Rewriting Conjunctive Queries using Views in Description Logics with Existential Restrictions

François Goasdoué and Marie-Christine Rousset
L.R.I., C.N.R.S. & University of Paris-Sud
Building 490, 91405, Orsay Cedex, France
Email: {goasdoue, mcr}@lri.fr

1 Introduction

In database, rewriting queries using views has received significant attention because of its relevance to several fields such as query optimization, data warehousing, and information integration. In those settings, data used to answer a query are restricted to be extensions of a set of predefined queries (views). The information integration context is typical of the need of rewriting queries using views for answering queries. Users of information integration systems do not pose queries directly to the (possibly remote) sources in which data are stored but to a set of virtual relations that have been designed to provide a uniform and homogeneous access to a domain of interest. When the contents of the sources are described as views over the (virtual) domain relations, the problem of reformulating a user query into a query that refers directly to the relevant sources becomes a problem of rewriting queries using views.

The instances of the rewriting problem that have been considered vary depending on the languages used for describing the queries and the views. The rewriting problem that has been extensively studied concerns the pure relational setting where queries and views are expressed as conjunctive queries over relational atoms (see [7] for a recent survey).

The first extension to DL has been considered in [2], which studied several instances of the problem of rewriting queries using views when the queries and the views are conjunctive queries *over DL expressions*. [8] deals with the problem of rewriting (union of) conjunctive queries over $\mathcal{ALN}$ expressions into conjunctive queries built on a set of views that are a set of distinguished $\mathcal{ALN}$ expressions. The rewriting algorithm presented in [8] has been implemented for answering users queries in the PICSEL information integration system [5]. In [1], the rewriting problem that is considered is still different since given a concept C expressed in a DL $\mathcal{L}_1$ and a TBox $\mathcal{T}$ expressed in a DL $\mathcal{L}_2$, the authors want to compute a minimal concept E in a DL $\mathcal{L}_3$ that is equivalent to C modulo $\mathcal{T}$.

In this paper, extending the work of [8], we study the problem of rewriting conjunctive queries over DL expressions into conjunctive queries using a set of views that are a set of distinguished DL expressions, for three DLs allowing existential restrictions: $\mathcal{FLE}$, $\mathcal{ALE}$ and $\mathcal{ALNE}$. Thus, our rewriting problem is: given a conjunctive query over expressions from a DL $\mathcal{L} \in \{\mathcal{FLE}, \mathcal{ALE}, \mathcal{ALNE}\}$ and a set of views $\mathcal{V}$ over expressions from $\mathcal{L}$, we want to compute a *representative set* of all the rewritings of the query that are conjunctive queries over $\mathcal{V}$. By *representative set* we mean that this set contains at least the rewritings that are maximally contained in the query.

For $\mathcal{FLE}$, we present an algorithm that computes a representative set of all the rewritings of a query. In the full version of this paper [6], we show how to adapt it to deal with the negation of atomic concepts in the queries and in the views, in order to obtain a rewriting algorithm for $\mathcal{ALE}$. Finally, we show that obtaining a *finite* set representative of all the rewritings of a query is not guaranteed in $\mathcal{ALNE}$.

It is important to emphasize that rewriting conjunctive queries into views provides an efficient way to effectively answer conjunctive queries posed to large while regular Aboxes. We consider an Abox as regular when it is composed of extensions of a fixed and rather small number of concepts and roles (compared to the number of facts). Answering conjunctive queries over Aboxes is an important issue that is crucial to solve in an effective way if we want to export DL in real applications dealing with huge amount of data. The problem of answering *boolean* conjunctive queries over Aboxes has been studied from a theoretical point of view in [3] within the setting the very expressive DL $\mathcal{DLR}$. However, no algorithm is provided that could lead to an efficient computation of the answer sets of queries. The approach that we advocate for answering a conjunctive query posed to a regular Abox, composed of extensions of the concepts or roles $cr_1, \ldots, cr_n$, is to (1) rewrite the query using views in $cr_1, \ldots, cr_n$, (2) evaluate each rewriting by considering it as a relational query posed to a standard relational database. Such an approach has the advantage to guarantee that the computation of the answer set of a query is polynomial in the size of the data. The completeness of this computation depends on the kind of rewritings (equivalent or maximally contained) and views (sound, complete or exact) that are used [4].

