Rewriting Queries Using Views in Description Logics

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

Alon Y. Levy
AI Principles Research Department
AT&T Labs
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

Abstract

The problem of rewriting queries using views is to find a query expression that uses only a set of views $\mathcal{V}$ and is equivalent to (or maximally contained in) a given query Q. Rewriting queries using views is important for query optimization and for applications such as information integration and data warehousing. *Description logics* are a family of logics that were developed for modeling complex hierarchical structures, and can also be viewed as a query language with an interesting tradeoff between complexity and expressive power. We consider the problem of rewriting queries using views expressed in description logics and conjunctive queries over description logics. We show that if the view definitions do not contain existential variables, then it is always possible to find a rewriting that is a union of conjunctive queries, and furthermore, this rewriting produces the maximal set of answers possible from the views. If the views have existential variables, the rewriting may be recursive. We present an algorithm for producing a recursive rewriting, that is guaranteed to be a maximal one when the underlying database forms a tree of constants. We show that in general, it is not always be possible to find a maximal rewriting.

1 Introduction

The problem of rewriting queries using views is the following: we are given a set of view definitions $\mathcal{V}$ and a query Q over the same database relations. The goal is to find a query expression that uses *only* the views in $\mathcal{V}$ and is equivalent to or maximally-contained in Q. Rewriting queries using views is important for several database applications. Originally, the problem has been considered for the purpose of query optimization [YL87, KB94, CKPS95]. More recently, rewriting queries using views has been used for information integration [LRO96, LRU96, Ull97], view adaptation and data warehouses [GMR95], and maintaining physical data independence [TSI94]. The rewriting problem has been considered for some classes of relational queries [YL87, LMSS95, Qia96], queries over multisets [CKPS95], queries with aggregation [SDJL96], object-oriented query languages [FRV96] and recursive queries [DG97].

Description logics are a family of logics that were developed for modeling complex hierarchical structures. A description logic (DL) deals with unary predicates (referred to as *concepts*), representing sets of objects, and binary predicates (called *roles*). Typically, some concept and role names (often called *primitive* concepts and roles) are given, and additional concepts and roles are defined by using *constructors*. A concept describes a class of instances in the domain; the constructors used in the definition of the concept determine sufficient and necessary conditions on instances of the class. DL's vary mainly in the constructors they contain, that allow to define such conditions at various levels of complexity. The semantics of DL's follows that of first order logic with equality. Furthermore, it is known that most description logics can be translated into first-order logic with three variable symbols [Bor96]. Varying the set of constructors allowed in a DL enables to easily tailor a description logic with a desired tradeoff between expressiveness and complexity. The algorithmic problems that have been addressed by the research in DL's include *satisfiability*, that is, for each concept, is there a universe with a non-empty extent for the concept, and *subsumption*, i.e., determining when one concept *always* describes a superset of a second concept. Several systems based on DL's have been built (e.g., CLASSIC [BBM+91], LOOM [Mac88], BACK [Pet91], KRIS [BH91]), and they have been used in several applications such as configuration management [WWB+93] natural language processing [WAF+93]), and information integration [LRO96, AKS96].

There are two perspectives on the use of description logics in database systems. The first is the use of description logics as a data modeling language. In this

*The work of this author was supported by a grant from the Israel Science Foundation.

role, DL's have the advantages of being able to express multiple kinds of relationships between concepts and various forms of *incomplete* information, and having the associated algorithms, e.g., that of subsumption which enables to automatically place a new class in a given hierarchy. While data modeling has been a traditional research area in databases, a substantial part of this research, with the exception of research on database design, has concentrated on methodology rather than on algorithmic issues. In contrast, algorithmic issues have been the focus of the work on description logics. Borgida [Bor95] surveys this use of DL's.

An alternative perspective of a DL is as a *query language*. This perspective is especially useful when DL's are used in applications such as data warehousing and integration of information from distributed and heterogeneous data sources. In these applications, database systems need to maintain and reason about large collections of views (or more generally, data descriptions), and reason about their differences and similarities. For example, the Information Manifold system [LRO96] represents the contents of information sources as conjunctive queries over concepts and roles in the CLASSIC DL.

When viewed as a query language, the subsumption problem of DL's corresponds to the problem of query containment, which is a core tool for many of the tasks required in these applications. In fact, the restriction of DL's to unary and binary predicates yields new cases for which containment is decidable, whereas it is undecidable without this restriction. Recently, containment algorithms for a query language combining datalog and DL's, CARIN, has been developed [LR96a, LR96b].

In order to use description logics to express views and/or rewritings, we have to consider view definitions and queries that are conjunctive queries, where the relations may be complex concepts defined by constructors of description logics. In this paper, we consider the problem of rewriting queries using views when the views and the rewritings are conjunctive queries over a variant of the CLASSIC description logic (called *core*-CLASSIC). We show the following results. First, when the definitions of the views do not contain existential variables, it is possible to find a rewriting of the query by a nonrecursive datalog program. The resulting rewriting is guaranteed to produce the maximal set of answers possible from the views. Second, when the views do contain existential variables, we show that we may need to resort to recursive datalog rewritings, and we show that it is not always possible to find a rewriting of the query using the views that yields the maximal set of answers. We identify a class of conjunctive queries and databases for which we can find a maximally-contained rewriting. An important observation is that the general case of query rewriting involves the use of inference procedures that do not rely on the *unique-names assumption*, which is assumed in essentially all previous work on DL's. In fact, we show that when we remove the unique names assumption from the polynomial-time CLASSIC language, the complexity of satisfiability testing becomes NP-complete.

