Answering Queries Determined by Views

Foto Afrati
Electrical and Computing Engineering
National Technical University of Athens
157 73 Athens, Greece
afrati@cs.ece.ntua.gr

July 16, 2006

Abstract

Answering queries using views is the problem which examines how to derive the answers to a query when we only have the answers to a set of views. In this paper we investigate this problem in the case where the answers to the views uniquely determine the answers to the query. We say that a view set $\mathcal{V}$ *determines* a query Q if for any two databases D_1, D_2 it holds: $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ implies $Q(D_1) = Q(D_2)$. We consider the case where query and views are defined by conjunctive queries. We ask the question: If a view set $\mathcal{V}$ determines a query Q, is there an equivalent rewriting of Q using $\mathcal{V}$? Clearly if we can find an equivalent rewriting then the complexity of answering the queries given a view instance is polynomial. In this paper we show that computing the answers to queries determined by views is in $NP \cap coNP$. We find cases, (such as chain queries, views without nondistinguished variables) where if a query is determined by a view set then there is an equivalent rewriting, hence the complexity of answering the query on a view instance is polynomial. We reduce the general problem to special cases (such as boolean queries, binary base predicates). We introduce a problem which is a special case of the general problem and relates determinacy to query equivalence.

1 Introduction

The problem of using materialized views to answer queries [LMSS95] has received considerable attention because of its relevance to many data-management applications, such as information integration [B$^+$97, C$^+$94, HKWY97, IFF$^+$99, LRO96, Ull97], data warehousing [TS97],[ACN00] web-site designs [FLSY99], and query optimization [CKPS95]. The problem can be stated as follows: given a query Q on a database schema and a set of views $\mathcal{V}$ over the same schema, can we answer the query using only the answers to the views, i.e., for any database D, can we find $Q(D)$ if we only know $\mathcal{V}(D)$?

A related fundamental question has recently arisen which is related to the information that is provided by a set of views for a specific query [SV05]. Thus, we say that a set $\mathcal{V}$ of views *determines* a query Q if for any two databases D_1, D_2 it holds: $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ implies $Q(D_1) = Q(D_2)$. A database query Q can be thought of as defining a partition of the set of all databases in the sense that databases on which the query produces the same set of tuples in the answer belong to the same equivalence class. In the same sense a set of views defines a partition of the set of all databases. Thus, if a view set $\mathcal{V}$ determines a query Q, then the views' partition is a refinement of the partition defined by the query. Thus, the equivalence class of $\mathcal{V}(D)$ uniquely determines the equivalence class of $Q(D)$. However, it is not easy to see whether the mapping from the views' equivalence class to the query equivalence class is even computable. We show in this paper that if a CQ view set determines a CQ query then computing the answers to the query is in $NP \cap coNP$.

A large amount of work in answering queries using views concerns finding rewritings of queries using a set of views. When there is an equivalent rewriting of a query Q using a set of views $\mathcal{V}$ then $\mathcal{V}$ determines Q. How about the converse? Given that $\mathcal{V}$ determines Q, can we say that there exists an equivalent rewriting of Q using $\mathcal{V}$? The existence of rewritings depend on the language of the rewriting and the language of the query and views. Given query languages $\mathcal{L}$, $\mathcal{L}_\mathcal{V}$, $\mathcal{L}_Q$ we say that a language $\mathcal{L}$ is *complete* for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewritings if whenever a set of views $\mathcal{V}$ in $\mathcal{L}_\mathcal{V}$ determines a query Q in $\mathcal{L}_Q$ then there is a rewriting of Q in $\mathcal{L}$ which uses only $\mathcal{V}$. In [SV05] this problem is investigated and is shown that there are cases where a certain language is not complete, e.g., it is shown that the language of union of conjunctive queries (UCQ) is not complete for UCQ-to-UCQ rewritings. However it is noticed that a hard case to settle and hence an open problem is whether the language of conjunctive queries (CQ) is complete for CQ-to-CQ rewritings. In this paper we answer this question positively in special cases and also we show that some special cases are as hard to resolve as the general problem by reducing the general problem to them. Finally we introduce a seemingly simpler problem that relates determinacy and query equivalence which also remains open, whereas here we solve a special case of it.

The organization and the contribution of the paper are as follows: In Section 3 we show that if the views determine the query then the query answering problem is in $NP \cap coNP$. In Section 4 we prove that there is an algorithm to find an equivalent rewriting in case there exists one. In particular, we show that if there is an equivalent rewriting then the *canonical rewriting* is such a rewriting. We show how to construct a canonical rewriting. In Section 5 we present three special cases of CQ-to-CQ rewritings for which CQ is complete. The special cases are: when views have no nondistinguished variables, when views and query are chain queries and when the query has a single variable and the view set contains a single view with one nondistinguished variable. Hence, in these special cases the query answering problem is in PTIME. A summary of all CQ cases for which we know that CQ is complete is in Table 1.

In Section 6 we ask the question: If a single view determines the query then are there some natural conditions to add so that the query and view are equivalent? We identify such conditions and show that if they hold for a view and query then the following is true: If CQ is complete for CQ-to-CQ rewritings, and the view determines the query, then the view and query are equivalent. Thus we have a new variant of the problem which, although it seems like an "easier" problem to solve, this also remains open; here we solve a special case of it. In Section 7 we reduce the original problem to a special case where, if an equivalent rewriting exists then it is a projection of a single view. In Section 8 we present more reductions of the general problem to special cases, such as when we use only binary base predicates and when the query is Boolean. We also include a reduction which addresses connectivity issues related to determinacy.

1.1 Related Work

In [SV05], the problem of determinacy is investigated for many languages including first order logic and fragments of second order logic and a considerable number of cases are resolved. The results closer to our setting show that if a language $\mathcal{L}$ is complete of UCQ-to-UCQ (i.e., unions of CQs) rewritings, then $\mathcal{L}$ must express non-monotonic queries. Moreover, this holds even if the database relations, views and query are restricted to be unary. This says that even Datalog is not complete for UCQ-to-UCQ rewritings. Datalog is not complete even for $\mathrm{CQ}^{\neq}$-to-CQ rewritings. For CQ query and views, it is shown in [SV05] that CQ is complete for CQ-to-CQ rewritings iff whenever a set of views $\mathcal{V}$ determines a query Q over finite instances then $\mathcal{V}$ determines Q over unrestricted (i.e., may be infinite too) instances. They also prove that for unary or Boolean CQ views, then CQ is complete for CQ-to-CQ rewritings. For the unrestricted case, they prove that CQ is complete for CQ-to-CQ rewritings, however no monotonic language is complete for UCQ-to-UCQ rewritings.

Determinacy and notions related to it are also investigated in [GT00] where the notion of subsumption is introduced and used to the definition of complete rewritings and in [CdGLV02] where the concept of lossless view with respect to a query is introduced and investigated both under the sound view assumption (a.k.a. open world assumption) and under the exact view assumption (a.k.a. closed world assumption) on regular path queries used for semi-structured data. Losslessness under the CWA is identical to determinacy. There is a large amount of work on equivalent rewritings of queries using views. It includes [LMSS95] where it is proven that it is NP-complete whether a given CQ query has an equivalent rewriting using a given set of CQ views, [CR97] where polynomial subcases were identified. In [RSU95], [ALM02], [DG97] cases were investigated for CQ queries and views with binding patterns, arithmetic comparisons and recursion, respectively. In some of these works also the problem of maximally contained rewritings is considered. Intuitively, maximally contained rewritings is the best we can do when there is no equivalent rewriting and want to obtain a query that uses only the views and computes as many certain answers [AD98] as possible. In [LBU01] the notion of p-containment and equipotence is introduced to characterize view sets that can answer the same set of queries. Answering queries using views in semi-structured databases is considered in [CdGLV02] and references therein.

2 Preliminaries

2.1 Basic Definitions

We consider queries and views defined by conjunctive queries (CQ for short) (i.e., select-project-join queries) in the form:

$$h(\bar{X}) : -g_1(\bar{X}_1), \ldots, g_k(\bar{X}_k).$$

Each subgoal in the *body* is a *relational atom*. In each subgoal $g_i(\bar{X}_i)$, predicate g_i defines a *base relation* (we use the same symbol for the predicate and the relation), and every argument in the subgoal is either a variable or a constant. A variable is called *distinguished* if it appears in the head. We shall use names beginning with lower-case letters for constants and relations, and names beginning with upper-case letters for variables.