The paper is organized as follows. In section 2, we formally define the rewriting problem that we consider. We provide a sound and complete rewriting algorithm for $\mathcal{FLE}$ in section 3. Finally, in the last section, we exhibit a counterexample showing that obtaining a finite set of representative rewritings is not guaranteed for $\mathcal{ALNE}$.

2 Preliminaries and examples

The views and the queries in the rewriting problem that we consider in this paper are built on $\mathcal{L}$ concept descriptions and roles, where $\mathcal{L}$ is first $\mathcal{FLE}$, then $\mathcal{ALE}$, and finally $\mathcal{ALNE}$. The $\mathcal{FLE}$ concept descriptions can be built from atomic concepts (including $\top$) and roles, using the constructors of conjunction $(C_1 \sqcap C_2)$, value restriction $(\forall r.C)$, and existential restriction $(\exists r.C)$. In addition to those constructors, $\mathcal{ALE}$ allows the primitive negation, i.e., negation $(\neg A)$ restricted to atomic concepts, and $\bot$. Finally, if we add the number restrictions $((\geq\ n\, r), (\leq\ n\, r))$, we obtain $\mathcal{ALNE}$.

In the rest of the paper, we consider concept descriptions that have been put in *normal form*, obtained by exhaustively applying the following normalization rules which preserves equivalence: $\forall r.(E \sqcap F) \longrightarrow \forall r.E \sqcap \forall r.F$, $\forall r.E \sqcap \exists r.F \longrightarrow \forall r.E \sqcap \exists r.(E \sqcap F)$, $\forall r.\top \longrightarrow \top$, $E \sqcap \top \longrightarrow E$. Note that the size of the normal form computed by those normalization rules may be exponential in the size of the original concept.

An important role is played by *Q-concepts* that are of the form $Qr.D$, where $Q \in \{\exists, \forall\}$. We define inductively the notions of *q-role chains* and *decomposition w.r.t. a q-role chain* of a Q-concept which will be important in our algorithm.

Definition 2.1: *q-role chains of a Q-concept* $Qr.\bigsqcap_{i=1}^{n} D_i$ *are defined inductively as follows:*

1. *$[Qr]$ is a q-role chain of depth 1 of $Qr.\bigsqcap_{i=1}^{n} D_i$ iff no D_i is a Q-concept. The only one decomposition (of depth 1) of $Qr.\bigsqcap_{i=1}^{n} D_i$ w.r.t. the q-role chain $[Qr]$ is: $decomp(Qr.\bigsqcap_{i=1}^{n} D_i, [Qr], 1) = \bigsqcap_{i=1}^{n} D_i$,*

2. *$[Qr \mid ch]$ is a q-role chain of depth $k+1$ of $Qr.\bigsqcap_{i=1}^{n} D_i$ iff there exists $p \in [1..n]$ such that ch is a q-role chain of depth k of D_p. The decompositions of depth m ($\forall m \in [1..k+1]$) of $Qr.\bigsqcap_{i=1}^{n} D_i$ w.r.t. the chain $[Qr \mid ch]$ are:*
   $decomp(Qr.\bigsqcap_{i=1}^{n} D_i, [Qr \mid ch], 1) = \bigsqcap_{i=1}^{p-1} D_i \sqcap \bigsqcap_{i=p+1}^{n} D_i$
   $decomp(Qr.\bigsqcap_{i=1}^{n} D_i, [Qr \mid ch], m+1) = decomp(D_p, ch, m)$, $\forall m \in [1..k]$.