2 Preliminaries and Examples

We begin with a brief introduction to description logics and their main features. A description logic considers unary relations (called concepts) and binary relations (called roles). A description logic terminology includes a set of *primitive* concepts and roles and definitions of *complex* concepts and roles. The definitions of complex concepts and roles are given via *descriptions* that are built using a set of *constructors*.

Consider an example in which the primitive concepts are person, smart and footballPlayer, and child is a primitive role. Various constructors can be used to define new concepts by imposing some restrictions on the number of fillers of a role, or on the classes to which fillers of a certain role have to belong. For instance, $(\leq 3\,\mathsf{child})$ is a complex concept built by applying a *number restriction* constructor on the role child: it defines the set of objects that have at most 3 fillers for the role child. $(\forall\,\mathsf{child.smart})$ is a description which is built from the *universal quantification* constructor $\forall$, and defines the class of objects for which all fillers of the role child belong to the class smart. $\exists\mathsf{child.footballPlayer}$ uses the *existential quantification* constructor and defines the set of objects for which there exists at least one filler of the role child which belongs to the class footballPlayer. The following concept description describes the complex concept c1, which is defined to include set of persons such that all of their children are smart, have at most 3 children, and have a child who is a football player:

$$\mathsf{c1} := \mathsf{person} \sqcap (\forall\ \mathsf{child.smart}) \sqcap (\leq\ 3\ \mathsf{child}) \sqcap (\exists\ \mathsf{child.footballPlayer}).$$

In addition to a terminology, a description logic knowledge base includes a set of ground facts (called the Abox). The ground facts are assertions of the form $C(a)$ or $R(a,b)$, where C is a concept in the terminology, R is a role in the terminology (both can be primitive or complex), and a, b are individuals. For instance, child(Bob, Mark) states that Mark is Bob's child. $(\forall\,\mathsf{child.footballPlayer})(\mathsf{Bob})$ states that Bob is an instance of the class $(\forall\,\mathsf{child.footballPlayer})$, i.e, that all of Bob's children are football players.

We now define formally the syntax and semantics of the description logics considered in this paper. We consider two description logics: (1) $\mathcal{ALCNR}$, which is the most expressive description logic underlying an implemented system [BH91], and (2) a subset of the CLASSIC description logic, which is the most powerful description logic in which subsumption is known to be polynomial, and is used in several large-scale applications (e.g., [WWB+93]).

Syntax: Descriptions in $\mathcal{ALCNR}$ are defined using the following syntax (A denotes a primitive concept, P_i's denote primitive roles, C and D represent concept descriptions and R denotes a role description):

$C, D \rightarrow$	$A \mid$	(primitive concept)
	$C \sqcap D \mid C \sqcup D \mid$	(conjunction, disjunction)

$$
\begin{array}{lll}
 & \neg C \mid & \text{(negation)} \\
 & \forall R.C \mid & \text{(universal quantification)} \\
 & \exists R.C \mid & \text{(existential quantification)} \\
 & (\geq n\,R) \mid (\leq n\,R) & \text{(number restrictions)} \\
R \rightarrow & P_1 \sqcap \ldots \sqcap P_m & \text{(role conjunction)}
\end{array}
$$

The second language we consider, which we call *core*-CLASSIC, is a subset of $\mathcal{ALCNR}$, allowing only the constructors $\sqcap$, $\forall R.C$, $(\geq n\,R)$, $(\leq n\,R)$, negation only on primitive concepts, and no role constructors (i.e., only primitive roles).

We use the following terms in our discussion. The *depth* of concept is defined as follows. $depth(A) = depth(\leq\ n\,R) = depth(\geq\ n\,R) = 1$. The depth of $C_1 \sqcap C_2$ (resp. $C_1 \sqcup C_2$) is the maximum of $depth(C_1)$ and $depth(C_2)$, and $depth(\neg C) = depth(C)$. Finally, $depth(\forall\ R.C) = depth(\exists\ R.C) = depth(C) + 1$. The *size* of a description is the number of constructors it contains.

Semantics: The semantics of the constructors is given via *interpretations*. An interpretation I contains a non-empty domain $\mathcal{O}^I$. It assigns a unary relation A^I over $\mathcal{O}^I$ to every primitive concept A, and a binary relation R^I over $\mathcal{O}^I \times \mathcal{O}^I$ to every primitive role R. The interpretations of concept and role descriptions are uniquely defined by the following equations: ($\sharp\{S\}$ denotes the cardinality of a set S):

$$
\begin{array}{l}
(C \sqcap D)^I = C^I \cap D^I, \\
(C \sqcup D)^I = C^I \cup D^I,\ (\neg C)^I = \mathcal{O}^I \setminus C^I, \\
(\forall R.C)^I = \{d \in \mathcal{O}^I \mid \forall e : (d,e) \in R^I \rightarrow e \in C^I\}, \\
(\exists R.C)^I = \{d \in \mathcal{O}^I \mid \exists e : (d,e) \in R^I \wedge e \in C^I\}, \\
(\geq\ n\,R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \geq n\}, \\
(\leq\ n\,R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \leq n\}, \\
(P_1 \sqcap \ldots \sqcap P_m)^I = P_1^I \cap \ldots \cap P_m^I.
\end{array}
$$

Given a set of ground facts, G, we extend an interpretation with a constant-mapping ψ^I from the constants in G to $\mathcal{O}^I$. An interpretation I is said to be a *model* of G whenever $C(a) \in G$ then $\psi^I(a) \in C^I$, and whenever $R(a,b) \in G$ then $(\psi^I(a), \psi^I(b)) \in R^I$. The interpretation is said to satisfy the *unique-names assumption* if ψ^I is a 1-1 mapping. Most of the literature on description logics has made the unique-names assumption.