Query	Views	Reference
any	without nondistinguished	this paper
any	unary	[SV05]
any	boolean	[SV05]
single variable	single view, binary, 1 nondist.	* this paper
chain	chain	* this paper

Table 1: Summary of polynomial cases: CQ is a complete language for rewritings for the listed subcases of CQ queries and views. The cases with asterisk (*) assume binary base predicates.

We use $V, V_1, \ldots, V_m$ to denote *views* that are defined by conjunctive queries on the base relations. We say that a CQ is *minimized* if there are not redundant subgoals, i.e, if we delete any subgoal then we obtain a query which is not equivalent to the original query. In the rest of this paper, we consider wlog minimized queries and views.

A *relational structure* is a set of atoms over a *domain* of variables and constants. A relational atom with constants in its arguments is called a *ground atom.* A database instance is a finite relational structure with only ground atoms. The body of a conjunctive query can be also viewed as a relational structure. A *homomorphism* is a mapping from the variables and constants of a relational structure S_1 to the variable and constants of another relational structure S_2 so that a constant maps to the same constant and an atom of S_1 maps on an atom of S_2 with the same predicate name.

DEFINITION 2.1. *(canonical database of query) A* canonical database *D_Q of conjunctive query Q is derived by freezing the variables of Q to distinct constants and adding in D_Q exactly all frozen subgoals in the body of Q.*

For conjunctive queries canonical database is unique up to renaming. We will use the notation D_Q to denote the canonical database of query Q without mentioning it.

DEFINITION 2.2. *(query containment and equivalence) A query Q_1 is* contained *in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$, if for any database D of the base relations, the answer computed by Q_1 is a subset of the answer 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 $Q_2 \sqsubseteq Q_1$.*

Chandra and Merlin [CM77] show that a conjunctive query Q_1 is contained in another conjunctive query Q_2 if and only if there is *containment mapping* from Q_2 to Q_1. A containment mapping is a *homomorphism* which maps the head and all the subgoals in Q_2 to Q_1. It maps each variable to either a variable or a constant, and maps each constant to the same constant.

DEFINITION 2.3. *(expansion of a query using views) The* expansion *of a query P on a set of views $\mathcal{V}$, denoted P^{exp}, is obtained from P by replacing all the views in P with their corresponding base relations. Existentially quantified variables (i.e., nondistinguished variables) in a view are replaced by fresh variables in P^{exp}.*

We denote by $\mathcal{V}(D)$ the result of computing the views on database D, i.e., $\mathcal{V}(D) = \bigcup_{V \in \mathcal{V}} V(D)$.

DEFINITION 2.4. *(equivalent rewritings) Given a query Q and a set of views $\mathcal{V}$, a query P is an* equivalent rewriting *of query Q using $\mathcal{V}$, if P uses only the views in $\mathcal{V}$, and for any database D on the schema of the base relations it holds: $P(\mathcal{V}(D)) = Q(D)$.*

For conjunctive queries and views the following is shown to be an equivalent definition.

DEFINITION 2.5. *(equivalent rewritings) Given a query Q and a set of views $\mathcal{V}$, a query P is an* equivalent rewriting *of query Q using $\mathcal{V}$, if P uses only the views in $\mathcal{V}$, and P^{exp} is equivalent to Q, i.e., $P^{exp} \equiv Q$.*

A CQ query is called *chain query* if it is defined over binary predicates and also the following holds: The body contains as subgoals a number of binary atoms which if viewed as labeled graph (since they are binary) they form a directed path and the start and end nodes of this path are the arguments in the head. For an example, this is a chain query: $q(X,Y) : -a(X,Z_1), b(Z_1,Z_2), c(Z_2,Y)$.

2.2 Determinacy

For two databases D_1, D_2, $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ means that for each $V_i \in \mathcal{V}$ it holds $V_i(D_1) = V_i(D_2)$.

DEFINITION 2.6. *(views determine query) Let query Q and views $\mathcal{V}$. We say that $\mathcal{V}$* determines *Q if the following is true: For any pair of databases D_1 and D_2, if $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ then $Q(D_1) = Q(D_2)$.*

Thus if a set of views $\mathcal{V}$ determines a query Q, then, given a view instance $I_{\mathcal{V}}$ we know that for any database D such that $I_{\mathcal{V}} = \mathcal{V}(D)$ the answer to the query $Q(D)$ depends only on $I_{\mathcal{V}}$. Hence given a view instance $I_{\mathcal{V}}$, then for any D such that $I_{\mathcal{V}} = \mathcal{V}(D)$ let $A_Q(I_{\mathcal{V}}) = Q(D)$. I.e., we use the notation $A_Q(I_{\mathcal{V}})$ for the set of answers that are computed by the query Q on any database D for which it holds $I_{\mathcal{V}} = \mathcal{V}(D)$.

The following example shows a view set and two queries, one query being determined by the view set the other query not.

EXAMPLE 2.1. *(i) Consider query:*

$$Q : q(X, Y) : -a(X, Z_1), a(Z_1, Z_2), b(Z_2, Y).$$

and views:

$$V_3 : v_3(X, Y) : -a(X, Z'_1), a(Z'_1, Z_2), b(Z_2, Y).$$

$$V_4 : v_4(X) : -b(X, X).$$

View set $\{V_3, V_4\}$ determines Q because V_3 is equivalent to Q.

(ii) Now consider query

$$Q' : q'(X, Y) : -a(X, X), b(X, Y).$$

View set $\{V_3, V_4\}$ does not determine Q' and also there is no rewriting of Q' using $\{V_3, V_4\}$. To see that $\{V_3, V_4\}$ does not determine Q', consider the databases $D_1 = \{a(x, z'_1), a(z'_1, z_2), b(z_2, y)\}$ and $D_2 = \{a(x, x), b(x, y)\}$. The output of the view computation is the same, i.e., $\mathcal{V}(D_1) = \mathcal{V}(D_2) = \{v_3(x, y)\}$ but on D_2 query Q' computes $Q'(D_2) = \{q'(x, y)\}$ and on D_1 query Q' computes the empty set.

DEFINITION 2.7. *(complete language for rewritings) Let $\mathcal{L}_Q$ and $\mathcal{L}_{\mathcal{V}}$ and $\mathcal{L}$ be query languages. We say that a language $\mathcal{L}$ is* complete for $\mathcal{L}_{\mathcal{V}}$-to-$\mathcal{L}_Q$ rewritings *if the following is true for any query Q in $\mathcal{L}_Q$ and set of views $\mathcal{V}$ in $\mathcal{L}_{\mathcal{V}}$: Suppose $\mathcal{V}$ determines Q; then there is a query R in $\mathcal{L}$ such that R is an equivalent rewriting of Q using $\mathcal{V}$.*

The following proposition states some easy observation about query and views when the views determine the query.

PROPOSITION 2.1. *Let query Q and views $\mathcal{V}$ be given by minimized conjunctive queries. Suppose $\mathcal{V}$ determines Q.*

Let Q' be query resulting from Q after deleting one or more subgoals. Let D_Q and $D_{Q'}$ be the canonical databases of Q and Q' respectively. Then the following hold:

a) $\mathcal{V}(D_Q) \neq \mathcal{V}(D_{Q'})$.

b) For any database D, the constants in the tuples in $Q(D)$ is a subset of the constants in the tuples in $\mathcal{V}(D)$.

c) All base predicates appearing in the query definition appear also in the views (but not necessarily vice versa).

d) $\mathcal{V}(D_Q) \neq \emptyset$.

Proof. The proof of the first part of the claim is by contradiction: If not, then $\mathcal{V}(D_Q) = \mathcal{V}(D_{Q'})$ and consequently $Q(D_Q) = Q(D_{Q'})$. Hence there is a homomorphism from Q to $D_{Q'}$ which yields a containment mapping from the subgoals of Q to the subgoals of Q', which is a contradiction. For the second part, suppose x is a constant in $Q(D)$ which does not appear in $\mathcal{V}(D)$. Then let us construct D_1 and D_2 to be isomorphic to D only that in D_1 we have renamed the constant x to c where c is a fresh constant. Now we have $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ but there is a tuple in $Q(D_1)$ which contains c and there is no such tuple in $Q(D_2)$. Hence $Q(D_1) \neq Q(D_2)$ contradiction.

To prove the third part, suppose p_i is the predicate name which appears in the query but does not appear in the views definition. Consider the canonical database D_Q of the query and a database D' which results from D_Q after deleting any fact $p_i(t)$. Then on both D_Q and D' the views compute the same relations but the query does not. The fourth part is a consequence of a similar construction, now D' is empty. Then again, on both D_Q and D' the views compute the same relations but the query does not. $\dashv$

It is easy to show that if there is an equivalent rewriting of a query using a set of views then this set of views determine the query.

