the answer to a query can be computed in a bottom-up fashion in time that is polynomial in the number of ground facts $\mathcal{A}$. A bottom-up evaluation starts with the ground EDB facts and applies the rules to derive facts for the IDB predicates. We continue applying the rules until no new facts are generated. However, when the EDB predicates are defined in a terminology, a simple evaluation of the query from the database may not suffice, because the database may contain *incomplete* information that affects the result.

Example 2.1: Suppose we have the simple terminology

parent := person ⊓ ($\geq$ 1 child).

and suppose our query is

$q(X) : -person(X)$ & $child(X, Y)$.

If our database contains the facts { parent(a), person(b), child(b,c) } then a simple evaluation of the query will yield only the result $\{(b)\}$. However, the answer should be $\{(a), (b)\}$. □

Sound and complete algorithms for evaluating datalog queries over our language are described in [Levy and Rousset, 1996a; 1996b].

The notion of containment plays a key role in obtaining query rewritings:

Definition 2.1: *(Containment) A datalog query $\mathcal{P}_1$ contains a datalog query $\mathcal{P}_2$, written $\mathcal{P}_2 \subseteq \mathcal{P}_1$, if for any set of ground atoms $\mathcal{A}$ for the concepts and roles, the answer of $\mathcal{P}_1$ is a superset of the tuples of the answer of $\mathcal{P}_2$. The two queries are said to be* equivalent *if $\mathcal{P}_1 \subseteq \mathcal{P}_2$ and $\mathcal{P}_2 \subseteq \mathcal{P}_1$.* □

A materialized view is a query whose result has been physically stored. The problem we consider in this paper is that we are given a query over the concepts and roles in a terminology $\mathcal{T}$, but the extensions of the relations in $\mathcal{T}$ are not given. Instead, we are given the definitions of a set of views over $\mathcal{T}$ that are materialized, and our goal is to find a *rewriting* of the query using the materialized views. A rewriting is a datalog query whose EDB predicates are the materialized views. There are two kinds of rewritings that we consider, *equivalent rewritings* and *maximal rewritings.*

Definition 2.2: *(rewriting) Let Q be a query over the terminology $\mathcal{T}$, and let Q' be another query, whose EDB predicates are the predicates $V_1, \ldots, V_m$ and the predicate $\neq$, and whose IDB predicates are disjoint from those of Q.*
(i) Q' is an equivalent rewriting of Q using the views $V_1, \ldots, V_m$ if Q' and Q are equivalent queries.
(ii) Q' is a maximal rewriting of Q using the views $V_1, \ldots, V_m$ if

- *$Q' \subseteq Q$, and*
- *there does not exist a query Q'' satisfying the first condition, such that $Q' \subseteq Q'' \subseteq Q$ and $Q' \not\supseteq Q''$.* □

It should be noted that in general, obtaining a complete answer to a datalog query over the language $\mathcal{L}_1$ is undecidable [Levy and Rousset, 1996b]. However, in this paper we will be concerned with case in which the views are conjunctive queries (i.e., datalog programs that contain only one Horn rule). A sound and complete algorithm for evaluating conjunctive queries follows from [Levy and Rousset, 1996a]. The rewritings of the queries using the views may be recursive; however, in that case, since we assume the views are complete (and can therefore be treated as ordinary EDB predicates), a bottom-up evaluation of the rewriting is sound and complete.

Example 2.2: Consider a terminology that includes the primitive concepts person and smart, and the role child. The concept happy-parent is defined by:

[1]Negation on primitive concepts is not explicitly allowed in CLASSIC, but can be added without affecting the complexity of subsumption.

happy-parent := person $\sqcap$ ($\forall$ child.smart).

Suppose we have the following views:

$v_1(X):-person(X)$ & $(\leq 1\,child)(X)$
$v_2(X):-person(X)$ & $child(X,Y)$ & $smart(Y)$
$v_3(X):-happy-parent(X)$ & $(\geq 2\,\text{child})(X)$
$v_4(X):-(\geq 3\,child)(X)$.

Suppose our query is

$q(X):-happy-parent(X)$.