Terminologies: A *terminology* includes (1) a set of *concept definitions*, which are statements of the form $C := D$, where C is a concept name and D is a description, and (2) a set of *role definitions*, of the form $R := P$, where R is a role name and P is a role-description. Intuitively, a terminology is a set of view definitions, where the concept descriptions are the view expressions. The meaning of a concept definition is that in every interpretation I, C^I is defined to be D^I, and similarly for role definitions. We assume that a concept or role name appears on the left hand side of at most one definition. A concept name A is said to *depend* on a concept name B if B appears in the concept definition of A. We assume the terminology does not have cycles in the dependency relation. Similarly, we assume there are no cycles in the role definitions. Finally, we assume that all concept and role definitions are *unfolded*, i.e., D contains only primitive concepts and roles.

A concept description C is said to be *subsumed* by a concept description D, w.r.t. a terminology $\mathcal{T}$, if in every model I of $\mathcal{T}$, $C^I \subseteq D^I$. The questions of decidability and complexity of subsumption in different description logics have been studied extensively in the description logic literature. In particular, algorithms and complexity results for subsumption in CLASSIC are given in [BPS94], and for $\mathcal{ALCNR}$ in [BDS93].

Description Logic Systems: A description logic system provides two main services, based on the formal semantics of the underlying description logic. First, it provides algorithms for testing subsumption between concepts. Given two concept descriptions, it can determine whether one description *always* describes a superset of the individuals of the other. For example, the system could infer that the concept c1 is subsumed by the concept parent, which is defined as follows:

$$\mathsf{parent} := \mathsf{person} \sqcap (\geq\ 1\,\mathsf{child}).$$

The subsumption follows because the definition of c1 requires that there be a child who is a football player, and this condition already entails that the definition of parent will be satisfied.

The other service of a description logic system is *membership checking*. For example, if we assert person(Bob) and child(Bob,Mark), the system will derive that Bob is an instance of parent. However,the system won't derive that Bob is an instance of c1 (in particular because we don't know whether Mark is smart). Since description logic systems do not make the closed-world assumption, the system will also not derive that Bob is not a member of c1, because Bob may have other children that the system does not know about.

The main features of systems based on description logics are the following:

- Automatic classification: given a set of concept definitions, the system can automatically compute the appropriate subsumption hierarchy between them.
- Representation of partial information: in a description logic knowledge base we do not have to provide *all* the information about an individual. For example, it is possible to assert $(\geq 2\ \mathsf{child})(\mathsf{Bob})$, denoting that Bob has at least two children, but without specifying *who* the children are. The system will still be able to deduce that Bob is a parent from this partial information.
- Automatic recognition: given the information about an individual, the system can recognize to which other concepts it belongs. In database terms, this gives us the ability to assert that an individual should be in the extension of a view (or a set of views), and the system will determine which database relations the individual should belong to and to which other views it should belong.

Datalog programs over terminologies: a datalog program over a terminology $\mathcal{T}$ is a program whose EDB predicates are descriptions built from the concepts and roles in $\mathcal{T}$, and the IDB predicates do not appear in $\mathcal{T}$. A datalog program is a collection of horn-rules of the form:

$$q(\bar{X}) \quad :- \quad p_1(\bar{X}_1) \ \& \ \ldots \ \& \ p_n(\bar{X}_n)$$

where q is an IDB predicate, and $p_1, \ldots, p_n$ may be names or descriptions of concepts or roles defined in $\mathcal{T}$. The *head* of the rule is $q(\bar{X})$, and variables in the tuple $\bar{X}$ are called the head 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. Variables that appear in the body and not in the head are called *existential variables.* Note that the IDB predicates of the program can be of any arity (as opposed to description logics that consider only unary and binary relations). 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.*

In the same way that an interpretation of the primitive concepts and roles determines the interpretations of complex descriptions, it also uniquely determines the interpretation of the IDB predicates of a datalog program.[1] It should be noted that in the datalog rules, we consider complex concept descriptions and roles as EDB relation names.

Containment and subsumption: a query Q_1 is contained in a query Q_2 if for any interpretation I, the interpretation of the query predicate of Q_1 is a subset of the interpretation of query predicate of Q_2. Note that when Q_1 and Q_2 are concept definitions containment can be reduced to subsumption.

2.1 The Rewriting Problem

The problem of rewriting a query Q using a set of views $\mathcal{V}$ is to find a query expression Q' that uses only the views in $\mathcal{V}$, and is equivalent to the query Q (or is the maximal query contained in Q).

Definition 2.1: *Let $\mathcal{V}$ be a set of views, and let Q be a query. A query Q' is a maximally-contained rewriting of the query Q using $\mathcal{V}$ if*

1. *the relation names used in Q' are only the view names in $\mathcal{V}$,*
2. *Q' is contained in the query Q, and*
3. *there does not exist a query P that also only uses only the views $\mathcal{V}$, such that P is contained in Q, and Q' is contained but not equivalent to P.*

A maximally-contained rewriting Q' is an equivalent rewriting if Q' is equivalent to Q. □

In this paper we consider the *datalog-rewriting* problem. This is the case in which we allow $\mathcal{V}$ to be conjunctive queries over *core*-CLASSIC terminologies, and Q to be concept queries (i.e., the queries have one conjunct in their body, which is of the form $C(X)$, where C is a description). In this case it turns out that in order to find a maximally-contained rewriting we must allow Q' to be a datalog program over the *core*-CLASSIC terminology.