PROPOSITION 2.2. *Let Q and $\mathcal{V}$ be query and views. If there is an equivalent rewriting of Q using $\mathcal{V}$ then $\mathcal{V}$ determines Q.*

Proof. Let P be an equivalent rewriting of Q using $\mathcal{V}$. Let D_1 and D_2 be databases such that $\mathcal{V}(D_1) = \mathcal{V}(D_2)$. Then $P(\mathcal{V}(D_1)) = P(\mathcal{V}(D_2))$. Since P is an equivalent rewriting, this yields that $Q(D_1) = Q(D_2)$. $\dashv$

3 Complexity of Query Answering

The following theorem shows that, in the case the views determine the query, then the query answering problem is in $NP \cap coNP$. The complexity is measured on the size of the view instance $I_{\mathcal{V}}$, i.e., we study the data complexity of the problem.

THEOREM 3.1. *Let query Q and views $\mathcal{V}$ be given by conjunctive queries. Suppose $\mathcal{V}$ determines Q. We are given a view instance $I_{\mathcal{V}}$ such that there exists a database D for which $I_{\mathcal{V}} = \mathcal{V}(D)$ and a tuple of constants t. Then it is in $NP \cap coNP$ to decide whether t is in $A_Q(I_{\mathcal{V}})$.*

Proof. We want to decide whether $t \in A_Q(I_{\mathcal{V}})$ or not. We will show that for every $I_{\mathcal{V}}$ there exists at least one database D_h of polynomial size such that $I_{\mathcal{V}} = \mathcal{V}(D_h)$. Hence we can decide by computing $Q(D_h)$.

We construct a database D' over the base relations. We "expand" $I_{\mathcal{V}}$ to D' (recall that $I_{\mathcal{V}}$ is over the schema of the views) by replacing each view tuple by a set of tuples of base relations which are the frozen subgoals of this view's definition. More specifically, nondistinguished variables in the view definition are frozen to fresh constants that are not used for any other frozen variable. (This construction is similar to the construction which obtains the expansion of a rewriting.)

Let D be any database such that $I_{\mathcal{V}} = \mathcal{V}(D)$. Clearly there is a homomorphism from D' to D. Also, for any database D_i such that $\mathcal{V}(D) = \mathcal{V}(D_i)$ if D_i is minimal (in the sense that if we delete one tuple from D_i then $\mathcal{V}(D) \neq \mathcal{V}(D_i)$), then D_i is a homomorphic image of D'. Thus, consider all databases that are homomorphic images of D'. Among those there exists at least one, say D_h, such that $I_{\mathcal{V}} = \mathcal{V}(D_h)$. To compute $A_Q(I_{\mathcal{V}})$ just compute $Q(D_h)$.

The above reasoning puts the problem in $NP \cap coNP$: The size of any homomorphic image of D' is polynomial in the size of $I_{\mathcal{V}}$. Thus, we guess a homomorphic image D_h of D'. Then in polynomial time, we compute the views $\mathcal{V}$ on D_h and verify that it is equal to $I_{\mathcal{V}}$. Finally we compute Q on D_h. Thus t is in $A_Q(I_{\mathcal{V}})$ if t is in $Q(D_h)$. Note that the negation of "t is in $A_Q(I_{\mathcal{V}})$" is equivalent to "t is not in $Q(D)$ for some D such that $\mathcal{V}(D) = I_{\mathcal{V}}$". However because $\mathcal{V}$ determines Q, this is equivalent to t is not in $Q(D_h)$. Thus this puts the problem to $coNP$ too. ⊣

4 Canonical Rewriting

In this section we show that given a query and views that are defined by conjunctive queries, then there is a particular conjunctive query R_c which uses only view atoms as subgoals which has the property: If there is an equivalent rewriting then R_c is an equivalent rewriting.

Let D_Q be the canonical database of Q. We compute the views on D_Q and get view instance $\mathcal{V}(D_Q)$ [ALU01]. We construct *canonical rewriting* R_c as follows. The body of R_c contains as subgoals exactly all unfrozen view tuples in $\mathcal{V}(D_Q)$ and the tuple in the head of R_c is as the tuple in the head of query Q. Here is an example which illustrates this construction.

EXAMPLE 4.1. *Suppose we have the query:*

$$Q : q(X,Y) :- a(X,Z_1), a(Z_1,Z_2), b(Z_2,Y).$$

and the views $\mathcal{V}$:

$$V_1 : v_1(X,Z_2) :- a(X,Z_1), a(Z_1,Z_2).$$

$$V_2 : v_2(X,Y) :- b(X,Y).$$

Then D_Q contains the tuples $\{a(x,z_1), a(z_1,z_2), b(z_2,y)\}$ and $\mathcal{V}(D_Q)$ contains the tuples $\{v_1(x,z_2), v_2(z_2,y)\}$. And thus, R_c is the following:

$$R_c : q(X,Y) :- v_1(X,Z_2), v_2(Z_2,Y).$$

For convenience of reference, we retain in R_c the names of the variables used in Q. Observe that all variables of R_c are also variables of Q but not necessarily vice versa.

PROPOSITION 4.1. *Let Q and $\mathcal{V}$ be conjunctive query and views and R_c be the canonical rewriting. Then the following hold: a) Query Q is contained in the expansion R_c^{exp} of R_c. b) If there is an equivalent rewriting of Q using $\mathcal{V}$ then the canonical rewriting R_c is such an equivalent rewriting.*

Proof. a) By construction of R_c there is a containment mapping from its expansion R_c^{exp} to Q.

b)Suppose there is an equivalent rewriting R of Q using $\mathcal{V}$. Then the expansion R^{exp} of R is equivalent to Q. Hence there is a containment mapping from R^{exp} to Q, and therefore, there is a homomorphism from R^{exp} to D_Q (the canonical database of Q). Thus if a view atom $v(t)$ is in the subgoals of R then there is a homomorphism from the expansion of $v(t)$ to D_Q. This establishes that all subgoals in R must be view atoms that result from view tuples of $\mathcal{V}(D_Q)$. But R_c contains all view tuples in $\mathcal{V}(D_Q)$. Thus, any equivalent rewriting R contains a subset of the subgoals of R_c, and hence R contains R_c and thus Q contains R_c^{exp}. ⊣

5 Query Answering in PTIME

In this section we prove that in certain special cases, if views determine the query then there is an equivalent rewriting, hence, computing the answers to the views can be done in polynomial time on the size of the view instance.

THEOREM 5.1. *Given are a query Q and a set of views $\mathcal{V}$ as in one of the cases below:*

1. *All views with no nondistinguished variables.*
2. *Binary base predicates, chain views and query.*
3. *Binary base predicates, query contains only one variable single binary view with only one non-distinguished variable.*

Suppose $\mathcal{V}$ determines Q. Then the canonical rewriting R_c is an equivalent rewriting of Q using $\mathcal{V}$.

Proof. For each special case we prove that the canonical rewriting R_c is an equivalent rewriting. We assume minimized query and views.

Case 1. (no nondistinguished variables in views)

Since there are no nondistinguished variables in view definitions R_c^{exp} contains exactly the variables of R_c. By construction, there is a one-to-one mapping from the variables of R_c to the variables of Q which can be extended to a containment mapping μ from R_c^{exp} to Q. Moreover, because of Proposition 2.1, μ uses as targets all subgoals of Q. Since the variables of R_c^{exp} are exactly the variables on R_c, μ is one-to-one and onto, hence μ^{-1} is a containment mapping from Q to R_c^{exp}.

Case 2. (Binary base predicates, chain views and query)

Preliminary observations: Let R_c^{exp} be the expansion of the canonical rewriting R_c. Let D_c^{exp} be the canonical database of R_c^{exp} and D_Q be the canonical database of Q. Recall that we keep the same names of variables in R_c as in Q and do the same for the constants in D_c^{exp} and D_Q. As in D_c^{exp} there are more constants than those in D_Q, we call the new constants *fresh* (they come from the frozen variables of the fresh variables in R_c^{exp}). Let X, Y be the variables in the head of Q and x, y the corresponding constants in D_Q. R_c^{exp} is an acyclic (no cycles if the body is viewed as a graph) query (otherwise there is no containment mapping from R_c^{exp} to Q). Let μ be a containment mapping from R_c^{exp} to Q. We may view μ also as a homomorphism from D_c^{exp} to D_Q. We consider the expansion $v_i^{exp}(t)$ of each view subgoal $v_i(t)$ separately in R_c^{exp}, we observe that its image on Q under μ is a chain isomorphic to $v_i^{exp}(t)$; hence for every $v_i^{exp}(t)$, μ^{-1} is an homomorphism. Let G be the underlined undirected graph of D_c^{exp} and G_c be the underlined undirected graph of the canonical database D_c of R_c. Note that connectivity of graphs G and G_c are related because all views definitions have a body which is connected if seen as a graph. Thus, each connected component of G_c creates a connected component of G and vice versa.