Example 2.1: Consider the $\exists$-concept $C : \exists r_1.(A_1 \sqcap \exists r_2.\exists r_3.(A_2 \sqcap A_3) \sqcap A_4)$. Its decompositions w.r.t. the chain $[\exists r_1, \exists r_2, \exists r_3]$ are: $decomp(C, [\exists r_1, \exists r_2, \exists r_3], 1) = A_1 \sqcap A_4$, $decomp(C, [\exists r_1, \exists r_2, \exists r_3], 2) = \top$ (due to the semantics of the empty conjunction), $decomp(C, [\exists r_1, \exists r_2, \exists r_3], 3) = A_2 \sqcap A_3$.

We will have to do the dual operation to build concept expressions from a list of roles to which we apply (existential or universal) restrictions, in which concepts are injected at each depth of the role chain.

Definition 2.2: *Given a sequence* $[Q_1 \mid l_q]$ *of* n *quantifiers* $\exists$ *or* $\forall$, *a sequence* $[r_1 \mid l_r]$ *of* n *roles and a sequence* $[C_1 \mid l_c]$ *of* n *concepts:*

- $compose([Q][r][C]) = Qr.C$,
- $compose([Q_1 \mid l_q], [r_1 \mid l_r], [C_1 \mid l_c]) = Q_1 r_1.(C_1 \sqcap compose(l_q, l_r, l_c))$.

Example 2.2: The composition of $[\exists, \exists, \forall]$, $[r_1, r_2, r_3]$ and $[A_1 \sqcap A_2, \forall r.B_1, A_3]$ is the concept: $\exists r_1.(A_1 \sqcap A_2 \sqcap \exists r_2.(\forall r.B_1 \sqcap \forall r_3.A_3))$.

Finally, we will have to consider the decomposition of a conjunction of concepts into k conjunctions of concepts.

Definition 2.3: *Let* C *be a concept obtained as the conjunction of a set* $\mathcal{C}$ *of concepts. A* k*-decomposition of* C *is a set of concepts* $\{D_1 \dots D_k\}$ *such that:*

- *each* D_i *is the conjunction of the concepts of a subset* $\mathcal{C}_i$ *of* $\mathcal{C}$,
- $\mathcal{C}_1, \dots, \mathcal{C}_k$ *is a partition of* $\mathcal{C}$.

2.1 Conjunctive queries over concepts and roles

We consider conjunctive queries of the form: $Q(\bar{X}) : \bigwedge_{i=1}^{n} p_i(\bar{X}_i, \bar{Y}_i)$, where the $p_i(\bar{X}_i, \bar{Y}_i)$'s are concept-atoms or role-atoms. We call *concept-atom* an atom $p(U)$ where p is a concept expression, and *role-atom* an atom $r(U_1, U_2)$ where r is a role name. The variables of $\bar{X} = \bigcup_{i=1}^{n} \bar{X}_i$ are called the *distinguished* variables of the query: they represent the variables of the query which the user is interested in knowing the instances of, when he asks the query. The variables that are not distinguished (denoted by $\bigcup_{i=1}^{n} \bar{Y}_i$) are called *existential* variables. They are existentially quantified and their use is to constrain the distinguished variables.

The answer of a query $Q(\bar{X})$ posed against a set of ground facts db is the set of tuples $\bar{a}$ such that: $db \models \exists \bar{Y}\, Q(\bar{a})$. If $Q'(\bar{X})$ is a query that entails $Q(\bar{X})$, the answers of $Q'(\bar{X})$ against db are included in the answers of $Q(\bar{X})$ (against the same db).

Example 2.3: Consider the query:

$$\begin{aligned} Q(X_1, X_2) :\ & Flight(X_1) \wedge Flight(X_2) \wedge Airline(X_1, U) \wedge Airline(X_2, U) \\ & \wedge (\forall Stop.AmCity)(X_1) \wedge Stop(X_2, S) \wedge EurCity(S) \end{aligned}$$

It contains the distinguished variables X_1, X_2 and the existential variables U, S. Its answers are all the possible instances of the tuple of distinguished variables X_1, X_2 that are flights depending on the same airline and such that the flights X_1 only stop (if they stop) in American cities, and the flights X_2 have a stop in an European city.

