AI Communications 19 (2006) 229–237
IOS Press

Rewriting queries using views with negation

Foto Afrati and Vassia Pavlaki
Department of Electrical and Computer Engineering, NTUA, Greece
E-mail: {afrati,vpavlaki}@softlab.ece.ntua.gr

Abstract. Data integration and query reformulation are classical examples of problems that require techniques developed in both AI and database systems fields. In this work we address the problem of rewriting queries using views which has many applications. In particular, we consider queries and views that are conjunctive queries with safe negation (CQNs). We prove that given a CQN query and a set of CQN views, finding equivalent rewritings is decidable in both cases where the rewriting is in the language of CQNs or unions of CQNs. We prove decidability by giving upper bounds on the number of subgoals of the rewritings. We limit the search space of potential equivalent CQN rewritings and Maximally Contained CQN Rewritings (MCRs) to the language of unions of CQs in the case where the query is CQ. Finally, we give a sound and complete algorithm for finding equivalent rewritings in the language of unions of CQNs and we prove that if we consider the rewritings without negated subgoals, then they compute only certain answers under the OWA. Of independent interest is a simple test for checking containment between two unions of CQNs.

Keywords: Query rewritings, negation, data integration

1. Introduction

Data integration and query reformulation are classical examples of problems that require techniques developed in both AI and database systems fields (see [8, 9] and references therein). In this work we address the problem of *rewriting queries using views* which has received significant attention because of its applications in data integration, query reformulation, maintenance of physical data independence, global information systems, mobile computing. In data integration views describe a set of autonomous heterogenous data sources so we usually search for *maximally-contained rewritings* (MCRs) to get the best answer, given the available sources. If it is possible to find a solution which is equivalent (instead of contained) to the original query then we search for *equivalent rewritings*.

For Conjunctive Queries (CQs) and views, there exist algorithms which find equivalent rewritings or MCRs. In the case of *conjunctive queries with safe negation* (CQNs) the problem becomes more complex. For CQs, if an equivalent rewriting exists in the language of union of CQs, then there is one which is single CQ. The following example shows that in the presence of negation this does not hold even when only views have negated subgoals [17].

Example 1.1. Consider the following CQ query Q and CQN views V_1, V_2:
$Q:\ q(X,Y)$:- $a(X,Y)$
V_1: $v_1(X,Y)$:- $a(X,Y) \wedge b(Y)$
V_2: $v_2(X,Y)$:- $a(X,Y) \wedge \neg b(Y)$.
Then, an equivalent rewriting of Q using V_1, V_2 is the following union:
$q(X,Y)$:- $v_1(X,Y)$
$q(X,Y)$:- $v_2(X,Y)$.
Note that there is no equivalent rewriting of Q using V_1, V_2 which is a single CQN.

Another complexity that arises is that the expansion (see Definition 2.2) of a rewriting may also be in the language of union of CQNs.

Example 1.2. Consider the rewriting R and views V_1, V_2.
$R(X,Y,Z)$:- $\neg v_1(X,Y,Z) \wedge v_2(X,Y,Z)$
$V_1:\ v_1(X,Y,Z)$:- $a(X,Y) \wedge \neg a(Y,Z)$
$V_2:\ v_2(X,Y,Z)$:- $b(X,Y) \wedge b(Y,Z)$.
Then, the expansion of the rewriting is as follows:
$R^{exp}(X,Y,Z)$:- $\neg(a(X,Y) \wedge \neg a(Y,Z)) \wedge (b(X,Y) \wedge b(Y,Z))$. That is,
$R^{exp}(X,Y,Z)$:- $(\neg a(X,Y) \wedge b(X,Y) \wedge b(Y,Z)) \vee (a(Y,Z) \wedge b(X,Y) \wedge b(Y,Z))$.
Writing the disjunction as union, then R is in the language of union of CQNs:
$R^{exp}(X,Y,Z)$:- $\neg a(X,Y) \wedge b(X,Y) \wedge b(Y,Z)$
$R^{exp}(X,Y,Z)$:- $a(Y,Z) \wedge b(X,Y) \wedge b(Y,Z)$.

When evaluating a query we search for *certain answers*, i.e., tuples that are in the answer for *any* database. The definition of certain answers depends on the views assumptions we make: *closed world* versus *open world*. Given a view instance I, under the *closed world assumption* (CWA), I stores *all* the tuples that satisfy the view definitions in V, i.e., $I = V(D)$ whereas under the *open world assumption* (OWA), instance I is possibly incomplete and might only store *some* of the tuples that satisfy the view definitions in $\mathcal{V}$, i.e., $I \subseteq V(D)$. In data integration we usually take the OWA whereas in query optimization the CWA. In this work we give results for both assumptions. The next paragraph summarizes our contributions.