We have the following rewritings of the query using the views:

$q_1(X):-v_3(X)$
$q_2(X):-v_1(X)$ & $v_2(X)$

The first rewriting gives us the happy parents that also have at least 2 children. The second rewriting is more subtle. The view v_1 provides parents with at most one child, while the view v_2 provides persons with a least one smart child. Therefore, persons found in both of the views have exactly one child who is smart, and are therefore happy parents. Note that neither of the rewritings is equivalent to the query, since an equivalent rewriting does not exist from these views. Consider the query

$p(X):-happy-parent(X)$ & $(\geq 3\,child)(X)$.

For this query it is possible to find an equivalent rewriting using the views:

$p'(X):-v_3(X)$ & $v_4(X)$.

Several papers consider the rewriting problem for the case of conjunctive queries and unions of such queries, but without terminologies. In [Levy *et al.*, 1995] it is shown that for conjunctive queries and views, if there exists a rewriting of the query using the views then there exists one whose length is at most the length of the query. As a result, we get a nondeterministic polynomial time procedure for finding rewritings, if they exist. As shown in [Levy *et al.*, 1995; Rajaraman *et al.*, 1995] the bound on the size of the rewriting changes when we make slight modifications to the query language. Finding a bound on the size of the rewriting, together with the existential entailment algorithm [Levy and Rousset, 1996a] (which is a generalization of containment algorithms for CARIN queries) would also yield a procedure for finding rewritings. However, as the following example shows, even if the queries and views are conjunctive queries, there may not always be a bound on the size of the rewriting, and we may have to resort to recursive rewritings.

Example 2.3: Consider the following two views:

$v_1(X,Y):-child(X,Y)$ & $child(X,Z)$ & $(\leq 1\,child)(Z)$
$v_2(X):-(\geq 3\,child)(X)$.

Suppose our query is to find all individuals who have at least 2 children:

$q(X):-(\geq 2\,child)(X)$.

For any n, the rewriting of the form

$q'(X):-v_1(X,Y_1)$ & $v_1(Y_1,Y_2)$ & $\ldots v_1(Y_n,U)$ & $v_2(U)$

is contained in the query, and each may yield answers that are not obtained by other rewritings. To see why, consider the variable Y_n. The view v_1 entails that it has one successor that has less than one child (the variable Z in the definition of v_1) while the view v_2 says that its child U has at least 3 children. Therefore, Y_n has at least 2 children. The same line of reasoning can be used to see that Y_{n-1} has at least 2 children, and continuing in the same way, we get that X has at least two children.

Therefore, we cannot find a bound on the size of the rewritings of the query. On the other hand, the following is a recursive rewriting that is maximal:

$q(X):-v_2(X)$
$q(X):-v_1(X,Y)$ & $q(Y)$.

This recursive program essentially simulates the line of reasoning we outlined above to any depth. □

In the following sections we consider the rewriting problem in increasing levels of difficulty. We begin with the case in which both queries and views are descriptions in $\mathcal{L}_1$. We then consider the case in which the views are conjunctive queries, but without existential variables. Finally, we consider the case in which the views have existential variables. Proofs are omitted due to lack of space.

3 Concept Queries and Views

We begin with the simple but common case in which both the queries and the views are concepts in $\mathcal{L}_1$. In this case, we can show the following result:

Theorem 3.1: *Let Q be a query of the form $q(X):-c(X)$ and $\mathcal{V}=V_1,\ldots,V_m$ be views of the form $V_i(X):-c^i(X)$ where $c,c^1,\ldots,c^m$ are descriptions in $\mathcal{L}_1$. The maximal rewriting of Q using $\mathcal{V}$ is the union of conjunctive queries of the form:*

$Q'(X):-v^1(X)$ & $\ldots$ & $v^l(X)$

where $l\leq m$, $v^1,\ldots,v^l\in\mathcal{V}$ and $c^1\sqcap\ldots\sqcap c^l\sqsubseteq c$. □

The main observation underlying this theorem is that there is no need to consider rewritings that contain existential variables in the rules. Therefore, the only atoms that need to be considered in the bodies are of the form $c(X)$ where $c\in\mathcal{V}$, and X is the head variable.

Theorem 3.1 shows that the rewriting problem is decidable for queries and views that are concepts. However, it still requires that we consider all possible subsets of views, which will be expensive if we have many views (this is the case when the views represent information sources on the network). The following theorem allows us to considerably prune the search for rewritings, by showing that the size of the rewriting can be bounded by the size of the query and depth of the terminology.