A variable U has a variable V as a successor (respectively predecessor) in a query if there exists a role atom $r(U, V)$ (respectively $r(V, U)$) in the query. The binding of the variables appearing in a query (or a subquery) can be described by a graph whose nodes are the variables and such that there is an edge from any variable to any of its successor. We say that a subquery has a tree structure if the associated graph is a tree.

We apply the following algorithm in order to get an equivalent *normalized query*:

1. Replacing exhaustively the different conjuncts $C_i(U)$ related to a same variable U in the query by a single concept-atom $C(U)$, where C is the normal concept form of the conjunction of the C_i's.

2. Replacing the subqueries having a tree structure by single concept-atoms (called their *Descriptive Support* as in [8]): it consists of iteratively replacing conjuncts $r(U, Y) \wedge C(Y)$ (or $r(U, Y)$ if there is no conjunct of the form $C(Y)$), for existential variables Y that have one and only one predecessor and no successor, by a single concept-atom $(\exists r.C)(U)$ (or $(\exists r.\top)(U)$).

3. Finally by replacing every concept-atom $(C \sqcap D)(X)$ where X is a distinguished variable by the conjuncts $C(X) \wedge D(X)$.

The underlying ideas of this computation is (1) to bring to the fore the implicit knowledge contained in the conjunction of the C_i's thanks to the normalized concept description C, (2) to capture most of the query's knowledge in concept descriptions and (3) to decompose any concept-atom in several concept-atoms that can by rewritten independently.

Example 2.4: The result of the normalization of the query $Q(X_1, X_2)$ (cf. example 2.3) is:

$$\begin{aligned} Q_{norm}(X_1, X_2) : \quad & Flight(X_1) \wedge Flight(X_2) \wedge Airline(X_1, U) \wedge Airline(X_2, U) \\ & \wedge (\forall Stop.AmCity)(X_1) \wedge (\exists Stop.EurCity)(X_2) \end{aligned}$$

Note that the existential variable S appearing in $Q(X_1, X_2)$ does not appear anymore in $Q_{norm}(X_1, X_2)$, while the existential variable U remains: the variable binding on U prevents from replacing the conjuncts $Airline(X_1, U) \wedge Airline(X_2, U)$ by existential restrictions on the role $Airline$.

2.2 The problem of rewriting queries using views

We suppose that we are given a finite set $\mathcal{V} = \{v_1 : def(v_1), \ldots, v_k : def(v_k)\}$ of views, where the v_i's are the names of the views, and $def(v)$ is the definition of the view named v. We consider views whose definitions are either roles or concept descriptions of $\mathcal{FLE}$ (respectively $\mathcal{ALE}$, $\mathcal{ALNE}$). If $def(v)$ is a role name (respectively a concept description), we call v a *role-view* (respectively a *concept-view*), and $v(X, Y)$ (respectively $v(X)$) is called a *view atom*.

Example 2.5: Suppose that we have the following set of views $\mathcal{V}$:
$\mathcal{V} = \{v_1 : Airline, v_2 : Stop, v_3 : HotelLocation, v_4 : Flight \sqcap \forall Stop.(AmCity \sqcap StateCapital), v_5 : HiltonHotel \sqcap \forall HotelLocation.EurCity\}$

The three first views are role-views: they express that we can access to extensions of the roles *AirLine*, *Stop*, and *HotelLocation*. The other ones are concept views. They indicate the concepts whose instances are available: flights which only stop in American cities that are also capitals of a state, and Hilton hotels that are located in an European city.

We normalize the set of views by splitting the concept-views $v : C_1 \sqcap \ldots \sqcap C_n$ in new concept-views $v : C_1 \ldots v : C_n$.