In Section 3 we show why existing algorithms for CQs can not be extended in a straightforward way to deal with CQNs. We prove that given a CQN query and a set of CQN views finding equivalent rewritings is decidable in both cases where the rewriting is in the language of CQNs or unions of CQNs. We prove decidability by giving upper bounds on the number of subgoals of the rewritings (Section 4). Moreover we show that given a CQ query and a set of CQN views, then if rewritings can be found in the language of unions of CQNs then rewritings can be found in the language of unions of CQs. So we prune the search space for potential rewritings (Section 5). We give a sound and complete algorithm that, given a CQN query and a set of CQN views finds equivalent rewritings in the language of unions of CQNs (Section 6). If we consider only the rewritings without negated subgoals, returned by the algorithm, then they compute only certain answers under the OWA (Section 7). Of independent interest is a simple test for checking containment between two unions of CQNs (Subsection 2.1.1).

1.1. Related work

There is a plethora of work addressing the *query rewriting problem* (see [7] for a survey). The problem has been shown to be NP-complete even when the queries describing the sources and the user query are CQs and do not contain interpreted predicates [10]. Several algorithms have been developed for finding rewritings of queries using views. The bucket [6], the inverse-rule [1,4,15], the MiniCon [14], and the Shared-Variable-Bucket [12] are some of them. An algorithm for finding equivalent rewritings with the smallest number of subgoals is given in [2]. The work in [11] considers the problem of answering CQs using infinite sets of views. The work in [1] searches for MCRs for a special case of Datalog queries and views that are unions of CQs. An algorithm for answering queries using materialized views in the presence of negation is given in [5]. It gives a good approximation and it is optimal for a significant class of queries.

2. Preliminaries

In this section we review the problem of rewriting queries using views. A *database schema* S is a set of relational names with given arity. A *database instance* over a given schema is a set of relations. A *query* is a mapping from databases to relations. A *view* is a predefined query. A query Q_1 *is contained* in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$, if for any database instance D, the answer computed by Q_1 is a subset of the answer computed by Q_2, i.e., $Q_1(D) \subseteq Q_2(D)$. The two queries are *equivalent*, denoted $Q_1 \equiv Q_2$, if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$. Given two CQs Q_1 and Q_2 the following holds; $Q_1 \sqsubseteq Q_2$ iff there is a *containment mapping* from Q_2 to Q_1, such that the mapping maps a constant to the same constant and maps a variable to either a variable or a constant [3]. Under this mapping, the head of Q_2 becomes the head of Q_1, and each subgoal of Q_2 becomes some subgoal in Q_1.

2.1. Conjunctive queries with negation (CQNs)

The general form of a conjunctive query with safe negated subgoals (CQN) is:

$$H :\text{-}\ G_1 \wedge \ldots \wedge G_n \wedge \neg F_1 \wedge \ldots \wedge \neg F_m$$

where H is $h(\overline{X})$ and G_i, F_i are of the form $g_i(\overline{X})$, $f_i(\overline{X})$. G's are called *positive* subgoals while F's are called *negative* subgoals. The variables in $\overline{X}$ are called *head* or *distinguished* variables, while the variables in $\overline{X}_i$ are called *body* variables. The body variables that are not distinguished variables are called *nondistinguished* variables. An *atom* is an expression of the form $a(t_1, \ldots, t_n)$, where a is a predicate of arity n and $t_1, \ldots, t_n$ are constants or variables. A *literal* is either a positive or negative atom. A subgoal is an atom or literal. A CQN is *safe* if each variable appearing in a negated atom appears also in a positive atom. In the sequel of the paper when we say *containment mapping* or simply *mapping* we refer to the corresponding CQs.

2.1.1. Containment test for unions of CQNs

Checking containment between two CQNs is complete for the class Π_2^p [16]. A test for checking containment between two CQNs was proposed in [16] and a more efficient algorithm for checking CQNs and unions of CQNs is given in [17]. In this work we do not address efficiency issues. We give a simple algorithm for checking containment between two *unions of CQNs*.

Theorem 2.1. *Let Q_1 and Q_2 be two queries each being a union of CQNs. Then $Q_1 \sqsubseteq Q_2$ iff Procedure given in* Fig. 1 *outputs "yes".*

Proof. ("If") Let D be any database and let t be a tuple which is in the answers of query Q_1, i.e., t is in $Q_1(D)$. Then, there is a CQN query Q_1^i in Q_1 such that t is in $Q_1^i(D)$. Hence, there is a homomorphism h from the positive subgoals in the body of Q_1^i to D that maps the head of Q_1^i on t. The part of D induced by the constants that are targets of h is isomorphic to a canonical database D_j that make the body of Q_1^i true. So t is in $Q_2(D)$ and therefore $Q_1 \sqsubseteq Q_2$.

("Only if") Suppose $Q_1 \sqsubseteq Q_2$. Then every database D_j that makes the body of some Q_1^i true is such that the frozen head of Q_1^i is in the answers of Q_1. Since $Q_1 \sqsubseteq Q_2$, the frozen head of Q_1^i is in the answers of Q_2 too. Hence, the test is successful. □

2.2. Query answering

Consider a database instance D, a view definition V and a view instance I. Given a query Q we want to compute $Q(D)$ but we only have access to I, so we try to get the best possible estimate of $Q(D)$ given I. Definition 2.1 formalizes the concept of certain answers under CWA and OWA.

Definition 2.1. Let V be a view definition, I be an instance of the view and Q a query. A tuple t is a *certain answer under OWA* if t is an element of $Q(D)$ for each database D with $I \subseteq V(D)$. A tuple t is a *certain answer under CWA* if t is an element of $Q(D)$ for each database D with $I = V(D)$.