Without loss of generality, we can assume that in every view definition we add the disequality $X \neq Y$ for every pair of variables that appears in the head of the view. Furthermore, in the datalog rewritings, we assume that in every rule we have an atom $X \neq Y$ for every pair of variables appearing in the rewriting. For clarity, we omit these atoms in our examples and algorithms. Note that this does not limit the set of rewritings we can obtain.

Example 2.1: The following example illustrates the datalog-rewriting problem. Consider a terminology that includes the primitive concepts person and smart, and the role child, in which the concept happyParent is defined by:

$\mathsf{happyParent} := \mathsf{person} \sqcap (\forall\ \mathsf{child}.\mathsf{smart})$.

Suppose we have the following views:

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

Suppose our query is

$q(X) : -\mathsf{happyParent}(X)$.

We have the following maximally-contained rewriting of the query using the views:

$q'(X) : -v_3(X)$
$q'(X) : -v_1(X)\ \&\ v_2(X)$

The first rule gives us the happy parents that also have at least 2 children. The second rule 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 we cannot find an equivalent rewriting of the query using these views. However, if the query is

$p(X) : -\mathsf{happyParent}(X)\ \&\ (\geq\ 3\,\mathsf{child})(X)$.

then the following is an equivalent rewriting using the views:

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

[1] It is important to emphasize that our exposition here is limited to viewing description logics as a query language. Systems based on description logics allow in addition to defining a terminology also to provide a set of ground facts, for both primitive and complex concepts. Furthermore, the closed world assumption is not made in such systems. Consequently, evaluating a conjunctive query is much more involved than evaluating a conjunctive query from a completely specified interpretation [LR96a].

3 Conjunctive views without existential variables

In this section we consider the datalog-rewriting problem when the views $\mathcal{V}$ are conjunctive queries without existential variables. In the cases we consider here we show that the maximally-contained rewriting is always a union of conjunctive queries. Hence, we are able to follow the general approach of [LMSS95], which first provides a bound on the size of the conjunctive queries needed in the rewriting of the query, and then tests the candidate rewritings using algorithms for query containment. The bounds of [LMSS95] or the containment algorithms used there do not apply in the cases considered here. The theorems we describe below provide the applicable bounds. To test the candidate rewritings we use algorithms for subsumption and for knowledge-base satisfiability developed for description logics [BPS94, BDS93]. Furthermore, since the rewritings are unions of conjunctive queries, we can use the algorithm for containment of non-recursive datalog programs over $\mathcal{ALCNR}$ terminologies [LR96a], to test whether the maximally-contained rewriting is also an equivalent rewriting.

We begin with the simple but common case in which both the queries and the views are concept queries. In fact, the result for this case applies to the wider class of $\mathcal{ALCNR}$ queries. We show the following:

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 $\mathcal{ALCNR}$. The maximally-contained datalog-rewriting of Q using $\mathcal{V}$ is the union of conjunctive queries of the form:

$$Q'(X) \quad :- \quad V_{i_1}(X) \;\&\; \ldots \;\&\; V_{i_l}(X)$$

where $l \leq m$, $i_1, \ldots, i_l \in [1...m]$ and $C_{i_1} \sqcap \ldots \sqcap C_{i_l}$ is subsumed by C. □

The proof of the theorem is based on showing that there is no need to consider rewritings that contain existential variables in the rules. Therefore, we need only consider rewritings that are unions of conjunctive queries, where each conjunctive query has only subgoals of the form $V(X)$ where $V \in \mathcal{V}$, and X is the head variable.

Theorem 3.1 shows that the datalog-rewriting problem is decidable for queries and views that are concepts in $\mathcal{ALCNR}$, and tells us specifically how to generate the rewriting. However, it still requires that we consider a rule for every possible subset of the views. This is expensive if we have many views, which tends to be the case in applications such as information integration and data warehousing. The following theorem shows that when the views are in *core*-CLASSIC we can considerably prune the search for rewritings. In particular, it shows that the maximal size of the conjunctive queries that we need to consider for the rewritings is bounded by the size of the query and the maximal depth of a concept in 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 core-CLASSIC. *Let S be the maximal number of constructors in either C or in one of $C_1, \ldots, C_m$, and let k be the maximal depth of a concept in $C_1, \ldots, C_m$. Theorem 3.1 holds even if we restrict l to be $2kS$. In this case, the time complexity of finding a maximally-contained rewriting of C using $\mathcal{V}$ is $O(m^{kS})$.* □

The proof of Theorem 3.2 is based on using the *canonical form* of descriptions. Specifically, by pushing the $\sqcap$ constructor outside, we can reformulate every description D in *core*-CLASSIC as a conjunction D' of descriptions, each of the form $\forall R_1.\forall R_2.\ldots\forall R_n.\tau$, 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 D' is at most the number of constructors in D. If Q is a conjunction of view atoms and it is contained in D', then Q must be contained in each of the conjuncts of D'. We show that if Q is contained in a conjunct D_1, then a subset of Q containing at most $2k$ of Q's conjuncts is *already* contained in D_1. Therefore, the maximal length of Q that we need to consider is the product of $2k$ and the number of conjuncts in D'.

Next, we consider the case in which the views in $\mathcal{V}$ can be arbitrary conjunctive queries over a *core*-CLASSIC terminology, but without existential variables. Note that such queries allow subgoals that are role atoms in addition to concept atoms. Therefore, they are able to express properties of connections between variables. Consequently, subsumption algorithms alone are not sufficient to test query containment. The following theorem establishes a bound on the size of the rewriting of the query using the views. Given a conjunctive query using the views, the algorithm of Buchheit et al. [BDS93] can be used to check whether it is contained in the query.