Theorem 3.2: *Let Q be a query of the form $q(X) : -c(X)$ and $\mathcal{V} = V_1, \ldots, V_m$ be views of the form $V_i(X) : -c^i(X)$ where $c, c^1, \ldots, c^m$ are descriptions in $\mathcal{L}_1$. Let d_q be the number of constructors in c, and let d be the maximum number of constructors that appear in the concept descriptions $c^1, \ldots, c^m$. If c does not contain occurrences of the description $(\leq 0\, R)$, then Theorem 3.1 holds even if we restrict l to be d_q. If c has b occurrences of $(\leq 0\, R)$, then we can restrict l to be $d_q + db$.* □

To prove Theorem 3.2 we note that every description C in $\mathcal{L}_1$ can be rewritten equivalently (by pushing the $\sqcap$ constructor inside) as a conjunction C' of descriptions each of the form $\forall \bar{R}.\tau$, where $\bar{R}$ is a role-chain, or of the form τ, where τ is of the form $(\leq\, n\, R)$, $(\geq\, n\, R)$, A, or $\neg A$, and A is a primitive concept. The number of conjuncts in C' is at most the number of constructors in C. In order for a conjunction of view atoms to entail C', it has to entail all the conjuncts of C'. However, we show that if a conjunction of views entails a conjunct of C' in which τ is not $(\leq\, 0\, R_1)$, then it is entailed by a *single* view atom. Finally, we show that a conjunct that contains $(\leq\, 0\, R_1)$ may be entailed by a combination of $d + 1$ view atoms.

Note that these observations may be used to directly construct the maximal rewriting. Although the worst case time complexity of the resulting procedure is the same as the one derived from the space bounds, in cases where many views can be expected not to be relevant to a given query it is much smaller.

4 Conjunctive Queries as Views

The next case we consider is one in which the views in $\mathcal{V}$ can be arbitrary conjunctive queries, but without existential variables (i.e., any variable that appears in the body appears also in the head). Such queries allow conjuncts that are role atoms, and therefore enable to express arbitrary chains of variables (of bounded length). The restriction that all body variables appear in the head has the following advantage: To each conjunction of views, with a set of variables $\bar{X}$, we can add a conjunction of inequalities $x_i \neq x_j$, for all pairs in $\bar{X}$, thus ensuring that in every instantiation of the the rewriting, different variables are mapped to distinct constants. Thus, we can rely on the unique names assumption. If we want also to consider the case where $x_i = x_j$, for some i, j, we simply consider the conjunction in which all occurrences of x_j are replaced by x_i. The following theorem establishes a bound on the size of the rewriting of the query using the views, and hence, together with an algorithm for checking satisfiability of an $\mathcal{ALCNR}$ KBs [Buchheit *et al.*, 1993], yields the decidability of the rewriting problem.

We use the following notation in the theorem. We assume N is the maximal number appearing in the number restrictions in the query or in $\mathcal{V}$, and d denotes the maximal number of constructors in a concept that appears in the body of a view in $\mathcal{V}$. We denote the maximum of N and $(d+1)^2$ by M. Given a concept c we denote by $l(c)$ the maximal depth of nested $\forall R.C$ constructors in c (i.e., the maximal length of a role-chain after the $\sqcap$ constructor has been pushed all the way).

Theorem 4.1: *Let Q be a query of the form $q(X) : -c(x)$ where c is a description in $\mathcal{L}_1$, and let $\mathcal{V}$ be a set of conjunctive views such that in every view, all the body variables appear also in the head. Let S be the number of constructors in c. Let $N' = 2l(c)!SM^2$. The maximal rewriting of Q using $\mathcal{V}$ is the union of all conjunctive queries Q' over $\mathcal{V}$ such that:*

- *Q' is contained in Q, and*
- *the body of Q' contains at most N' atoms.* □

Given a rewriting, we check whether it entails the query by first expanding the definitions of the view atoms to obtain a set of ground facts in the language $\mathcal{L}_1$, and then check whether this set of ground facts entails $c(X)$. The proof of the theorem is based on considering how the atom $c(X)$ can be inferred from such a set of ground facts. We show by induction on the structure of c that at most N' ground atoms are needed in order to infer $c(X)$. Therefore, since each ground atom can originate from a different view atom, the length of the rewriting is bound by N'.