Containment of Union-CQNs (check $Q_1 \sqsubseteq Q_2$)
Input: Two union-CQNs $Q_1 = \cup Q_1^i$, $Q_2 = \cup Q_2^j$
Output: Boolean
Method:
(1) For every CQN query Q_1^i in Q_1 do:

1. Construct the set of basic canonical databases that correspond to all the partitions of the set of variables of Q_1^i.
   (a) For each partition, assign a unique constant to each block of the partition.
   (b) Create the basic canonical database DB by replacing each variable in the body of Q_1^i by the constant of its block and treat the resulting subgoals as the only tuples in the database (i.e., *freeze* the body of Q_1^i). The basic canonical DB is the set of resulting *positive* subgoals.
2. For each basic canonical DB D constructed in step (1), check: if $Q_1^i(D)$ contains the frozen head of Q_1^i, then so does $Q_2(D)$.
3. If $Q_1^i(D)$ contains the frozen head of Q_1^i, we must also consider the larger set of (*extended*) canonical DB's D' formed by adding to D other tuples that are formed from the same symbols as D, but not any of the tuples that are the negated subgoals of Q_1^i. Check that $Q_2(D')$ contains the frozen head of Q_1^i (with respect to D).
4. If so for all D', then $Q_1^i \sqsubseteq Q_2$; if not, then not.

(2) If step (1) holds for all Q_1^i in Q_1 output "yes".

Fig. 1. Containment test for unions of CQNs.

2.2.1. Rewriting CQN queries using CQN views

We start with Definition 2.2 which formalizes the concept of *expansion* of a rewriting.

Definition 2.2. Let Q be a CQN query, V a set of CQN views and R an equivalent rewriting of Q using V. The *expansion* of R denoted R^{exp} is obtained as follows: we replace each view atom of R with the body of the corresponding view definition. Nondistinguished variables are replaced by fresh nondistinguished variables. Thus, the body of R has been turned into logical expression which is not anymore a conjunction of literals (Example 1.2). Then, we turn this into disjunctive normal form. R^{exp} is the union of CQNs, one CQN for each disjunct: the head is the head of R and the body is the corresponding disjunct. In all results in this paper, the expansion of a rewriting is assumed to be a safe query too.

Given a query Q, a set of views V and a rewriting language L (e.g. unions of conjunctive queries), a query R is a *contained rewriting* of Q using V w.r.t. L, if R is in L, R uses only the views in V, and for the expansion R^{exp} of R it holds $R^{exp} \sqsubseteq Q$. We call R an *equivalent rewriting* of Q using V w.r.t. L, if R is in L, R uses only the views in V, and $R^{exp} \equiv Q$. We call R a *maximally contained rewriting (MCR)* of Q using V w.r.t. L if (1) R is a contained rewriting of Q using V (w.r.t. L), and (2) there is no contained rewriting R_1 of Q using V (w.r.t. L) such that R_1 properly contains R. We assume also rewritings to be safe queries.

Definition 2.3. Let Q be a CQN query and V a set of CQN views. A rewriting R of Q using V is equivalent iff for every database instance D it holds: $Q(D) = R(V(D))$.

Proposition 2.1. *Let Q be a CQN query and R a rewriting of Q using CQN views V. Then, R is an equivalent rewriting of Q using V iff the expansion of R is equivalent to Q.*

Proof. The proof for both directions follows easily from the fact that $R^{exp}(D) = R(V(D))$ for any database D. W.l.o.g., suppose that R is a single CQN. We will prove here only the direction $R^{exp}(D) \subseteq R(V(D))$. Let t be a tuple in $R^{exp}(D)$. R^{exp} is a union of CQNs. Then there is an R_i^{exp} among those CQNs such that t is in $R_i^{exp}(D)$. Hence there is a homomorphism h from the subgoals of R_i^{exp} to D which computes t. For the positive subgoals of R, it is straightforward to see that h can be used to compute a corresponding view tuple in $V(D)$. For any negative subgoal v_k in R, and since rewriting and expansion are safe queries, h can be used to prove that one of the subgoals in the view definition corresponding to v_k is not satisfied, hence the corresponding view tuple is not in $V(D)$. Thus, we conclude that all subgoals of R are satisfied in $V(D)$ hence t is also in $R(V(D))$. □

Example 1.1 showed that given a CQ query and CQN views, then (a) the rewriting might not have negated subgoals and (b) there is no equivalent rewriting which is single CQN. Example 2.1 shows that this holds even if the query has negated subgoals. In addition it shows how we can use Proposition 2.1 and the containment test for CQNs to prove that a rewriting is equivalent to query.