Theorem 3.3: *Let Q be a query of the form*

$$q(X) : -C(X)$$

where C is a description in core-CLASSIC. *Let $\mathcal{V}$ be a set of conjunctive views over a* core-CLASSIC *terminology, such that in every view, all the body variables appear also in the head. Let S be the number of constructors in C and d be the depth of C. Let l be the maximal depth of a concept in C or $\mathcal{V}$, and m be the maximal number that appears in number restrictions in C or $\mathcal{V}$.*

Let $N = S(3l + m)m^d$. The maximally-contained 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.* □

In order to check whether a conjunctive query Q' using $\mathcal{V}$ is contained in the query, we consider the subgoals of Q' as a database and check whether the head of Q is entailed from that database using a slight modification of the algorithm described in [BDS93]. The crux of the proof is based on showing that if the query can be entailed, then it can be entailed by a database of bounded size, where the bound depends only on the terminology and the size of the query. This is the first result of its kind for description logics, and in fact, it does not hold for the language $\mathcal{ALCNR}$. It should be noted that the bound is based on the fact that models of the database must satisfy the unique-names assumption. Since the views do not have existential variables, and we use the $\neq$ predicate to distinguish every pair of head variables, the database constructed from the body of Q' can only be satisfied by interpretations in which the mapping on the constant-mapping satisfies the unique-names assumption.

4 Allowing Existential Variables in the Views

In this section we consider the datalog-rewriting problem in the case in which the view definitions of $\mathcal{V}$ contain existential variables. As the following example shows, the approach taken in the previous section will not be applicable in this case because there may be no *bound* on the size of the conjunctive rewritings that need to be considered.

Example 4.1: Consider the following two views:

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

Suppose the query Q is to find all individuals who have at least 2 children, i.e.:

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

For any n, the following conjunctive query is a rewriting that is contained in Q:

$$\begin{aligned} q'_n(X) \ \ :- \ \ & v_1(X,Y_1)\ \&\ v_1(Y_1,Y_2)\ \&\ \ldots\ \& \\ & v_1(Y_n,U)\ \&\ v_2(U). \end{aligned}$$

To see why, consider the variable Y_n. The view v_1 entails that it has one filler for the role child 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, i.e., $q(X)$ holds. The point is that inferring $q(X)$ required a chain of inference that involved an arbitrary number of constants, and the length of the chain does not depend on the query or the views.

Furthermore, for any value of n, we can build a database such that the union of $q'_1, \ldots, q'_n$ is *not* the maximally-contained rewriting, and therefore we cannot find a maximally-contained rewriting that is a union of conjunctive queries.

Interestingly, we can still find a maximally-contained rewriting in this example if we allow the rewriting to be a *recursive* datalog program. The following is a maximally-contained rewriting:

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

This recursive program essentially mimics the line of reasoning we described above. □

Example 4.1 leads to the question of under which conditions we can find a recursive datalog program that is a maximally-contained rewriting. In order for that to be possible, we must be able to mimic the inference needed to derive the query with a datalog program, as we did above. This entails that the inference must take time that is polynomial in the size of the data.

When views contain existential variables, we cannot guarantee that they are bound to *distinct* values. Therefore, the inference problem we are faced with concerns description logic knowledge bases in which the unique-names assumption does not necessarily hold. As the following theorem shows, the complexity results of inference in description logics heavily depend on the unique-names assumption. In particular, if we remove this assumption from *core*-CLASSIC, reasoning becomes NP-complete, whereas it is polynomial with the unique-names assumption [BPS94].

Theorem 4.1: *Let $\mathcal{T}$ be a* core-CLASSIC *terminology, and let G be a set of atoms for the concepts and roles appearing in $\mathcal{T}$. The problem of deciding whether there exists an interpretation of $\mathcal{T}$ that is a model of G, which does not necessarily satisfy the unique-names assumption is NP-complete in the size of G.* □

Proof: We prove the theorem by reduction from the 3-SAT problem. Suppose we have a 3-SAT problem Γ, described by the clauses $c_1, \ldots, c_m$, containing the propositional variables $p_1, \ldots, p_n$. We construct a *core*-CLASSIC knowledge base Δ in which the individuals $x_1, \ldots, x_n$, $y_1, \ldots, y_n$ and $z_1, \ldots, z_m$ do not satisfy the unique-names assumption.

The individuals $c_1, \ldots, c_m$ represent the clauses in Γ. The individuals $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ represent propositional literals, where the a's represent positive literals and the b's represent negative literals. We use the two disjoint concepts $(\leq\ 1\,R)$ and $(\geq\ 2\,R)$ to distinguish between literals that are assigned $True$ and $False$, respectively.

In order to assure that the a's and b's behave appropriately, (i.e., that for each i, exactly one is assigned $True$ and the other $False$), we use the individuals $l_1, \ldots, l_n$, and the atoms:

- $S(l_i, a_i)$, $S(l_i, b_i)$ and $(\leq 2\, S)(l_i)$,
- $S(l_i, y_i)$, $S(l_i, x_i)$, and
- $(\leq 1\, R)(x_i)$ and $(\geq 2\, R)(y_i)$

for $i \in [1 \ldots n]$.

Note that in any model I of Δ, x_i and y_i will be mapped to a_i^I and b_i^I (not necessarily respectively), and therefore, only interpretations in which the unique-names assumption does not hold can be models of Δ.

We use the role $contains(s,t)$ to denote that clause s contains the propositional literal t. Hence, if in Γ the literal p_i is in clause c_j, then Δ has the atom $contains(c_j, a_i)$, and if the literal $\neg p_i$ is in c_j, then Δ has the atom $contains(c_j, b_i)$. To encode the fact that every clause has at most 3 literals, Δ contains for every $j \in [1..m]$ the atoms $(\leq 3\, contains)(c_j)$.