The proof uses connectivity arguments on G. Let x and y be the two constants that are the frozen distinguished variables of R_c^{exp}. Then D_c^{exp} can be viewed as a graph with three parts as concerns connectivity of G: (i) a connected component of G that includes x, (ii) a connected component of G that includes y and (iii) the remaining part of G. We have two cases: a) the parts (i) and (ii) are the same, i.e., x and y belong to the same connected component of G and b) the parts (i) and (ii) are disconnected.

For the first case, we argue that there is a rewriting. It is done by induction on the distance between x and y on the graph G. Inductive hypothesis: There is an injective homomorphism from the expansions of the views on the path from x to y on graph G. If the distance is one then there is necessarily a view on the tuple (x, y) and this view consists the rewriting because μ^{-1} provides the containment mapping from R_c^{exp} to Q. The inductive step is also proven by similar argument.

For the second case we argue that this leads to contradiction. We have the following cases:

a) The connected component in G which contains x is empty. This means that in G_c also the connected component that contains X is empty and hence there is no tuple in $\mathcal{V}(D_Q)$ which contains X. Then we construct the following two database instances: D_1 is a copy of D_Q and D_2 is a copy of D_Q with x and the tuples that contains it being deleted. Clearly $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ but $Q(D_1) \neq Q(D_2)$, contradiction. The case where the connected component in G which contains y is empty is symmetrical.

b) There are only two connected components in G, one which contains x and one which contains y. Then we construct the following two database instances: D_1 is a copy of D_Q and D_2 is the disjoint union of two database instances, one for each component of G. For component G_x we take the homomorphic image $\mu(G_x)$ and same for the other component. Note that $\mu(G_x)$ is isomorphic to the longest path in G_x because all homomorphisms that produce view tuples are isomorphisms in our case. We claim that $\mathcal{V}(D_1) = \mathcal{V}(D_2)$. Suppose not. Then there is a view tuple in $\mathcal{V}(D_2)$ which is not in $\mathcal{V}(D_1)$. We argue on the graph G_c inductively on the distance from x. If this tuple contains x then this means that there is a homomorphism of some view definition on D_2 and not on D_1. However this homomorphism on D_2 would have created a view tuple in component G_x and hence would have created a view tuple in D_1 too. This argument can be inductively (on the distance of x' from x in G_c^x, the component of G_c that corresponds to G_x) applied to all x' where x' is a constant in a view tuple in component G_x. Thus all view tuples in component G_x belong to both $\mathcal{V}(D_1)$ and $\mathcal{V}(D_2)$. Hence $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ and by the determinacy assumption we conclude that $Q(D_1) \neq Q(D_2)$, contradiction since tuple $Q(x, y)$ is missing in $Q(D_2)$ because x and y are not connected (chain queries are such that they imply a path from x to y).

c) There are in G at least three connected components and two of those contain x and y each. In this case, we consider D_1, D_2 as in previous case (with the only difference that D_2 is the disjoint union of the homomorphic image under μ for each component of G) but we are not done yet with the construction of counterexample database instances.

There is a homomorphism μ_2 from D_2 to D_1. μ_2 may map several constants in D_2 to the same constant c_i of D_1. We call these constants *copies of* c_i and name them $c_i^1, c_i^2, \ldots$ and we call a view tuple $v(t)$ in D_2 which contains copies of constants of a tuple $v(t')$ in D_1 or D_2 a *copy tuple* of $v(t')$. We consider tuples that are in $\mathcal{V}(D_2)$ and not in $\mathcal{V}(D_1)$. As we showed above such a tuple does not contain constants in G_c^x. Let $v(t)$ be such a tuple. Since D_2 homomorphically maps on D_Q, a copy $v(t')$ of $v(t)$ exists in $\mu(G_x)$ hence in $\mathcal{V}(D_Q)$ hence in D_2. If the component of D_2 in which $v(t')$ appears is such that no "fresh" tuple from G_y appears, then we merge this component (its homomorphic image actually) with

G_x and create D'_2. We do the same for each tuple that satisfies this condition. Thus we need to argue on tuples $v(t')$ that do not satisfy this condition. We do that by an inductive argument on the length of the expansion of view tuple $v(t')$ and in each inductive step we create a connected component in the counterexample database which is shorter than the component we created in the previous inductive step. It remains to argue on the additional tuples in $\mathcal{V}(D_2)$ that have copies in both components G_x and G_y. Suppose there is such a tuple, then G_x and G_y are not disconnected, contradiction. Thus we have constructed databases D_1, D_2 such that $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ and by the determinacy assumption we conclude that $Q(D_1) \neq Q(D_2)$, contradiction.

Case 3. (Binary base predicates, query contains only one variable single binary view with only one non-distinguished variable)

Let the distinguished variables of the view V be called X, Y and the nondistinguished Z. In this case the canonical rewriting is $R_c : q'(A) : -v(A, A)$. We consider the expansion R_c^{exp} of R_c and the canonical database D_c^{exp} of R_c^{exp}. Note that in R_c^{exp} (D_c^{exp} respectively) there are only two variables (constants respectively), let us call them A and B (a and b respectively). Suppose there is no mapping from the query to R_c^{exp}. Then there is a base relation atom $r(X, X)$ in the body of the query and there is no atom $r(X, X)$ in the body of R_c^{exp}. Then there is a database instance D which is a homomorphic image of the view definition with the property: D contains two constants c, d and $V(D)$ contains tuple (c, d) and there is no tuple $r(c, c)$ in D. To construct this database just take the view definition and identify variable Z to Y. Then rename X to c and Y to d. Symmetrically, there is also a database instance D' which is a homomorphic image of the view definition with the property: D' contains two constants c', d' and $V(D')$ contains tuple (d', c') and there is no tuple $r(c', c')$ in D'. To construct this database just take the view definition and identify variable Z to X. Then rename Y to c' and X to d'.

We construct two databases D_1 and D_2. D_1 is constructed by the disjoint union of D_c^{exp} with D and D' and moreover by identifying a with c and c' and then b with d and d'. D_2 is the union of D_1 with the query canonical database and we identify the constant in the query with a. Now both $V(D_1)$ and $V(D_2)$ contain all four possible tuples but $Q(D_2)$ contains (a, a) and $Q(D_1)$ does not, hence contradiction. $\dashv$

6 Determinacy and query equivalence

The problem that we investigate in this paper relates determinacy to query rewriting. Since the problem for the CQ case remains open, we address in this section the question whether a simpler (and probably easier to resolve) variant of the problem may relate determinacy to query equivalence. First we ask: If Q_1 is contained in Q_2 and Q_2 determines Q_1, then are Q_1 and Q_2 equivalent? The following simple example shows that this statement does not hold: Let $Q_1 : q_1(X, X) : -a(X, X)$ and $Q_2 : q_2(X, Y) : -a(X, Y)$. Obviously Q_1 is contained in Q_2. Also Q_2 determines Q_1 because there is an equivalent rewriting of Q_1 using Q_2, it is $R : q(X, X) : -q_2(X, X)$. But Q_1 and Q_2 are *not* equivalent.

We add some stronger conditions: Suppose in addition that there is a containment mapping that uses as targets all subgoals of Q_1 and this containment mapping maps the variables in the head one-to-one. Still the following counterexample shows that we can not conclude that Q_1 and Q_2 are equivalent.

EXAMPLE 6.1. *In this example we have two queries:*
$Q_1 :\ q_1(X, Y, Z, W, A, B) : -r(Y, X), s(Y, X), r(Z, W), s(Z, Z_1), s(Z_1, Z_1), s(Z_1, W), s(A, A_1), s(A_1, A_1), s(A_1, B).$
and
$Q_2 :\ q_2(X, Y, Z, W, A, B) : -r(Y, X), s(Y, X), r(Z, W), s(Z, Z_1), s(Z_1, Z_2), s(Z_2, W), s(A, A_1), s(A_1, A_1), s(A_1, B).$
Clearly Q_1 is contained in Q_2. Also Q_2 determines Q_1 because there is an equivalent rewriting of Q_1 using Q_2:
$R : q'_1(X, Y, Z, W, A, B) : -q_2(X, Y, Z, W, A, B),$
$q_2(X_1, Y_1, Z_1, W_1, Z, W).$
Moreover there is a homomorphism from Q_2 to Q_1 that uses all subgoals of Q_1 and is one-to-one on the head variables. But Q_1 and Q_2 are not equivalent.