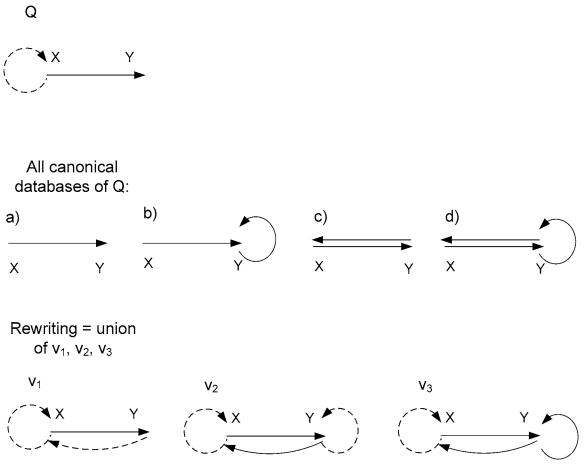

Fig. 2. Example 2.1.

Example 2.1. Suppose we are given the following query Q and views V_1, V_2, V_3:
$Q: \ q()$:- $a(X,Y) \wedge \neg a(X,X)$
V_1: $v_1(X,Y)$:- $a(X,Y) \wedge \neg a(Y,X) \wedge \neg a(X,X)$
V_2: $v_2(X,Y)$:- $a(X,Y) \wedge a(Y,X) \wedge \neg a(X,X) \wedge \neg a(Y,Y)$
V_3: $v_3(X,Y)$:- $a(X,Y) \wedge a(Y,X) \wedge \neg a(X,X) \wedge a(Y,Y)$
An equivalent rewriting is the following union:
$q()$:- $v_1(X,Y)$, $q()$:- $v_2(X,Y)$, $q()$:- $v_3(X,Y)$.
Note that there is no rewriting in the form of a single CQN. Figure 2 depicts the four canonical databases (a) (b) (c) (d) of Q. Databases (a), (b) are covered by view V_1, (c) by V_2 and (d) by V_3.

The claim of Proposition 2.2 was observed in [17].

Proposition 2.2. *There is a CQN query Q and a set of CQN views V such that it holds; there is an equivalent rewriting R of Q using V in the language of union of CQNs and there is no rewriting R' of Q using V in the language of CQNs.*

Proof. Consider the Example 2.1. The rewriting given in the example is an equivalent rewriting because, following the containment test for CQNs and Proposition 2.1 we consider all canonical databases of Q. Then V_1 covers (a) and (b), V_2 covers (c) and V_3 covers (d) in Fig. 2. To see that there is no CQN equivalent rewriting, observe that any CQN rewriting R' which contains one of the views as positive subgoal (with no variables in the head) is strictly contained in the query (i.e., the query is not contained in the rewriting). Hence, if we construct a CQN rewriting adding more subgoals to R', this will be contained in R', therefore it will not contain the query either. □

3. Technical challenges

In this section we show why existing algorithms dealing with CQs can not be extended in a straightforward way to incorporate also CQNs. The reason is that algorithms for CQs find *locally minimal* or *containment minimal* rewritings. Consider a rewriting R of a CQ query Q using CQ views V. The crucial thing in what follows is to distinguish between *equivalence as rewritings* and *equivalence as queries*. Let R_m be the result after dropping subgoals from R up to the point where equivalence *as queries* between R and R_m is retained. We say "as queries" because we do not consider the expansions in this case. Concerning equivalence to Q now, we are interested in the expansion of the rewriting. So we might still be able to remove some view subgoals from R_m (after the expansion now), while retaining its equivalence to Q. That is although we can not drop any subgoal from R_m without harming equivalence between R_m and R *as queries* it is possible that we can drop subgoals from the expansion of R_m without harming equivalence *as rewritings* between R_m and Q. In general, two rewritings may not be equivalent as queries, although they both compute the same answer to the query i.e., they are equivalent as rewritings. A rewriting R of Q using V is said to be *locally minimal* (LMR in short) if we can not remove any literals and still retain equivalence to Q. We say that a LMR is a *containment-minimal* rewriting if there is no other LMR that is properly contained in this rewriting as queries.

Example 3.1. Suppose we are given the following query Q and set of views $V = \{V_1, V_2, V_3\}$:
$Q : q()$:-$a(X_1, X_2) \wedge a(X_2, X_3) \wedge a(X_3, X_4) \wedge a(X_4, X_5) \wedge a(X_5, X_6) \wedge a(X_6, X_7) \wedge a(X_7, X_1) \wedge \neg a(X_2, X_2) \wedge \neg a(X_5, X_5)$
$V_1 : v_1(X_1, X_4)$:- $a(X_1, X_2) \wedge a(X_2, X_3) \wedge a(X_3, X_4) \wedge a(X_4, X_5) \wedge a(X_5, X_6) \wedge a(X_6, X_7) \wedge a(X_7, X_1) \wedge \neg a(X_3, X_3)$
$V_2 : v_2(X_3, X_5)$:- $a(X_1, X_2) \wedge a(X_2, X_3) \wedge a(X_3, X_4) \wedge a(X_4, X_5) \wedge a(X_5, X_6) \wedge a(X_6, X_7) \wedge a(X_7, X_1) \wedge \neg a(X_4, X_4)$
$V_3 : v_3(X, Y)$:- $a(X, X_2) \wedge a(X_2, Y)$.
An equivalent rewriting is: $R : r()$:- $v_1(X, Y) \wedge v_2(Z, X) \wedge v_3(Y, Z)$. Figure 3(a) depicts the body of Q and Fig. 3(b) illustrates the R^{exp}. The two heptagons corresponding to the (expansions of the) atoms $v_1(X, Y)$ and $v_2(Z, X)$ are those with a common vertex labelled X. The path formed by the arcs $Y \rightarrow X_2''$ and $X_2'' \rightarrow Z$ corresponds to the expansion of