In order to state that every clause is satisfied by at least one literal, we use the individuals $z_1, \ldots, z_m$, and the atoms:

- $contains(c_j, z_j)$ (c_j contains the literal z_j),
- $(\leq 1\, R)(z_j)$ (z_j is assigned $True$)

for $j \in [1 \ldots m]$.

The construction of Δ guarantees that Γ is satisfiable if and only if there is a model M of Δ whose domain is $a_1^M, \ldots, a_n^M, b_1^M, \ldots, b_n^M, c_1^M, \ldots, c_m^M$ and $l_1^M, \ldots, l_n^M$. The size of Δ is linear in the size of Γ, and therefore the reduction follows. $\Box$

The construction in the proof of Theorem 4.1 implies that even under severe limitations on the appearances of existential variables in the views, it may not be possible to find a maximally-contained rewriting. Specifically, as the following example shows, even if every existential variable in the view definition is connected (via a role atom) to only one head variable and to no other existential variables, there may not be a maximally-contained rewriting.

Example 4.2: Consider the following views.

$$\begin{array}{rcl} v_1(X,Y) & :- & contains(X,Y)\ \&\ (\leq 3\, contains)(X)\ \& \\ & & contains(X,Z)\ \&\ (\leq 1\, r)(Z) \\ v_2(X,Y) & :- & s(X,Y)\ \&\ (\leq 2\, s)(X) \\ v_3(X) & :- & s(X,Y)\ \&\ (\leq 1\, r)(Y) \\ v_4(X) & :- & s(X,Y)\ \&\ (\geq 2\, r)(Y). \end{array}$$

It follows from the proof of Theorem 4.1 that the extensions of views v_1-v_4 are a linear size encoding of a 3-SAT problem, where the concepts $(\leq 1\, r)$ and $(\geq 2\, r)$ encode the propositions that are assigned the truth values $True$ and $False$, respectively. Therefore, the following query asks for all the positive atoms that are entailed by a set of clauses.

$$q(X) \ :- \ (\leq 1\, r)(X)$$

Unless NP=P, there will be no datalog program that produces the maximally-contained rewriting for q, using v_1-v_4. The reason is that the complexity of evaluating the datalog program is polynomial in the size of the data. On the other hand, deciding whether a set of clauses entails the query is NP-hard in the size of the data. $\Box$

Given Example 4.2, we consider possible restrictions on the underlying data for which a maximally-contained rewriting always exists. It turns out that the cause of the high complexity of inference is the possible existence of *cycles* in the data. We can consider an interpretation I of description logic knowledge base as a graph, whose nodes are the objects in the domain of I, and there is an arc from node n_1 to node n_2 if the pair (n_1, n_2) is in the extension of one of the roles.

In what follows we show that if we only consider *tree models* of our data, then inference in *core*-CLASSIC without the unique-names assumption can be done in polynomial time. We then describe an algorithm for creating rewriting that is guaranteed to be a maximally-contained rewriting if we consider only tree models. In other cases, the algorithm will generate a correct rewriting, but it may not be the maximally-contained one.

We denote by $\Delta \models_T g$ the fact that all tree models of a knowledge base Δ are models of the atom g. We now describe a sound and complete inference procedure for deciding whether $\Delta \models_T D(a)$, where D is a concept description in *core*-CLASSIC, and Δ is a *core*-CLASSIC knowledge base. We consider knowledge bases for which the unique-names assumption does not hold. In such knowledge bases we distinguish between *individuals* and *variables*. An interpretation of such a knowledge base must map every individual to itself, but the variables may be mapped either to themselves or to individuals. We use the term *object* to denote the union of the individuals and the variables. Without loss of generality, we can assume that there are no primitive concepts or negations of primitive concepts. A pair of concepts A and $\neg A$ can be simulated by inventing a new role R_A, and defining the two new disjoint concepts $(\leq 1\, R_A)$ and $(\geq 2\, R_A)$. For simplicity of exposition we assume that a variable v in Δ can appear in

- only one binary atom, which has the form $R(X, v)$, where X is an individual, and
- unary atoms of the form $(\leq n\, R)(v)$ or $(\geq n\, R)(v)$.

Before we begin, it is important to note that an exponential-time inference procedure for the class of knowledge bases we consider can be easily derived from CLASSIC's inference procedure. However, our inference procedure has two important properties. First, it is based on a set of *propagation rules* that iteratively add additional facts to the knowledge base, which are guaranteed to terminate in polynomial time. Second, each propagation rule has the property that it considers only a bounded number of objects, where the bound depends only on the size of the concept D and the concept descriptions that appear in Δ (and not on the number of objects in Δ). These two properties will be the key to simulating the inference procedure with a datalog program.

In our description, we use the letters s, t to refer to objects, the letters a, b, c to refer to individuals and x, y, z to refer to variables. Since we do not make the unique-names assumption, we add to Δ explicit disequalities: for any pair (b, c) of individuals in Δ, we add $b \neq c$ to Δ.

A *number-restriction* formula describes the different possibilities for assigning numbers of role fillers to an object. Formally, a number-restriction formula is a disjunction of conjunctions of atoms. Each atom in the conjunctions is of the form $(\leq\ n\, R)(s)$ or $(\geq\ n\, R)(s)$. Only one object can appear in a number-restriction formula. We say that τ is a number-restriction formula for s if s is the object appearing in τ. The following is an example of a number-restriction formula:

$$[(\leq\ 2\, R_1)(s) \wedge (\leq\ 3\, R_2)(s) \wedge (\geq\ 3\, R_3)(s)] \vee$$
$$[(\leq\ 3\, R_1)(s) \wedge (\leq\ 2\, R_2)(s) \wedge (\geq\ 3\, R_3)(s)].$$

Note that given an integer M and a set of roles $R_1, \ldots, R_m$, there is a finite number of non-equivalent number-restriction formulas that involve the roles $R_1, \ldots, R_m$ and integers less or equal to M.