Example 2.6: By normalizing the views of $\mathcal{V}$ in example 2.5, we obtain the set of normalized views:

$$\begin{aligned}\mathcal{V}_{norm} = \{&v_1 : Airline, v_2 : Stop, v_3 : HotelLocation, v_4 : Flight,\\ &v_4 : \forall Stop.AmCity, v_4 : \forall Stop.StateCapital, v_5 : HiltonHotel,\\ &v_5 : \forall HotelLocation.EurCity\}\end{aligned}$$

Note that in a normalized set of views, a given name v can appear in several $v : C_i$'s. When we consider a normalized set of views $\mathcal{V}$, the definition of a view v is obtained by taking the conjunction of all the C_i's associated with v in $\mathcal{V}$.

Let $Q_{\mathcal{V}}(\bar{U})$ be a conjunction of view atoms. Its *definition*, denoted $Q_{def(\mathcal{V})}(\bar{U})$, is the conjunction of concept-atoms and role-atoms obtained by replacing each view name in $Q_{\mathcal{V}}(\bar{U})$ by its definition.

The rewriting problem that we consider can now be formally defined:

Definition 2.4: *Let $Q(\bar{X})$ be a normalized query whose existential variables are $\bar{Y}$. Let $\mathcal{V}$ be a set of normalized views. Let $Q_{\mathcal{V}}(\bar{X}, \bar{Z})$ be a conjunction of view atoms. $Q_{\mathcal{V}}(\bar{X}, \bar{Z})$ is a rewriting of $Q(\bar{X})$ iff its definition entails $Q(\bar{X})$, i.e., $\exists \bar{Z}\, Q_{def(\mathcal{V})}(\bar{X}, \bar{Z}) \models \exists \bar{Y}\, Q(\bar{X})$.*

Example 2.7: Continuing our example, consider the following query, which is the subquery of the query $Q(X_1, X_2)$ which has not a trivial rewriting using $\mathcal{V}_{norm}$:

$$Q'(X_2) : (\exists Stop.EurCity)(X_2)$$

$Q'_{\mathcal{V}}(X_2) : v_2(X_2, Y) \wedge v_3(U, Y) \wedge v_5(U)$ is a rewriting of $Q'(X_2)$ using $\mathcal{V}_{norm}$, because its definition, $Stop(X_2, Y) \wedge HotelLocation(U, Y) \wedge (HiltonHotel \sqcap \forall HotelLocation.EurCity)(U)$, entails $(\exists Stop.EurCity)(X_2)$.

3 Computation of rewritings in $\mathcal{FLE}$

The rewriting algorithm takes as input a normalized query $Q(\bar{X})$, and a normalized set of views $\mathcal{V}$. It returns as output a set of conjunctions of view atoms that is a representative set of all the rewritings of $Q(\bar{X})$ using views in $\mathcal{V}$.
The core procedure of the algorithm is $Rewrite(p(\bar{U}), \mathcal{V})$ which computes the rewritings of a (role or concept) atom $p(\bar{U})$ using $\mathcal{V}$. Based on that procedure, the set of rewritings of $Q(\bar{X}) : \bigwedge_{i=1}^{n} p_i(\bar{X}_i, \bar{Y}_i)$ is then obtained by building all the conjunctions of rewritings of its conjuncts, which we denote: $\bigotimes_{i=1}^{n} Rewrite(p_i(\bar{U}_i), \mathcal{V})$. Note that if there exists k $(1 \leq k \leq n)$ such that $Rewrite(p_k(\bar{U}_k), \mathcal{V}) = \emptyset$ then $\bigotimes_{i=1}^{n} Rewrite(p_i(\bar{U}_i), \mathcal{V}) = \emptyset$.
The result of the rewriting of a role-atom $r(U_1, U_2)$ is trivial, because the query has been normalized: it consists of the view atoms $v(U_1, U_2)$ such that $v : r \in \mathcal{V}$.
The rewriting procedure for a concept-atom $C(U_1)$ is given in Figure 1.