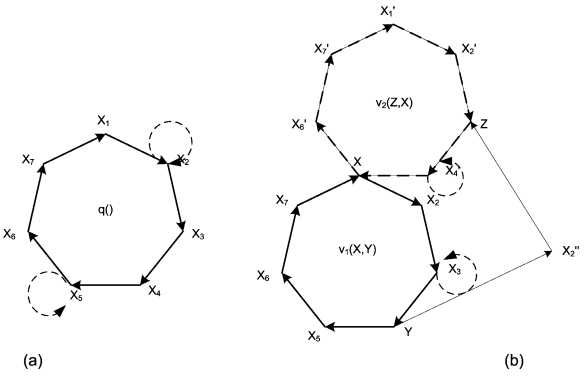

Fig. 3. Example 3.1.

$v_3(Y, Z)$. It is easy to see that $Q \sqsubseteq R$ since there is a mapping from Fig. 3(b) to Fig. 3(a). To check that $R \sqsubseteq Q$, note that there is a mapping from Q to the heptagon of Fig. 3(b) with vertices $Z, X_4', X, X_2, X_3, Y, X_2''$. In particular, vertex X_1 maps to Z, X_2 to X_4', X_3 to X, X_4 to X_2, X_5 to X_3, X_6 to Y, X_7 to X_2''.

If we had used one of the existing algorithms to find an equivalent rewriting of a CQ query using CQ views and then add to it some negated subgoals, we would not be able to find an equivalent rewriting of the original CQN query using CQN views. We explain this point. Consider the query Q' and views V_1', V_2', V_3' which are defined as Q, V_1, V_2, V_3 but with the negated subgoals dropped. An equivalent rewriting of Q' using V', computed by the CoreCover algorithm [2] for instance, would be $R' : r()$:- $v_1'(X, Y)$. Adding the negated subgoals of Q to R' does not give us equivalent rewriting of Q using V. Instead, we need the rewriting $R'' : r()$:- $v_1'(X, Y) \wedge v_2'(Z, X) \wedge v_3'(Y, Z)$. However R'' would have not been computed by existing algorithms dealing with CQs because they find LMRs. R'' in our example contains the subgoals $v_2'(Z, X)$ and $v_3'(Y, Z)$ which if they were dropped in the CQ case they would not harm equivalence. However these view subgoals are needed to compute equivalent rewritings in the CQN case.

Example 3.2. Consider the following query Q and views V_1, V_2:
$Q : q(X, Y)$:-$a(X, Z_1) \wedge a(Z_1, 2) \wedge b(2, Z_2) \wedge b(Z_2, Y) \wedge \neg a(Z_1, Z_1) \wedge \neg a(Z_2, Z_2)$
$V_1 : v_1(X, Y)$:-$a(X, Z_1) \wedge a(Z_1, 2) \wedge b(2, Z_2) \wedge b(Z_2, Y) \wedge \neg a(Z_1, Z_1)$
$V_2 : v_2(X, Y)$:-$a(X, Z_1) \wedge a(Z_1, 2) \wedge b(2, Z_2) \wedge b(Z_2, Y) \wedge \neg a(Z_2, Z_2)$.
An equivalent rewriting is R:-$r(X, Y)$:-$v_1(X, Y') \wedge v_2(X', Y)$. Consider Q', V_1', V_2' defined as Q, V_1, V_2 but with the negated subgoals dropped. In this case the

rewriting of Q' using V' is a LMR. Still, it is not a containment minimal rewriting, i.e., there is another LMR of Q' using V' which is the following: R': $r'(X,Y)$:- $v_1(X,Y)$. Some existing algorithms dealing with CQs compute containment minimal rewritings (see for instance [2]). So adding the negated subgoals of Q to R' would not result to an equivalent rewriting of Q using V.

4. Decidability for equivalent rewritings

In this section we prove that finding an equivalent rewriting of a CQN query using CQN views is decidable for the case where the rewriting is a single CQN (Theorem 4.1) or union of CQNs (Theorem 4.2). We prove decidability by giving upper bounds on the number of subgoals of rewritings.