We denote by $\sigma(S, s)$ the set of all facts in the knowledge base S that involve the object s. We denote by $V(S, b)$ the set of variables connected to b in S by a role atom (i.e., the set of variables v, such that $R(b, v) \in S$). $\Sigma(S, b)$ denotes the set of all facts in S that involve either b or variables in $V(S, b)$. We denote by M the integer that is 1 more than the maximal integer that appears in restrictions of the form $(\leq\ n\, R)$ in Δ. The size of the concept D is denoted by L_D, and the number of roles in Δ by $\bar{R}$. Finally, we assume that all the concepts in Δ are in their canonical form.

The propagation rules are the following. S denotes the current state of the knowledge base when the premises of the propagation rule is being checked. We only add new facts if they are not already in the knowledge base (or, in the case of a number-restriction formula, we add it only if there is not an equivalent one already in the knowledge base).

1. If $(\forall\, R.C)(s), R(s, t) \in S$, add $C(t)$.
2. If $(\geq\ m\, R)(s) \in S$, and there are only l objects $s_1, \ldots, s_l$ such that $R(s, s_i) \in S$, and $l < m$, then add $m - l$ new individuals $a_1, \ldots, a_{m-l}$, and the facts $R(s, a_i)$ for $i \in [1...m - l]$. Add $a_i \neq a_j$ for every pair of individuals.
3. If $(\leq\ n\, R)(a)$,
$$R(a, s_1), \ldots, R(a, s_m), C(s_1), \ldots, C(s_m) \in S,$$
and
$$\sigma(S, s_1) \cup \ldots \cup \sigma(S, s_m) \models (\geq\ n\, R)(a),$$
then add $(\forall\, R.C)(a)$.
4. If $C_1(s), \ldots, C_n(s) \in S$ and $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq (\forall\, R.\bot)$, then add $(\leq\ 0\, R)(s)$.
5. If $l \leq M\bar{R}$, and
   - for every s_j, $j \in [1 \ldots l]$, there exists a role R such that $R(a, s_i) \in S$, and
   - $\Sigma(S, a) \cup \Sigma(S, s_1) \cup \ldots \cup \Sigma(S, s_l) \models g$, where g is a number-restriction formula of a or of one of $s_1, \ldots, s_l$,

   then add g.

Note that the third and fifth rules are semantic rules. To check the premises of these rules we need to check entailment of a disjunction of atoms from a set of facts. We can use a modification of the algorithm of [BPS94] for this entailment check. The point to note is that the complexity of checking the premises does not depend on the size of S, but only on the number of constants involved in the premise of the rule.

We apply the propagation rules until no new facts are added to the knowledge base. We denote the resulting knowledge base by Δ_f. The following theorem shows that the inference procedure is sound and complete.

Theorem 4.2: *If Δ has a satisfying tree model, then $\Delta \models_T D(a)$ if and only if for every conjunct D' in the canonical form of D, $D'(a) \in \Delta_f$.* □

Given Theorem 4.2, we can describe an algorithm for computing a maximally-contained datalog rewriting $\mathcal{P}$ of a query Q given a set of views $\mathcal{V}$. We assume that Q is of the form $q(X) : -D(X)$, where D is a concept in *core*-CLASSIC.

Given the views $\mathcal{V}$ and the query Q, there is a bound on the number of concepts that can appear in Δ_f, because we can only have concepts whose size is smaller than the maximal size of concept appearing in $Q \cup \mathcal{V}$, and uses the integers $[1...M]$. The program $\mathcal{P}$ includes an IDB predicate for every such concept and for every number-restriction formula appearing in Δ_f (i.e., if $g(a) \in \Delta_f$, then $\mathcal{P}$ contains an IDB predicate for g).

The key to constructing $\mathcal{P}$ is to bound the number of subgoals in its rules. The bound must guarantee that every way of instantiating our propagation rules with specific concept descriptions is mirrored by a datalog rule. In the worst case, every view literal will contribute one literal to the premises of a propagation rule. Therefore, we need to consider rules whose size is at most the maximal number of literals that can be used in the premises of a propagation rule.

The only subtle point in bounding the number of literals participating in every propagation rule is to bound the size of $V(\Delta, b)$. To do so, we observe that if two variables in $V(\Delta, b)$ have exactly the same constraints (i.e., $\sigma(\Delta, v_1) = \sigma(\Delta, v_2)$) then we can omit one of them, because the second does not add any information about b. If b appears twice in the same argument position of a view $V \in \mathcal{V}$, it will produce variables with exactly the same constraints. Therefore, it is enough to consider one occurrence of b in every argument position of every view in $\mathcal{V}$ in order to obtain all of $V(\Delta, b)$. We denote by B the maximal number of literals that can be used in a premise of one of our propagation rules.

The rules of $\mathcal{P}$ are defined as follows. Let A be a set of atoms of predicates of $\mathcal{P}$ (either views or IDB predicates), whose size is at most B. Let $C(X)$ be an atom, where C is an IDB predicate in $\mathcal{P}$ and X is a variable in A. We denote by A' the result of replacing the view literals in A by their expanded definitions. If $A' \models C(X)$, then we add the rule $C(X) : -A$ to $\mathcal{P}$.
The following theorem shows that $\mathcal{P}$ is a maximally-contained rewriting.