Finally, in order to be convinced that R is a rewriting, let us consider the expansion
R^{exp}*:* $q'_1(X, Y, Z, W, A, B) : -r(Y, X), s(Y, X), r(Z, W), s(Z, Z_1), s(Z_1, Z_2), s(Z_2, W), s(A, A_1), s(A_1, A_1), s(A_1, B), r(Y_1, X_1), s(Y_1, X_1), r(Z_1, W_1), s(Z_1, Z'_1), s(Z'_1, Z'_2), s(Z'_2, W_1), s(Z, A'_1), s(A'_1, A'_1), s(A'_1, W).$
Then homomorphism μ_1 is a containment mapping from R^{exp} to Q_1 and homomorphism μ_2 is a containment mapping from Q_1 to R^{exp}:
μ_1*:* $\{X \to X, Y \to Y, Z \to Z, W \to W, A \to A, B \to B, Z_1 \to Z_1, Z_2 \to Z_1, A_1 \to A_1, A'_1 \to Z_1, X_1 \to X, Y_1 \to Y, Z_1 \to Z, W_1 \to W, Z'_1 \to Z_1, Z'_2 \to Z_1\}$
μ_2*:* $\{X \to X, Y \to Y, Z \to Z, W \to W, A \to A, B \to B, A_1 \to A_1, Z_1 \to A'_1\}$.

Finally we add another condition which we denote by $Q_2(D_1) \subseteq_s Q_2(D_2)$, where D_1, D_2 are the canonical databases of Q_1, Q_2 respectively.

We need first explain the notation $Q(D_1) \subseteq_s Q(D_2)$ which in general expresses some structural property of databases D_1 and D_2 with respect to Q and this property is invariant under renaming. We say that $Q(D_1) \subseteq_s Q(D_2)$ holds if there is a renaming of the constants in D_1, D_2 such that $Q(D_1) \subseteq Q(D_2)$. For an example, say we have query $Q : q(X, Y) : -r(X, Y)$ and three database instances $D_1 = \{r(1, 2), r(2, 3)\}$, $D_2 = \{r(a, b), r(b, c)\}$ and $D_3 = \{r(a, b), r(a, c)\}$.

Then it holds that $Q(D_1) \subseteq_s Q_1(D_2)$ and $Q(D_1) \subseteq_s Q(D_2)$ because there is a renaming of D_2 (actually here D_1, D_2 are isomorphic) such that $Q(D_1) \subseteq Q_1(D_2)$ and $Q(D_1) \subseteq Q(D_2)$. But the following does not hold: $Q(D_3) \subseteq_s Q(D_2)$.

We may also allow some constants in D_1, D_2 that are special as concerns renaming. Although we need incorporate these constants in the notation, we will keep (slightly abusively) the same notation here since we always mean the same constants. Thus let us return to queries Q_1, Q_2 and their canonical databases D_1, D_2 respectively. Then by $Q_2(D_1) \subseteq_s Q_2(D_2)$ we mean in addition that (i) the frozen variables in the head of the queries are identical component-wise, i.e., if in the head of Q_1 we have tuple $(X_1, \ldots, X_m)$ then in the head of Q_2 we also have same tuple $(X_1, \ldots, X_m)$ and in both D_1, D_2 these variables freeze to constants $x_1, \ldots, x_m$ and (ii) if we need to rename, then we are not allowed to rename the constants $x_1, \ldots, x_m$.

We introduce a new problem which relates determinacy to query equivalence:

DEFINITION 6.1. *Determinacy and query equivalence: Let Q_1, Q_2 be conjunctive queries. Suppose Q_2 determines Q_1, and Q_1 is contained in Q_2. Suppose also that the following hold: a) there is a containment mapping from Q_2 to Q_1 which (i) uses as targets all subgoals of Q_1 and (ii) maps the variables in the head one-to-one, and b) $Q_2(D_1) \subseteq_s Q_2(D_2)$, where D_1, D_2 are the canonical databases of Q_1, Q_2 respectively. Then are Q_1 and Q_2 equivalent?*

Theorem 6.1 states that if CQ is complete for CQ-to-CQ rewritings then the answer to the above question is "yes".

THEOREM 6.1. *For the following two statements it holds: Statement (A) implies statement (B).*

A) Let Q and $\mathcal{V}$ be conjunctive query and views. Suppose $\mathcal{V}$ determines Q. Then there is an equivalent rewriting of Q using $\mathcal{V}$.

B) Let Q_1, Q_2 be conjunctive queries. Suppose Q_2 determines Q_1, and Q_1 is contained in Q_2. Suppose also that the following hold: a) there is a containment mapping from Q_2 to Q_1 which (i) uses as targets all subgoals of Q_1 and (ii) maps the variables in the head one-to-one, and b) $Q_2(D_1) \subseteq_s Q_2(D_2)$, where D_1, D_2 are the canonical databases of Q_1, Q_2 respectively. Then Q_1 and Q_2 are equivalent.

Proof. Suppose statement (A) is true. Since Q_2 determines Q_1 and statement (A) is true, there is an equivalent rewriting of Q_1 using Q_2. Then the canonical rewriting R_c is such a rewriting (see Proposition 4.1). Hence R_c^{exp} is equivalent to Q_1. Thus, if we prove that R_c^{exp} is equivalent to Q_2, then this implies that Q_1 and Q_2 are equivalent. We know that there is a containment mapping from Q_2 to R_c^{exp}. In order to prove that there is a containment mapping from R_c^{exp} to Q_2, we observe that $Q_2(D_1) \subseteq_s Q_2(D_2)$ implies such a mapping. The reason is that there is a containment mapping from R_c^{exp} to Q_1 and this mapping produces some tuples in $Q_2(D_1)$. Therefore analogous tuples must be produced in $Q_2(D_2)$. Hence there is a containment mapping from R_c^{exp} to Q_2. $\dashv$

The *determinacy and query equivalence* question remains open. The following theorem settles a special case where we have replaced condition (b) in Definition 6.1 with a stronger one.

THEOREM 6.2. *Let Q_1, Q_2 be conjunctive queries. Suppose Q_2 determines Q_1, and Q_1 is contained in Q_2. Suppose also that the following hold: a) there is a containment mapping that uses as targets all subgoals of Q_1 and this containment mapping maps the variables in the head one-to-one, and b) $Q_2(D_1)$ contains exactly one tuple, where D_1 is the canonical databases of Q_1. Then Q_1 and Q_2 are equivalent.*

Proof. Intuitively the first condition in the statement of the theorem says that Q_1 is a homomorphic image of Q_2, i.e., formed from Q_2 by identifying variables. Moreover it also says that the distinguished variables are *not* to be identified, so Q_1 is Q_2 with (perhaps) nondistinguished variables identified. Suppose towards contradiction that Q_2 and Q_1 are not equivalent. Then there is a query Q_1' such that $Q_1 \sqsubseteq Q_1' \sqsubseteq Q_2$ and Q_1' differs from Q_1 only in that Q_1 results from Q_1' by identifying only two variables (not both distinguished). The following lemma states formally this observation.

LEMMA 6.1. *Suppose Q_1 is contained in Q_2 but they are not equivalent. Then there is a query Q_1' with the properties: a) Q_1' is contained in Q_2 b) Q_1 is properly contained in Q_1' and c) the containment mapping h from Q_1' to Q_1 is identity for all variables of Q_1' except that $h(X_1) = h(X_2) = X_1$.*

Proof. We construct Q_1'. Observe that during the construction we conveniently keep the names of "corresponding" variables in Q_1 and Q_1' (except X_2 which appears only in Q_1').