Theorem 4.1. *Let Q be a CQN query and V a set of CQN views. It is decidable whether there is an equivalent rewriting R for the query Q using the views V in the language of CQNs. If such a rewriting exists, then there is an algorithm which finds it in 7-exponential time.*

Proof. Suppose there is an equivalent rewriting R which is a single CQN. Suppose Q has n variables. Consider all possible canonical databases of Q (2^{2^n}) that compute the head of Q. For every one of them there must be a containment mapping from R^{exp}, the expansion of R, to Q that satisfies the negated subgoals of R^{exp}. Associate with each variable of R^{exp} a list of the n variables that each of these mappings sends the variable of R^{exp} to. We define two variables of R^{exp} to be "equivalent" if their lists are the same. Since lists are of length 2^{2^n} and each entry on the list has one of n values, there are at most $n^{2^{2^n}}$ equivalence classes. Design a new solution R' that equates all equivalent variables. R' is contained in R after expansion, since all we did was equating variables, thus restricting R and R^{exp}. However, R'^{exp}, the expansion of R', has containment mappings to Q for all canonical databases. The reason is that we equated variables that always went to the same variable of Q for each canonical database. Thus Q is contained in R'. Since Q contains R^{exp}, which contains R'^{exp}, it is also true that R'^{exp} is contained in Q. Thus, R' is another equivalent rewriting of Q. There is a quadruple exponential bound on the size of R, since there are only $n^{2^{2^n}}$ variables and a linear number of predicates each with linear arity. So we need to look only at some quadruply exponentially sized solutions. The search space is 5-exponential and the upper bound is 7-exponential. □

Theorem 4.2. *Let Q be a CQN query Q and V a set of CQN views. It is decidable whether there is a CQN equivalent rewriting R for the query Q using the views V in the language of finite unions of CQNs. If such an equivalent rewriting exists, then there is an algorithm which finds it in octuply exponential time.*

Proof. Let $R = \cup R_i$ (union of CQNs) be an equivalent rewriting of Q using V. As in the proof of Theorem 4.1 we consider all canonical databases of Q (2^{2^n}). Now, for every canonical database, there must be a containment mapping from the expansion of one R_i to Q that satisfies the negated subgoals of R_i^{exp}. Then, we argue as in the proof of Theorem 4.1. We need to look only at quadruply exponentially sized solutions for each R_i. So there are 5-exponentially many combinations and we need to look at all of them. The search space is 6-exponential and the upper bound for finding equivalent rewritings is octuply-exponential. □

5. Rewritings of CQ query using CQN views

In this section we study the case where the query is CQ but the views are CQNs. We prove that if rewritings can be found in the language of unions of CQNs then rewritings can be found in the language of unions of CQs. So we prune the search space for potential equivalent rewritings and MCRs.

Proposition 5.1. *Let Q be a CQ query and V a set of CQN views. If there is a contained rewriting R of Q using V in the language of union of CQNs, then there is a contained rewriting R' of Q using V in the language of union of CQs such that $R \sqsubseteq R'$.*

Proof. Let $R = \cup R_i$ be a rewriting in the language of union of CQNs. If we drop the negative view subgoals from R_i obtaining R_i', then R_i is contained in R_i'. We have to prove that R_i' is contained in Q. We take the expansion R_i^{exp} which is a union of CQNs. If there is a negative subgoal v_j in R_i then (because all queries are safe) there are CQNs P_1 and P_2 in R_i^{exp} such that P_1 and P_2 have each a number of k subgoals of which $k-1$ are the same in both and the k-th subgoal in P_1 is a negated atom (which is positive in the body of v_j) and the k-th subgoal in P_2 is a positive atom (which

is a negative atom in the body of v_j). Since Q homomorphically maps on every canonical database of both P_1 and P_2 and Q has no negative atoms to be satisfied, then Q homomorphically maps on every canonical database of the common $k-1$ subgoals of P_1 and P_2. Thus, if we drop v_j from R_i producing R_{ij} then R_{ij} is contained in Q. Dropping one by one all negative subgoals of R_i we produce R_i' which is contained in Q. □

Proposition 5.2. *Let Q be a CQ query and V a set of CQN views. If there is an equivalent rewriting of Q using V in the language of union of CQNs, then there is an equivalent rewriting of Q using V in the language of union of CQs.*

Proof. Suppose R is an equivalent rewriting of Q using V in the language of union of CQNs. Since R is also a contained rewriting, from Proposition 5.1 there exists a contained rewriting R' of Q using V in the language of union of CQs such that $R \sqsubseteq R'$. Since $Q \sqsubseteq R$, then $Q \sqsubseteq R'$. □

Proposition 5.3. *Let Q be a CQ query and V a set of CQN views. If there is an MCR of Q using V in the language of union of CQNs, there is an MCR of Q using V in the language of union of CQs.*

Proof. Suppose R is an MCR in the language of unions of CQNs. From Proposition 5.1 there exists a contained rewriting R' with $R \sqsubseteq R'$. So R' is also an MCR in the language of unions of CQs. □

6. The algorithm

In this section we give a sound and complete algorithm which given a CQN query and set of views finds equivalent rewritings in the language of unions of CQNs. Note that our algorithm can be seen as a chase algorithm adjusted to the containment test for CQNs [13]. We treat in separate the positive and negative subgoals of views. What is gained w.r.t. a brute-force algorithm is that we prune the search space of positive view subgoals.

We start with considering all canonical databases of Q and we keep only those that compute the head of the query, or if the query is boolean, that make the body of the query true. Let $D_1, \ldots, D_k$ be the canonical databases we keep. We then check which views will be potentially used by equivalent rewritings. We keep only the views from which there is a mapping to some D_i. We call these views *useful views* and we use V' to denote the set. Note that the concept of a *useful* view was also used in [17] defining a superset of the set of views defined here.