Example 3.1: Let us illustrate on an example the complex case of the algorithm, the rewriting of the Q-concept-atom $C(U) = (\exists r_1.(A \sqcap \exists r_2.B))(U)$. C contains only one q-role chain of depth 2: $ch = [\exists r_1, \exists r_2]$. The decomposition of depth 1 (respectively 2) of C w.r.t. ch is $D_1 : \{A\}$ (respectively $D_2 : \{B\}$). The arrays of quantifiers are:

$\exists$	$\exists$
$\forall$	$\forall$
$\forall$	$\forall$

$\forall$	$\exists$
$\exists$	$\forall$
$\forall$	$\forall$

From the first array, we obtain the rewritings:
$(\exists r_1.(A \sqcap \exists r_2.\top))(U) \wedge (\forall r_1.\forall r_2.B)(U), (\exists r_1.\exists r_2.B)(U) \wedge (\forall r_1.A)(U),$
$(\exists r_1.\exists r_2.\top))(U) \wedge (\forall r_1.A)(U) \wedge (\forall r_1.\forall r_2.B)(U).$
From the second one, we obtain as rewritings:
$(\forall r_1.\exists r_2.B)(U) \wedge (\exists r_1.A)(U), (\forall r_1.\exists r_2.B)(U) \wedge (\exists r_1.\top)(U) \wedge (\forall r_1.A)(U),$
$(\forall r_1.\exists r_2.\top)(U) \wedge (\exists r_1.(A \sqcap \forall r_2.B))(U),$
$(\forall r_1.\exists r_2.\top)(U) \wedge (\exists r_1.A)(U) \wedge (\forall r_1.\forall r_2.B)(U),$
$(\forall r_1.\exists r_2.\top)(U) \wedge (\exists r_1.\forall r_2.B)(U) \wedge (\forall r_1.A)(U),$
$(\forall r_1.\exists r_2.\top)(U) \wedge (\exists r_1.\top)(U) \wedge (\forall r_1.A)(U) \wedge (\forall r_1.\forall r_2.B)(U).$

The following theorem states the correctness and the completeness of the atom rewriting algorithm given in Figure 1.

Theorem 3.1: *Let $p(\bar{U})$ be a conjunct of a normalized query, and let $\mathcal{V}$ be a normalized set of views. Let $result = \{Q_{\mathcal{V}_1}(\bar{U}, \bar{Z}_1) \ldots Q_{\mathcal{V}_n}(\bar{U}, \bar{Z}_n)\}$ be the output returned by $Rewrite(p(\bar{U}), \mathcal{V})$:*

1. *each $Q_{\mathcal{V}_i}(\bar{U}, \bar{Z}_i)$ is a rewriting of $p(\bar{U})$,*

2. *every rewriting of $p(\bar{U})$ includes (up to a renaming of existential variables) all the view-atoms of atleast one rewriting returned by $Rewrite(p(\bar{U}), \mathcal{V})$.*

procedure $Rewrite(C(U_1), \mathcal{V})$

$result := \{v_0(U_{m+1}) \wedge \bigwedge_{i=1}^{m} v_i(U_{m+2-i}, U_{m+1-i}) \mid v_0 : \forall r_1 \ldots \forall r_m.D \in \mathcal{V}$ s.t $D \preceq C, \forall i \in [1..m]\ v_i : r_i \in \mathcal{V}\}$

(* note that if $m = 0$ $result := \{v_0(U_1) \mid v_0 : D \in \mathcal{V}$ s.t $D \preceq C\}$ *)

if $C(U_1) = (\exists r.\top)(U_1)$ **then:**
  $result := result \cup \{v(U_1, U_2) \mid v : r \in \mathcal{V}$ where U_2 is a fresh variable$\}$

if $C(U_1) = (\exists r.D)(U_1)$, $D \neq \top$, **then:**
  $result := result \cup \{v(U_1, U_2) \mid v : r \in \mathcal{V}\} \otimes Rewrite(D(U_2), \mathcal{V})$
  where U_2 is a fresh variable

if $C(U_1) = (\prod_{i=1}^{n} C_i)(U_1)$ **then:**
  $result := result \cup \bigotimes_{i=1}^{n} Rewrite(C_i(U_1), \mathcal{V})$