Theorem 4.3: *Let $Q : q(X) : -D(X)$ be a query where D is a concept in* core-Classic*, and let $\mathcal{V}$ be a set of conjunctive views over a* core-Classic *terminology, such that every existential variable Y in $\mathcal{V}$ appears in at most one role atom of the form $R(X, Y)$, where X is a head variable. For any tree database d, the program $\mathcal{P}$ is a maximally-contained rewriting of Q using $\mathcal{V}$.* □

Proof: (sketch) To prove the theorem we need to show that if Q' is a conjunctive query over the views $\mathcal{V}$, such that $Q' \subseteq Q$, then $Q' \subseteq \mathcal{P}$. If we consider the expansion of Q', Q'', then it is a *core*-Classic knowledge base for which Theorem 4.2 provides a sound and complete inference procedure. Therefore, there is a sequence of knowledge bases $S_1, \ldots, S_k$, such that $S_1 = Q''$, $D(X) \in S_k$, and S_k is obtained from applying a propagation rule to S_{k-1}. By induction on k it is easy to see that $\mathcal{P}$ will mimic the application of the propagation rules, if we apply it to the database including the subgoals of Q''. Hence, $Q' \subseteq \mathcal{P}$. □

5 Conclusions

We considered the problem of rewriting queries using views, when the views are conjunctive queries over the *core*-Classic description logic. We showed that when the views do not contain existential variables, we can find a maximally-contained rewriting of the query using the views, which is a union of conjunctive queries. For the case in which the views contain existential variables, we presented an algorithm that produces a recursive rewriting, and is guaranteed to be a maximally-contained rewriting when the underlying database forms a tree. In fact, the proof of Theorem 4.1 shows that even if the underlying database is a DAG, it is already not possible to always find a maximally-contained rewriting of the query using the views. It is also possible to show that our rewriting algorithm will produce a maximally-contained rewriting when, in the underlying database, there is a single undirected path between every pair of constants.

A second version of the rewriting problem is the *DL-rewriting* problem. In this version, the queries, the views and the rewritings are all expressed in the description logic. That is, a query is a concept description, and the answer to the query is the set of all individuals belonging to the concept. Solutions to the DL-rewriting problem are important also for the problem of *translating* between terminologies. In the extended version of the paper we show that the DL-rewriting problem is decidable when the queries and views are concepts in the expressive $\mathcal{ALCNR}$ description logic. When the query and views are concepts in *core*-Classic, it can be shown that the problem is decidable in polynomial time.

References

[AKS96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *International Journal on Intelligent and Cooperative Information Systems*, (6) 2/3:99–130, June 1996.

[BBM+91] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. Living with classic: When and how to use a KL-ONE-like language. In John Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, CA, 1991.

[BDS93] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[BH91] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *In Proceedings of the Workshop on Processing Declarative Knowledge, PDK-91, Lecture Notes in Artificial Intelligence*, pages 67–86. Springer-Verlag, 1991.

[Bor95] Alex Borgida. Description logics in data management. *IEEE Trans. on Know. and Data Engineering*, 7(5):671–682, 1995.

[Bor96] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82:353–367, 1996.

[BPS94] Alexander Borgida and Peter Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *Proceedings of International Conference on Data Engineering*, 1995.

[DG97] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona.*, 1997.

[FRV96] Daniela Florescu, Louiqa Rashid, and Patrick Valduriez. Answering queries using OQL view expressions. In *Workshop on Materialized Views, in cooperation with ACM SIGMOD, Montreal, Canada*, 1996.

[GMR95] Ashish Gupta, Inderpal Singh Mumick, and Kenneth A. Ross. Adapting materialized views after redefinitions. In *Proceedings of SIGMOD-95*, 1995.

[KB94] Arthur M. Keller and Julie Basu. A predicate-based caching scheme for client-server database architectures. In *Proceedings of PDIS-94*, 1994.

[LMSS95] 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.

[LR96a] 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.

[LR96b] 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.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd VLDB Conference, Bombay, India.*, 1996.

[LRU96] 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*, 1996.

[Mac88] R. M. MacGregor. A deductive pattern matcher. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 403–408, 1988.

[Pet91] C. Petalson. The BACK system : an overview. In *Proceedings of the SIGART bulletin*, volume 2(3), pages 114–119, 1991.

[Qia96] Xiaolei Qian. Query folding. In *Proceedings of the 12th International Conference on Data Engineering, New Orleans, LA*, pages 48–55, 1996.

[SDJL96] Divesh Srivastava, Shaul Dar, H. V. Jagadish, and Alon Y. Levy. Answering SQL queries using materialized views. In *Proceedings of VLDB*, 1996.

[TSI94] Odysseas G. Tsatalos, Marvin H. Solomon, and Yannis E. Ioannidis. The GMAP: A versatile tool for physical data independence. In *Proceedings of the 20th International VLDB Conference, Santiago, Chile*, pages 367–378, 1994.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of the International Conference on Database Theory*, 1997.

[WAF+93] W. Wahlster, E. Andre, W. Finkler, H.J. Profitlich, and T. Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63(1-2):387–428, 1993.

[WWB+93] J.R Wright, E.S. Weixelbaum, K. Brown, G.T Vesonder, S.R Palmer, J.I Berman, and H.H Moore. A knowledge-based configurator that supports sales, engineering and manufacturing at AT&T network systems. In *Proceedings of the Conference on Innnovative Applications of Artificial Intelligence Conference*, pages 183–193, 1993.

[YL87] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proceedings of the 13th International VLDB Conference*, pages 245–254, 1987.