Definition 6.1. Given a CQN query Q and a set of CQN views V, we call *useful views* the set V' defined as the set V after removing the views from which there is no mapping h to any canonical database of Q such that h computes at least one tuple of the view.

Example 6.1. (Continued from Example 3.2). Some views which would not be useful are:
V_3 : $v_3(X, Y)$:- $a(X, X)$
V_4 : $v_4(X, Y)$:- $c(Y, Z_1)$.

For every $D_i, i = 1, \ldots, k$ we compute the view tuples $T_i(V')$ by applying the useful views V' on D_i and restoring back the variables in the tuples. $T(V')$ denotes the set of view tuples after all useful views have been applied to the databases. We call these view tuples *useful* positive view tuples.

Example 6.2. (Continued from Example 3.2). The view tuples $v_1(z_1, z_2)$, $v_2(z_1, z_2)$ are not useful.

The algorithm proceeds with a complete enumeration. We consider all possible combinations of view tuples from $T(V')$ with all negative view subgoals from V to create rewritings. Finally, we check every one of them if it is an equivalent rewriting. Figure 4 summarizes the steps of the algorithm. Note that we consider the view tuples in their *most relaxed forms* i.e., in each relaxed form of the particular view tuple t we rename a proper subset of its variables to fresh (not used in other view subgoals of the rewriting) variables and we do that for all possible proper subsets of variables in t.

Example 6.3. (Continued from Example 3.2). View tuples being in their most relaxed form means the following. All variables of the rewritings are renamed as to have all different names. For this example we would have a rewriting R_0:-$r(X, Y)$:-$v_1(X', Y') \wedge v_2(X'', Y'')$. We make all possible equations of the variables and for each one of them we test if it is a contained rewriting. In Example 3.2 we showed that an equivalent rewriting is R:-$r(X, Y)$:-$v_1(X, Y') \wedge v_2(X'', Y)$ which derived from R_0 after equating X' to X and Y'' to Y.

Procedure Finding Equivalent Rewritings:
Input: A CQN query Q and set of CQN views V.
Output: (a) contained rewritings of Q using V and (b) equivalent rewritings of Q using V.
Method:

(1) $\mathcal{R} := \emptyset$.
(2) Construct all canonical databases for Q.
(3) Keep only those canonical databases which compute the head of Q.
(4) Find the set of useful views V'.
(5) For every canonical database D_i do:
   (a) Compute the view tuples $T_i(V')$ by applying the useful view definitions V' on D_i.
   (b) If for a canonical database D_i it holds $T_i(V') := \emptyset$ then stop (as there is no rewriting).
   (c) Keep the set of useful positive view tuples $T(V') = \cup T_i(V')$.
(6) Consider all combinations of most relaxed forms of view tuples from $T(V')$ with negative subgoals from V to create rewritings R_i.
(7) For every R_i check if it is a contained rewriting of Q using V. If it is, then add R_i to $\mathcal{R}$.
(8) Output the set $\mathcal{R}$.
(9) Check whether the union of all R_i in $\mathcal{R}$ is an equivalent rewriting. If it is, output the rewriting. Otherwise, output "fail".

Fig. 4. The algorithm.

Lemmata 6.1, 6.2 prove that by keeping only useful views and useful positive view tuples we do not lose solutions. Finally, Theorem 6.1 proves that our algorithm is sound and complete.

Lemma 6.1. *Let Q be a CQN query, V a set of CQN views and V' the set of CQN useful views. There is an equivalent rewriting of Q using V in the language of unions of CQNs iff there is an equivalent rewriting in the language of unions of CQNs of Q using V' for positive subgoals and V for negative subgoals.*

Proof. One direction is straightforward since $V' \subseteq V$. Let R be an equivalent rewriting of Q using V and V_i be a view used in a positive subgoal. There is a mapping from R^{exp} to every canonical database of Q, hence there is a mapping from the view definition of V_i to a canonical database of Q, therefore V_i is in V'. □

Lemma 6.2. *Let Q be a CQN query, V a set of CQN views, V' the set of useful views in V and $T(V')$ the set of useful positive view tuples. There is an equivalent rewriting in the language of unions of CQNs of Q using V iff there is an equivalent rewriting in the language of unions of CQNs which uses only view tuples from $T(V')$ for positive subgoals.*

Proof. One direction is straightforward since $T(V')$ is a subset of the view tuples that we would potentially use. Let R be an equivalent rewriting of Q using V and let v be a view tuple used in a positive subgoal in R. There is a mapping from R^{exp} to every canonical database of Q. Hence there is a mapping from v to some canonical database of Q, consequently v is in $T(V')$. □

Theorem 6.1. *Given a CQN query Q and set of CQN views V, then the procedure "Finding Equivalent Rewritings" of* Fig. 4 *finds an equivalent rewriting (if there exists one) of Q using V in the language of unions of CQNs.*

Proof. Completeness: from Lemmata 6.1, 6.2.
Soundness: consequence of step 6 in Fig. 4. □

7. Certain answers under OWA

In general, finding MCRs is more difficult than finding equivalent rewritings. Example 7.1 shows that we may need a language more expressive than that of the query and views to have an MCR. In particular, we need to move to Datalog.