if $C(U_1) = (Q_1 r_1.D)(U_1)$, $D \neq \top$, **then:**
  for each q-role chain $ch = [Q_1 r_1, \ldots, Q_c r_c]$ of depth c of $Q_1 r_1.D$:
    let n be the number of Q_i occurring in ch that are $\exists$,
    let D_i be the decomposition of depth i of $Q_1 r_1.D$ w.r.t. $[Q_1 r_1, \ldots, Q_c r_c]$, $\forall i \in [1..c]$.
    for each k-decomposition of each $D_i : \{D_i^1, \ldots, D_i^k\}$, $k \in [2..n+1]$:
      for each array of quantifiers $\exists$ or $\forall$ (modulo line permutations):

$$\begin{array}{ccc} Q_{11} & \ldots & Q_{1c} \\ \ldots & \ldots & \ldots \\ Q_{k1} & \ldots & Q_{kc} \end{array}$$

      where there is exactly one $\exists$ in the column i if $Q_i = \exists$ in ch, and none otherwise.
        for p = 1 to k:
          $C_p = Normalization(compose([Q_{p1}, \ldots, Q_{pc}], [r_1, \ldots, r_c], [D_1^p, \ldots, D_c^p]))$
        let E_q's be the C_p's that are not $\top$, $q \in [1..k']$, $k' \in [1..k]$.
        if there is no E_p syntacticaly equal to C **then:**
          $result := result \cup \bigotimes_{q=1}^{k'} Rewrite(E_q(U_1), \mathcal{V})$

if $C(U_1)$ is a concept-atom of the query, where U_1 is an existential variable **then:**
  (* in that case C can be a conjunction of concepts *)
  for each collection $\{r_1, \ldots, r_m\}$ of roles appearing in view definitions, such that $1 \leq m \leq$ the maximal depth of concept expressions $\exists r.D$ appearing in the view definitions:
    $result := result \cup Rewrite((\exists r_1 \ldots \exists r_m.C)(Y), \mathcal{V})$
    where Y is a fresh variable

return result.

Figure 1: Rewriting algorithm of a $\mathcal{FLE}$ concept-atom

The proof is given in [6]. The following corollary states the correctness and the completeness of our query rewriting algorithm.

Corollary 3.2: *Let $Q(\bar{X}) : \bigwedge_{i=1}^{n} p_i(\bar{U}_i)$ be a normalized query:*

1. *each element of $\bigotimes_{i=1}^{n} Rewrite(p_i(\bar{U}_i), \mathcal{V})$ is a rewriting of $Q(\bar{X})$,*

2. *every rewriting of $Q(\bar{X})$ includes (up to a renaming of existential variables) atleast one rewriting returned by $\bigotimes_{i=1}^{n} Rewrite(p_i(\bar{U}_i), \mathcal{V})$.*

4 Conclusion

In the extended version of this paper ([6]), we show how to adapt our rewriting algorithm for $\mathcal{FLE}$ in order to take $\mathcal{ALE}$ concept descriptions into account in the query and in the view definitions. For $\mathcal{ALE}$, we cope with the atomic negation and $\perp$ by adding new normalization rules: $A \sqcap \neg A \longrightarrow \perp$, $\exists r.\perp \longrightarrow \perp$, $C \sqcap \perp \longrightarrow \perp$.

Moreover, we modify the *Rewrite* procedure in order to allow the following rewritings:

$$\underbrace{(\forall r_1 \ldots \forall r_n.\perp)(U_1)}_{\text{rewriting}} \models \underbrace{(\forall r_1 \ldots \forall r_m.D)(U_1)}_{\text{concept-atom of the query}}, n \in [1..m], m \geq 1$$

$$\underbrace{(\forall r_1 \ldots \forall r_m.A)(U_1) \wedge (\forall r_1 \ldots \forall r_m.\neg A)(U_1)}_{\text{rewriting}} \models \underbrace{(\forall r_1 \ldots \forall r_m.\perp)(U_1)}_{\text{concept-atom of the query}}, m \geq 1$$

$$\underbrace{(\forall r_1 \ldots \forall r_m.\exists r^1 \ldots \exists r^n.(A \sqcap \neg A))(U_1)}_{\text{rewriting}} \models \underbrace{(\forall r_1 \ldots \forall r_m.\perp)(U_1)}_{\text{concept-atom of the query}}, n \geq 1, m \geq 1$$