Let h_1 be the homomorphism from the subgoals of Q_2 to the subgoals of Q_1. Based on h_1 we construct homomorphism h which defines a homomorphic image of Q_2 such that it has the properties of the body of Q_1' as in the statement of the lemma. Since Q_1 and Q_2 are not equivalent and also have the properties as in the statement of Theorem 6.2, there are variables X_1, X_2 of Q_2 such that $h_1(X_1) = h_1(X_2)$. Then h is the following: It is the same as h_1 except that for h we have $h(X_1) \neq h(X_2)$. The image $h(Q_2)$ defines the body of Q'_1 and its head is as the head of Q_2. Hence in Q'_1 we have all variable names as in Q_1 and an additional variable X_2. It is easy to see that Q'_1 properly contains Q_1 and is contained in Q_2. $\dashv$

Let Q'_1 be a query with the properties of the lemma above. Consider the canonical database D'_1 of Q'_1 and compute $Q_2(D'_1)$. We claim that $Q_2(D'_1)$ contains at most two tuples. Because: $Q_2(D_1)$ (which contains only one tuple) is a homomorphic image of $Q_2(D'_1)$ and thus (because Q'_1 has the property in Lemma 6.1 wrto Q_1, i.e, the homomorphism h which maps Q'_1 to Q_1 is the identity except what concerns the variables X_1 and X_2) there may be only one additional tuple in $Q_2(D'_1)$, the one that contains X_2.

Thus we have two cases: a) $Q_2(D'_1)$ contains one tuple. b) $Q_2(D'_1)$ contains two tuples. In the first case, $Q_2(D'_1) = Q_2(D_1)$, hence $Q_1(D'_1) = Q_1(D_1)$ which is false, hence a contradiction. In the second case, we construct database D''_1 : D''_1 is $D_1$with additional facts the frozen subgoals that contain X_2 in Q'_1 – observe that D'_1 is a subset of D''_1. We observe that $Q_2(D''_1)$ contains at most two tuples for the same reason for which $Q_2(D'_1)$ contains at most two tuples. Hence $Q_2(D''_1) = Q_2(D'_1)$ and consequently $Q_1(D''_1) = Q_1(D'_1)$ a contradiction. $\dashv$

Theorem 6.2 covers interesting special cases that include: a) queries do not contain self-joins and b) query Q_1 contains a single variable.

7 Principal view set reduction

Considering the two statements in Theorem 6.1 we do not know whether (B) implies (A), i.e., whether the problem of "determinacy and query equivalence" is as hard as the original problem of whether CQ is complete for CQ-to-CQ rewritings. Short of proving that we show in this section that if we relax the requirement about the view set containing only one view then we can reduce the original problem to it. By "reduction" here (and in the next section) we mean that if we can solve the special case then we can also solve the general case.

Thus in this section we prove that CQ is complete for CQ-to-CQ rewritings only if CQ is complete for CQ-to-CQ rewritings in the special case where the view set includes also a view with the properties wrto the query Q same as the properties of query Q_2 wrto query Q_1 in the definition of the "Determinacy and query equivalence" problem. We call this view *principal view* and call *principal view set* a view set that contains a principal view. The expected property of a principal view set $\mathcal{V}$ is the following: if there is an equivalent rewriting of Q using $\mathcal{V}$ then a projection of the principal view can produce such a rewriting. Proposition 7.2 proves it. In the rest of this section, after we define formally principal view and view set, we show in Theorem 7.1 how reduce the general problem to the special case where the view set is principal.

Let D_p^{exp} be the canonical database of the expansion V_p^{exp} of the principal view V_p. We use the notation $V_p(D_Q) \subseteq_s V_p(D_p^{exp})$ where again we assume that a set of frozen variables are shared between D_Q and D_p^{exp} and are not allowed to be renamed. This is the set of variables in the head of V_p which includes the set of variables in the head of Q and is a subset of the set of variables in Q. In the following definition we have restricted a little more (than explained in the beginning of this section) the definition of the principal view (we have included constrain (b) in the Definition 7.1). The reason is that the proofs did work also for this more restricted case.

DEFINITION 7.1. *(principal view set) Let Q and $\mathcal{V}$ be CQ query and views.*

- *A view $V_p \in \mathcal{V}$ is called* principal view for Q *if the following hold: a) There is a homomorphism from the definition of V_p to Q which uses all subgoals of Q as targets and is one-to-one for the distinguished variables of V_p. b) Let t be the tuple of frozen distinguished variables in D_p^{exp}. Then all tuples in $V_p(D_Q)$ use only constants from t and c) $V_p(D_Q) \subseteq_s V_p(D_p^{exp})$.*
- *A view set is called* principal view set for Q *if it contains a principal view V_p.*

EXAMPLE 7.1. *We use the queries Q_1, Q_2 defined in Example 6.1. Thus let $Q = Q_1$ and let the view set contain a single view $V = Q_2$. Then conditions (a) and (b) in Definition 7.1 are true but condition (c) is not.*

In order to show that $V(D_Q) \subseteq_s V(D^{exp})$ does not hold let us consider canonical databases, D_Q, of query Q and D^{exp}, of the expansion of view V. It is important however to name constants in D^{exp} according to the targets in Q of the homomorphism claimed in part (a) in Definition 7.1. Thus these databases are:
$D_Q = \{r(y,x), s(y,x), r(z,w), s(z,z_1), s(z_1,z_1), s(z_1,w), s(a,a_1), s(a_1,a_1), s(a_1,b)\}$
$D^{exp} = \{r(y,x), s(y,x), r(z,w), s(z,z'_1), s(z'_1,z'_2), s(z'_2,w), s(a,a_1), s(a_1,a_1), s(a_1,b)\}$
We compute $V(D_Q) = \{(x,y,z,w,a,b), (x,y,z,w,z,w)\}$ and $V(D^{exp}) = \{(x,y,z,w,a,b)\}$. Hence $V(D_Q) \subseteq_s V(D^{exp})$ does not hold, therefore V is not a principal view for Q and view set $\{V\}$ is not a principal view set for Q.

Given arbitrary query Q and views $\mathcal{V}$ we can construct a principal view from the canonical rewriting R_c. We call this view *canonical view* and denote by V_c.

DEFINITION 7.2. *We construct view V_c to have as subgoals exactly all subgoals of the expansion of R_c and in the head of V_c we have exactly all variables of R_c.*

Equivalently, V_c can be viewed as the expansion of the extended canonical rewriting *of Q using $\mathcal{V}$. The* extended canonical rewriting *R'_c can be obtained from the canonical rewriting R_c by retaining the body as is and projecting all variables in the head (R_c's head contains in general only a subset of the variables in the head).*

EXAMPLE 7.2. *From Example 4.1 we have that the canonical rewriting for $\{V_1, V_2\}$ and Q is*

$$R_c : q(X, Y) : -v_1(X, Z_2), v_2(Z_2, Y).$$

and hence the canonical view is

$$V_c : v_c(X, Y, Z_2) : -a(X, Z_1), a(Z_1, Z_2), b(Z_2, Y).$$

PROPOSITION 7.1. *Let Q and $\mathcal{V}$ be query and views defined by conjunctive queries. Then the canonical view V_c is a principal view for Q.*

Proof. Clearly $V_c(D_Q)$ contains at least one tuple t by construction of V_c. Moreover t is obtained by a homomorphism which maps one-to-one the distinguished variables of V_c and uses all subgoals of Q according to Proposition 2.1. This proves part (a) of Definition 7.1.

Now observe that any homomorphism from the definition of V_c to D_Q creates view tuples in $\mathcal{V}(D_Q)$. Suppose there is an homomorphism from the definition of V_c to D_Q which does not use constants in t. Then this homomorphism creates view tuples not in R_c which is a contradiction because R_c contains as subgoals all view tuples in $\mathcal{V}(D_Q)$. This proves part (b) of the definition. For the same reason, the following leads to contradiction and hence proves part (c) of the definition: If there is a tuple in $V_p(D_Q)$ and not in $V_p(D_p^{exp})$ then this yields that a view tuple in $\mathcal{V}(D_Q)$ does not appear in the subgoals of R_c. $\dashv$

The following theorem reduces the general problem to the problem where we have a principal view set.

THEOREM 7.1. *Let Q and $\mathcal{V}$ be query and views. Then there is a principal view set $\mathcal{V}'$ such that the following hold:*

1. If $\mathcal{V}$ determines Q then $\mathcal{V}'$ determines Q.

2. If there is an equivalent rewriting of Q using $\mathcal{V}'$ then there is an equivalent rewriting of Q using $\mathcal{V}$.

Proof. View set $\mathcal{V}'$ is the union of $\mathcal{V}$ and the set that contains only the canonical view. We proved in Proposition 7.1 that the canonical view is a principal view. Clearly (1) in the statement of the theorem holds because $\mathcal{V}$ is a subset of $\mathcal{V}'$.