Example 7.1. Consider Q and V_1, V_2, V_3 as follows:
$Q() :- e(X,Z) \wedge e(Z,Y) \wedge b(X,X) \wedge \neg b(Y,Y)$
$V_1(X,Y) :- e(X,Z) \wedge e(Z,Y) \wedge b(Z,Z)$
$V_2(X,Y) :- e(X,Z) \wedge e(Z,Y) \wedge \neg b(Z,Z)$
$V_3(X,Y) :- e(X,Z_1) \wedge e(Z_1,Z_2) \wedge e(Z_2,Z_3) \wedge e(Z_3,Y)$.
For any $k > 0$, R_k is a contained rewriting: $R_k :- v_1(X,Z_1) \wedge v_3(Z_1,Z_2), \ldots, v_3(Z_{k-1},Z_k) \wedge v_2(Z_k,Y)$.
In fact, the following *recursive* Datalog program is a contained rewriting of the query:
$Q() :- v_1(X,W) \wedge T(W,Z) \wedge v_2(Z,Y)$
$T(W,W) :-$
$T(W,Z) :- T(W,U) \wedge v_3(U,Z)$.

If from the rewritings returned by the algorithm presented in Fig. 4 we consider those rewritings that do not contain negated subgoals, then these rewritings compute only certain answers under the OWA. Theorem 7.1 proves this claim.

Theorem 7.1. *Let Q be a CQN query, V a set of CQN views and R a contained rewriting of Q using V where R does not have negated subgoals. Then all answers computed by R on a given view instance I are certain answers under the OWA.*

Proof. Let D be any database such that $I \subseteq V(D)$. Let t_i be any view V_j tuple in I. Then there is an evaluation h_{ij} of t_i on D according to V_j. Suppose R computes tuple t in I. Then there is a mapping h from the body of R to I which computes t. Suppose h maps one V_j subgoal of R to tuple t_i in I. Since $V_j(t_i)$ exists in I there is an evaluation h_{ij} of t_i on D according to V_j. Combining h with the evaluations h_{ij} for each view tuple in R, results in an evaluation of t when applying the expansion of the rewriting R^{exp} to D. Since $R^{exp} \sqsubseteq Q$, t is an answer to the query for any database D such that $I \subseteq V(D)$. Hence t is a certain answer. □

8. Conclusions

In this work we addressed the problem of rewriting CQN queries using CQN views with safe negation. In [17] it is stated explicitly that the results hold only for safe negation whereas the containment test in [16] (which we use here) deals also with unsafe negation. Many of the results in this paper may be extended to deal with unsafe negation. The definition of certain answers depends on the views assumptions we make: closed world vs open world. This work contains useful results for both assumptions.

Acknowledgements

We would like to thank the reviewers for their useful and insightful comments.

Work supported by HERAKLEITOS EPEAEK programme, EU and Greek Ministry of Education, cofunded by the European Social Fund (75%) and National Resources (25%).

References

[1] F. Afrati, M. Gergatsoulis and T. Kavalieros, Answering queries using materialized views with disjunctions, in: *ICDT*, 1999, pp. 435–452.

[2] F. Afrati, C. Li and J. D. Ullman, Generating efficient plans using views, in: *SIGMOD*, 2001, pp. 319–330.

[3] A.K. Chandra and P.M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: *STOC*, 1977, pp. 77–90.

[4] O.M. Duschka and M.R. Genesereth, Answering recursive queries using views, in: *PODS*, 1997.

[5] S. Flesca and S. Greco, Rewriting queries using views, *IEEE Trans. Knowl. Data Eng.* **13**(6) (2001), 980–995.

[6] G. Grahne and A.O. Mendelzon, Tableau techniques for querying information sources through global schemas, in: *ICDT*, 1999, pp. 332–347.

[7] A.Y. Halevy, Answering queries using views: A survey, *VLDB Journal* **10**(4) (2001), 270–294.

[8] C. Koch, Query rewriting with symmetric constraints, *AI Communications* **17**(2) (2004), 41–55.

[9] A. Levy, Combining artificial intelligence and databases for data integration, *Special issue of LNAI: Artificial Intelligence Today*, 1999.

[10] A. Levy, A.O. Mendelzon, Y. Sagiv and D. Srivastava, Answering queries using views, in: *PODS*, 1995.

[11] A. Levy, A. Rajaraman and J. Ullman, Answering queries using limited external query processors, *J. Comput. Syst. Sci.* **58**(1) (1999), 69–82.

[12] P. Mitra, An algorithm for answering queries efficiently using views, in: *ADC*, 2001, pp. 99–106.

[13] L. Popa, Object/Relational Query Optimization with Chase and Backchase, PhD thesis, University of Pennsylvania, 2000.

[14] R. Pottinger and A. Halevy, A scalable algorithm for answering queries using views, in: *VLDB*, Volume 10, 2001, pp. 182–198.

[15] X. Qian, Query folding, in: *ICDE*, 1996, pp. 48–55.

[16] J.D. Ullman, Information integration using logical views, in: *ICDT*, 1997, pp. 19–40.

[17] F. Wei and G. Lausen, Containment of conjunctive queries with safe negation, in: *ICDT*, 2003.