5 Allowing Existential Variables in the Views

When the views contain existential variables, the set of ground facts resulting from expanding the view definitions does not necessarily obey the unique names assumption. This is because the existential variables do not necessarily represent distinct individuals. As a result, we cannot apply exactly the same inference procedure that allowed us to establish the bound of Theorem 4.1. In fact the following theorem shows that when we remove the unique names assumption, inference in $\mathcal{L}_1$ becomes intractable, whereas it is a polynomial time decision problem with the unique name assumption.

Theorem 5.1: *Let Δ be a description logic knowledge base in $\mathcal{L}_1$. Checking whether Δ is satisfiable is NP-complete in the number of ground facts in Δ.* □

Fortunately, the knowledge bases resulting from expanding query rewritings are not arbitrary. In the extended version of the paper we show that in some cases it is possible to find a recursive datalog program that is the maximal rewriting of the query. Below we describe an algorithm that provides the maximal rewriting in the case where:

- if $R(X, Y)$ is an atom in one of the views, then X is a distinguished variable,
- every existential variable appears in at most one atom role in the body of the view.

Given the query Q of the form $q(X) : -c(X)$, we denote by $\mathcal{C}$ the set of concepts that can be formed using the primitive concepts and roles mentioned in c, the integers [0..j], where j is the largest integer mentioned in number restrictions in c or in the views, and have at most the number of constructors in c or in one of the views. We denote by M the product of the number of views and the maximum arity of the views.

The datalog rewriting of the query Q using the views $\mathcal{V}$ includes an IDB predicate for every concept in $\mathcal{C}$. The program includes all the rules of the following form:

$$p(X) : -v^1(\bar{X}_1) \ \& \ \ldots \ \& \ v^M(\bar{X}_M) \ \& \ p_1(Y_1) \ldots \ \& \ p_l(Y_l)$$

where

1. $M \leq Max(N, j+1)$ and $v^1, \ldots, v^M \in \mathcal{V}$,
2. $\{Y_1, \ldots, Y_l\} \subseteq \bar{X}_1 \cup \ldots \bar{X}_M$, and
3. $p_1, \ldots, p_l, p$ are IDB predicates corresponding to the concepts $c_1, \ldots, c_l, c_0$ in $\mathcal{C}$, and
4. $v^1(\bar{X}_1) \ \& \ \ldots \ \& \ v^M(\bar{X}_M) \ \& \ c_1(Y_1) \ldots \ \& \ c_l(Y_l) \ \Rightarrow c_0(X)$ holds.

Note that in the rules there may be several predicates p_i predicates for the same variable Y. The intuition underlying the construction of the rewriting is that the rules of the program simulate a set of inference rules that are sound and complete on the KB resulting from expanding the definitions of the views.

References

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Chaudhuri *et al.*, 1995] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *Proceedings of International Conference on Data Engineering*, 1995.

[Levy and Rousset, 1996a] Alon Y. Levy and Marie-Christine Rousset. CARIN: a representation language integrating rules and description logics. In *Proceedings of the European Conference on Artificial Intelligence, Budapest, Hungary*, 1996.

[Levy and Rousset, 1996b] Alon Y. Levy and Marie-Christine Rousset. The limits on combining recursive horn rules and description logics. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[Levy *et al.*, 1995] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, CA*, 1995.

[Levy *et al.*, 1996a] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, pages 40–47, 1996.

[Levy *et al.*, 1996b] Alon Y. Levy, Anand Rajaraman, and Jeffrey D. Ullman. Answering queries using limited external processors. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Montreal, Canada (to appear)*, 1996.

[Rajaraman *et al.*, 1995] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, CA*, 1995.

[Srivastava *et al.*, 1996] Divesh Srivastava, Shaul Dar, H. V. Jagadish, and Alon Y. Levy. Answering SQL queries using materialized views. In *Proceedings of VLDB*, 1996.

[Ullman, 1989] Jeffrey D. Ullman. *Principles of Database and Knowledge-base Systems, Volumes I, II.* Computer Science Press, Rockville MD, 1989.

[Ullman, 1997] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of the International Conference on Database Theory, to appear*, 1997.

[Yang and Larson, 1987] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proceedings of the 13th International VLDB Conference*, pages 245–254, 1987.