To prove (2), suppose R is an equivalent rewriting of Q using $\mathcal{V}'$. Then R uses views in $\mathcal{V}$ and possibly the principal view V_c of $\mathcal{V}'$. If R does not use V_c then R is also a rewriting of Q using $\mathcal{V}$. Otherwise, we claim that the following rewriting R' is an equivalent rewriting: R' is produced from R by replacing all subgoals of V_c in the body of R by the body of the extended canonical rewriting R'_c. In more detail, let $v_c(\bar{X})$ be a subgoal in R; we replace $v_c(\bar{X})$ by the body of R'_c after unifying appropriately $\bar{X}$ with the variables in the head of R'_c and carrying this unification over to the body of R'_c. Note (see Definition 7.2) that the expansion of R'_c is equivalent to the principal view V_c. Hence the expansion of R'_c is equivalent (as queries) to the expansion of R_c.

The following proposition is an immediate consequence of Proposition 4.1.

PROPOSITION 7.2. *Let Q and $\mathcal{V}$ be query and views such that $\mathcal{V}$ is a principal view set for Q. If there is an equivalent rewriting of Q using $\mathcal{V}$, then there is an equivalent rewriting which is a projection of the principal view of $\mathcal{V}$.*

Proposition 7.2 states that in the case of principal view set, if there is an equivalent rewriting then a projection of the principal view is an equivalent rewriting. Note that in Section 6 we essentially discussed a case where the principal view does not need a projection to produce a rewriting.

8 Other Reductions

In this section we prove the following theorem and provide some formal considerations about connectivity too. By CQ$_{bin}$ we mean the language of CQs where the base predicates are binary. By CQ$_{Bool,bin}$ we mean the language of CQ$_{bin}$ where the query is boolean.

THEOREM 8.1. *If the language of CQ is complete for CQ$_{bin}$-to-CQ$_{Bool,bin}$ rewritings then it is complete for CQ-to-CQ rewritings.*

8.1 Binary base predicates

Here we prove:

THEOREM 8.2. *If the language of CQ is complete for CQ$_{bin}$-to-CQ$_{bin}$ rewritings then it is complete for CQ-to-CQ rewritings.*

The above theorem is an immediate consequence of the following theorem.

THEOREM 8.3. *Let Q and $\mathcal{V}$ be query and views. Then there is a view set $\mathcal{V}'$ and a query Q' defined on binary base predicates such that the following hold:*

1. If $\mathcal{V}$ determines Q then $\mathcal{V}'$ determines Q'.

2. If there is an equivalent rewriting of Q' using $\mathcal{V}'$ then there is an equivalent rewriting of Q using $\mathcal{V}$.

Proof. We construct query Q' and views $\mathcal{V}'$ as follows: We introduce for each base predicate p_i of arity r_i a collection of $r_i(r_i - 1)/2$ distinct binary base predicates

$a^i_{j,j'}, j < j' = 1, \ldots, r$. From query Q (view V_j respectively) we construct query Q' (view V'_j respectively) as follows. We replace each occurrence of predicate atom $p_i(X_1, \ldots, X_r)$ by $r_i(r_i - 1)/2$ binary predicates atoms which in particular are: $a^i_{j,j'}(X_j, X_{j'}), j < j' = 1, \ldots, r$. Intuitively each base predicate is replaced by a clique but the edges of the clique have different predicate names in order to retain the information about the positions of the variables when they are used as arguments in the non-binary predicates.

Proof of part 1. Let D'_1, D'_2 be databases over the binary predicates such as $\mathcal{V}'(D'_1) = \mathcal{V}'(D'_2)$. We construct from D'_1, D'_2 database instances D_1, D_2 over the predicates used in $\mathcal{V}$. For each homomorphic image of a clique in D'_i we add in D_i a tuple accordingly in a way that corresponds to the construction above. The homomorphic mappings of the views $\mathcal{V}'$ on D'_i carry over to homomorphic mappings of the views $\mathcal{V}$ on D_i and vice versa. Hence $\mathcal{V}'(D'_i) = \mathcal{V}(D_i)$ and $Q'(D'_i) = Q(D_i)$. Therefore $\mathcal{V}(D_1) = \mathcal{V}(D_2)$. This yields that $Q(D_1) = Q(D_2)$.

Proof of part 2. If there is an equivalent rewriting of Q' using $\mathcal{V}'$ then we claim that the same with the primed view atoms replaced by unprimed view atoms is an equivalent rewriting of Q using $\mathcal{V}$. The containment mappings that are used in the former case carry over to the latter because the distinct binary base predicates introduced take care of the positions of the variables in the non-binary base predicates of the latter case. $\dashv$

8.2 Boolean Queries Here we prove Theorem 8.1 which is an immediate consequence of the following:

THEOREM 8.4. *Let Q and $\mathcal{V}$ be query and views over binary predicates. Then there is a view set $\mathcal{V}'$ and a boolean query Q' over binary predicates such that the following hold:*

1. If $\mathcal{V}$ determines Q then $\mathcal{V}'$ determines Q'.

2. If there is an equivalent rewriting of Q' using $\mathcal{V}'$ then there is an equivalent rewriting of Q using $\mathcal{V}$.

Proof. Given query Q of arity r and views $\mathcal{V}$ we construct query Q' and views $\mathcal{V}'$ as follows. First introduce r new predicates for new base relations let them be $a_1, \ldots, a_r$. Query Q' is a Boolean query with all subgoals in Q and additional subgoals one for each argument in the head. More specifically, if variable X is in the head in position i then we add $a_i(X, Z)$ where Z is also a new variable used (different from all other variables used also in Q). The views $\mathcal{V}'$ include all views in $\mathcal{V}$ and some new views. We add new views one for each a_i defined by $v_i(t) : -a_i(t)$.

Now consider two databases on which views $\mathcal{V}'$ compute the same relations. Since $\mathcal{V}$ is a subset of $\mathcal{V}'$ then views $\mathcal{V}$ compute the same relations on both databases. Hence query Q computes the same relation on both databases. If the query output is empty on both then query Q' is false. Otherwise it might be either false or true depending on how the new relations $v_i, i = 1, 2 \ldots$ compute on the two databases. However, since they produce the same output *and* Q produces the same output on both, then Q' is either false on both or true on both. The reason is that Q' can be rewritten using Q and the relations $v_i, i = 1, 2 \ldots$. $\dashv$

8.3 Connectivity The following example shows the intuition for the result in this section.

EXAMPLE 8.1. *Suppose we have query:*

$$Q : Q(X) : -r(Y, X), s(Y, X), s(Z, Z_1), s(Z_1, Z).$$

and views $\mathcal{V}$:

$$V_1 : v_1(X, Y) : -r(Y, X).$$

$$V_2 : v_2(X, Y) : -s(Y, X), s(Z, Z_1), s(Z_1, Z).$$

Clearly the rewriting $R_1 : Q(X) : -v_1(X, Y), v_2(X, Y)$. is an equivalent rewriting of Q using $\mathcal{V}$. Note however, that the two last subgoals in the query are (what we call in the definition below semi-covered) such that either they are covered by a single view among the views in the particular view set or not at all. The reason is that these two subgoals do not contain any variables in the view tuples of $\mathcal{V}(D_Q)$ (D_Q is the canonical database of Q) hence it is not possible to "glue together" several views to "cover" these two subgoals. In that case, we might be able to simplify the problem by reducing it to the following query Q' and views $\mathcal{V}$:

$$Q' : Q'(X) : -r(Y, X), s(Y, X).$$

and views $\mathcal{V}$:

$$V'_1 : v'_1(X, Y) : -r(Y, X).$$

$$V'_2 : v'_2(X, Y) : -s(Y, X).$$

Then the same rewriting provides an equivalent rewriting of Q' using $\mathcal{V}'$. We make this formal in this subsection.

DEFINITION 8.1. *(Connectivity graph of query) Let Q be a conjunctive query. The nodes of the* connectivity graph *of Q are all the subgoals of Q and there is an (undirected) edge between two nodes if they share a variable or a constant.*

A *connected component* of a graph is a maximal subset of its nodes such that for every pair of nodes in the subset there is a path in the graph that connects them. A *connected component of a query* is a subset of subgoals which define a connected component in the connectivity graph. A graph with only one connected component is called *connected.* A query is *connected* if its connectivity graph is connected.

DEFINITION 8.2. *(semi-covered component) Let Q and $\mathcal{V}$ be CQ query and views. Let G be a connected component of query Q. Suppose that any variable in G is such that there is no tuple in $\mathcal{V}(D_Q)$ (D_Q is the canonical database of Q) that contains it. Then we say that G is a* semi-covered component of Q wrto $\mathcal{V}$.