We also show that, for $\mathcal{ALE}$, we need to verify that a rewriting $Q_{\mathcal{V}}(\bar{X}, \bar{Z})$ is consistent. To achieve this, it suffices to check that during the normalization of its definition $Q_{def(\mathcal{V})}(\bar{X}, \bar{Z})$, $\perp(U)$ is not produced.

We will end this paper by exhibiting a counterexample showing that obtaining a finite set of representative rewritings is not guaranteed for $\mathcal{ALNE}$. Consider the set of views $\mathcal{V} : \{v_1 : r, v_2 : \exists r.(\leq 1\, r), v_3 : (\geq 3\, r)\}$, and the query $Q(X) : (\geq 2\, r)(X)$.

For any n, the following is a rewriting of $Q(X)$: $v_1(X, Y_1) \wedge v_2(X) \wedge v_1(Y_1, Y_2) \wedge v_2(Y_1) \wedge \ldots \wedge v_1(Y_{n-1}, Y_n) \wedge v_2(Y_{n-1}) \wedge v_3(Y_n)$.

To see why, consider the variable Y_{n-1}. The view atom $v_2(Y_{n-1})$ entails that it has one r- successor that has itself less than one r- successor while the view atoms $v_1(Y_{n-1}, Y_n)$ and $v_3(Y_n)$ state that its r-successor Y_n has at least 3 r-successors. Therefore, Y_{n-1} has at least 2 r-successors. The same line of reasoning can be used to see that Y_{n-2} has at least 2 r-successors and continuing in the same way, we get that X has at least 2 r-successors, i.e., the query $Q(X) : (\geq 2\, r)(X)$ holds.

The point is that inferring $Q(X) : (\geq\ 2\,r)(X)$ required a chain of inference that involved an arbitrary number of view atoms, and the length of the chain does not depend on the query or the views. It is easy to check that none of those rewritings is contained in another one. Therefore, it cannot exist a finite set of conjunctions of view atoms that is representative of all the possible rewritings of $Q(X) : (\geq\ 2\,r)(X)$.

References

[1] Franz Baader, Ralf Kurster and Ralf Molitor. Rewriting Concepts Using Terminologies. In *Proceedings of KR'2000*, 2000.

[2] Catriel Beeri, Alon Y.Levy and Marie-Christine Rousset. Rewriting Queries Using Views in Description Logics. In *Proceedings of PODS'97*, 1997.

[3] Diego Calvanese, Giuseppe De Giacomo and Maurizio Lenzerini. Answering Queries Using Views in Description Logics. In *Proceedings of DL'99 and KRDB'99.*

[4] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini and Moshe Y. Vardi. View-based Query Processing and Constraint Satisfaction. In *Proceedings of LICS-00, 2000.* http://www.dis.uniroma1.it/pub/calvanes/calv-degi-lenz-vard-lics-2000.ps.gz

[5] François Goasdoué, Véronique Lattes and Marie-Christine Rousset. The Use of Carin Language and Algorithms for Information Integration: the Picsel System. *To appear in International Journal on Cooperative Information Systems.*

[6] François Goasdoué and Marie-Christine Rousset. Rewriting Conjunctive Queries using Views in Description Logics with Existential Restrictions. *Technical report.* http://www.lri.fr/~goasdoue/biblio/GoasdoueRousset-TR-DL-2000.ps

[7] Alon Levy. Answering Queries Using Views: a Survey. *To appear.*

[8] Marie-Christine Rousset. Backward Reasoning in Aboxes for Query Answering. In *Proceedings of DL'99 and KRDB'99.*