We prove the following which is the main result of this subsection.

THEOREM 8.5. *If the language of CQ is complete for CQ-to-CQ rewritings where the query does not contain any semi-covered components wrto the views then it is complete for CQ-to-CQ rewritings.*

THEOREM 8.6. *Let Q and $\mathcal{V}$ be query and views. Then there is a view set $\mathcal{V}'$ and a query Q' such that Q' has no semi-covered components wrto $\mathcal{V}'$ and the following hold:*

1. If $\mathcal{V}$ determines Q then $\mathcal{V}'$ determines Q'.

2. If there is an equivalent rewriting of Q' using $\mathcal{V}'$ then there is an equivalent rewriting of Q using $\mathcal{V}$.

Proof. We define:

DEFINITION 8.3. *(covering subgoals) Let view V, query Q and let S be a subset of the subgoals of Q. We say that V* covers *S if there is a homomorphism μ from the view definition to Q such that S is a subset of $\mu(v^{exp})$ i.e., a subset of the targets of the view's subgoals under μ.*

We use the following lemma:

LEMMA 8.1. *Let Q and $\mathcal{V}$ be conjunctive query and views. Suppose $\mathcal{V}$ determines Q. Let G_Q be a disconnected component of Q which is semi-covered by $\mathcal{V}$. Then there is a view in $\mathcal{V}$ which contains a connected component which is isomorphic to G_Q.*

Proof. Because of Proposition 2.1, all subgoals of G_Q are targets of some view tuple mapping when we compute $\mathcal{V}(D_Q)$.

First we need to prove that in any mapping μ from the views to D_Q there is only one view which covers G_Q. Towards contradiction, suppose there is a mapping μ for which G_Q is covered by more than one view, say it is covered by views (wlog) v_1, v_2. This means that the union of $\mu(v_1)$ and $\mu(v_2)$ contains all subgoals in G_Q. Now, we do some construction: First rename all variables in $\mu(v_1)$ and $\mu(v_2)$ so that they take names from disjoint sets for each $\mu(v_i)$. Then consider canonical databases for $\mu(v_i)$'s and G_Q – for simplicity we keep the same name. We construct a database D_1 by taking D_Q and deleting G_Q and adding all $\mu(v_i)$'s. Clearly D_1 and D_Q are such that $\mathcal{V}(D_1) = \mathcal{V}(D_Q)$ (because component G_Q did not produce any view tuples, and its replacement G'_Q is such that G_Q is a homomorphic image of G'_Q, hence G'_Q does not produce any view tuples either). Then D_1 and D_Q are such that $\mathcal{V}(D_Q) = \mathcal{V}(D_1)$ but $Q(D_Q) \neq Q(D_1)$. $\dashv$

We construct Q' and $\mathcal{V}'$. Let Q' be the query which has the same head as Q and all subgoals of Q except those that are semi-covered. Let $\mathcal{V}'$ be a set of views same as the set $\mathcal{V}$ but in their definition we have deleted the subgoals that cover semi-covered components of the query. It is easy to see from the above lemma that if there is an equivalent rewriting R of Q' using $\mathcal{V}'$ then there is an equivalent rewriting R_1 of Q using $\mathcal{V}$.

Now the first part in the statement of Theorem 8.6 remains to be proven. Let D_1 and D_2 be databases on which $\mathcal{V}'$ computes the same answer, i.e., $\mathcal{V}'(D_1) = \mathcal{V}'(D_2)$. We want to construct database instances D'_1, D'_2 such that $\mathcal{V}(D'_1) = \mathcal{V}(D'_2)$ and $Q'(D_i) = Q(D'_i, i = 1, 2$. It is worth noting that a construction that is straightforward does not work: Construct D'_i as the disjoint union of D_i and database D_M which is a copy of all semi-covered components of Q (with variables frozen appropriately similar to the case of constructing a canonical database of a query). The reason is that a tuple $v_i(t)$ might exist which is in $\mathcal{V}(D'_1)$ and not in $\mathcal{V}(D'_2)$. This is possible for tuples t which contain constants from D_M – remember that databases D_1, D_2 may be quite different from each other.

Construction of databases D'_1 and D'_2. We will construct databases D'_1 and D'_2 as follows: They are both isomorphic copies of a database D where D is the disjoint union of D_1 and D_2 and database D_M. Observe that we can partition the variables of D into three subsets A_1, A_2, A_3 which are those that belong to D_1, those that belong to D_2 and those that belong to D_M. By hypothesis, we have that $\mathcal{V}'$ computes the same view tuples on D_1 and D_2. Thus, we can rename the constants in D so that we create two databases D'_1 and D'_2: In D'_1 we do not do any renaming. In D'_2, we swap names among constants that are contained in view tuples when we are computing views $\mathcal{V}'(D_1)$ and $\mathcal{V}'(D_2)$. More precisely, if view tuple $v'_j(x_1, x_2, x_3)$ is in $\mathcal{V}'(D_1)$ then a "corresponding" view tuple $v'_j(y_1, y_2, y_3)$ is in $\mathcal{V}'(D_2)$. We swap x_i for y_i. This kind of swap would run into inconsistencies if it were not true that $\mathcal{V}'(D_1) = \mathcal{V}'(D_2)$. Hence it is feasible in our case.

Now it is easy to show that on D'_1 and D'_2, views $\mathcal{V}$ compute the same tuples (notice that although D'_1 and D'_2 are isomorphic, because of the renaming, they might not compute the same set of view tuples). Hence the answers of Q are the same on D'_1 and D'_2. This implies in a straightforward way that the answers of Q' are the same on D_1 and D_2.

9 Conclusion

We have investigated the problem whether CQ is complete for CQ-to-CQ rewritings. We have presented a number of special cases for which CQ is complete. On the other end, we have reduced the general problem to more restricted variants. A new problem is introduced in Section 6 that relates determinacy to query equivalence and seems "easier" to resolve than the original problem. However this also remains open, since here we have solved only a (pretty broad) special case of it. Reducing the original problem to the *determinacy and query equivalence* problem is another open question. Further we provided a reduction in Section 7 which shows that the problem can be reduced to the case where a projection of a special view provides an equivalent rewriting if there exists one.

References

[ACN00] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in microsoft sql server. In *Proc. of VLDB*, 2000.

[AD98] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

[ALM02] Foto Afrati, Chen Li, and Prasenjit Mitra. Answering queries using views with arithmetic comparisons. In *PODS*, 2002.

[ALU01] Foto Afrati, Chen Li, and Jeffrey D. Ullman. Generating efficient plans using views. In *SIGMOD*, pages 319–330, 2001.

[B$^+$97] Roberto J. Bayardo Jr. et al. Infosleuth: Semantic integration of information in open and dynamic environments (experience paper). In *SIGMOD*, pages 195–206, 1997.

[C$^+$94] Sudarshan S. Chawathe et al. The TSIMMIS project: Integration of heterogeneous information sources. *IPSJ*, pages 7–18, 1994.

[CdGLV02] D. Calvanese, G. de Giacomo, M. Lenzerini, and M. Vardi. Lossless regular views. In *PODS*. ACM, 2002.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *ICDE*, pages 190–200, 1995.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. *STOC*, pages 77–90, 1977.

[CR97] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *ICDT*, 1997.

[DG97] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *PODS*, pages 109–116, 1997.

[FLSY99] Daniela Florescu, Alon Levy, Dan Suciu, and Khaled Yagoub. Optimization of run-time management of data intensive web-sites. In *Proc. of VLDB*, pages 627–638, 1999.

[GT00] Stéphane Grumbach and Leonardo Tininini. On the content of materialized aggregate views. In *PODS*, pages 47–57, 2000.

[HKWY97] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, pages 276–285, 1997.

[IFF$^+$99] Zachary Ives, Daniela Florescu, Marc Friedman, Alon Levy, and Dan Weld. An adaptive query execution engine for data integration. In *SIGMOD*, pages 299–310, 1999.

[LBU01] Chen Li, Mayank Bawa, and Jeff Ullman. Minimizing view sets without losing query-answering power. In *ICDT*, 2001.

[LMSS95] Alon Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.

[LRO96] Alon Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, 1996.

[RSU95] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *PODS*, pages 105–112, 1995.

[SV05] Luc Segoufin and Victor Vianu. Views and queries: Determinacy and rewriting. In *PODS*. ACM, 2005.

[TS97] Dimitri Theodoratos and Timos Sellis. Data warehouse configuration. In *Proc. of VLDB*, 1997.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *ICDT*, pages 19–40, 1997.