Rewriting Conjunctive Queries Determined by Views

Foto Afrati

National Technical University of Athens, Greece
afrati@softlab.ntua.gr

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. Constructing rewritings is a widely studied technique to derive those answers. In this paper we consider the problem of existence of rewritings in the case where the answers to the views uniquely determine the answers to the query. Specifically, 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 and investigate the question: If a view set $\mathcal{V}$ determines a query Q, is there an equivalent rewriting of Q using $\mathcal{V}$? We present here interesting cases where there are such rewritings in the language of conjunctive queries. Interestingly, we identify a class of conjunctive queries, CQ_{path}, for which a view set can produce equivalent rewritings for "almost all" queries which are determined by this view set. We introduce a problem which relates determinacy to query equivalence. We show that there are cases where restricted results can carry over to broader classes of queries.

1 Introduction

The problem of using materialized views to answer queries [19] has received considerable attention because of its relevance to many data-management applications, such as information integration [6,11,16,18,20,26], data warehousing [25],[5] web-site designs [14], and query optimization [10]. 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 $\mathcal{V}(D)$? Constructing rewritings is a widely used and extensively studied technique to derive those answers [17].

A related fundamental question concerns the information provided by a set of views for a specific query. In that respect, 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 $Q(D_1) = Q(D_2)$ [24]. A 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

L. Kučera and A. Kučera (Eds.): MFCS 2007, LNCS 4708, pp. 78–89, 2007.

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 $Q(D)$. Hence, a natural question to ask is: if a set of views determines a query is there an equivalent rewriting of the query using the views? In this paper we consider the case where query and views are defined by conjunctive queries (CQ for short) and investigate decidability of determinacy and the existence of equivalent rewriting whenever a view set determines a query.

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-$\mathcal{L}_Q$ rewriting if whenever a set of views $\mathcal{V}$ in $\mathcal{L}_\mathcal{V}$ determines a query Q in $\mathcal{L}_Q$ then there is an equivalent rewriting of Q in $\mathcal{L}$ which uses only $\mathcal{V}$. We know that CQ is not complete for CQ-to-CQ rewriting [22]. However there exist interesting special cases it is complete [24,22].

In this paper we consider subclasses of CQs and investigate a) decidability of determinacy, b) special cases where CQ or first order logic is complete for rewriting and c) the connection between determinacy and query equivalence. In more detail, our contributions are the following:

1. We show that CQ is complete for the cases a) where the views are full (all variables from the body are exported to the head) and b) where query has a single variable and view set consists of a single view with two variables.
2. We show that determinacy is decidable for chain queries and views.
3. We identify a class of conjunctive queries, CQ_{path}, which is almost complete for CQ_{path}-to-CQ_{path} rewriting. This is the first formal evidence that there are well behaved subsets of conjunctive queries.
4. Query rewritings using views is a problem closely related to query equivalence. Hence it is natural to ask what is the connection between determinacy and query equivalence. We investigate this question and introduce a new problem which concerns a property a query language may have and is a variant of the following: For a given query language, if Q_1 is contained in Q_2 and Q_2 determines Q_1, then are Q_1 and Q_2 equivalent? We solve special cases of it such as for CQ queries without self joins.
5. We make formal the observation that connectivity can be used to simplify the problem of determinacy and as a result of it we provide more subclasses with good behavior.

1.1 Related Work

In [24], 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 for UCQ-to-UCQ (i.e., unions of CQs) rewriting, 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 $\text{CQ}^{\neq}$-to-CQ rewritings. In [22,24], special classes of conjunctive queries and views are

identified for which the language of conjunctive queries is complete: when views are unary or Boolean and when there is only one path view. It is shown that determinacy is undecidable for views and queries in the language of union of conjunctive queries [24].

Determinacy and notions related to it are also investigated in [15] where the notion of subsumption is introduced and used to the definition of complete rewritings and in [7,8] 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 [19] where it is proven that it is NP-complete to decide whether a given CQ query has an equivalent rewriting using a given set of CQ views, [12] where polynomial subcases were identified. In [23], [4], [13] 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 for rewritings in a certain language when there is no equivalent rewriting in the language and want to obtain a query that uses only the views and computes as many certain answers [1] as possible. In [21] 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 [7] 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 $g_i(\bar{X}_i)$ in the *body* is a *relational atom*, where 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 $h(\bar{X})$.

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* or *database* is a finite relational structure with only ground atoms. The body of a conjunctive query can be also viewed as a relational structure and we call it *canonical database of query* Q and denote D_Q; we say that in D_Q the variables of the query are *frozen* to distinct constants. A query Q_1 is *contained* in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$, if for any database D on the base relations, the answer computed by Q_1 is a subset of the answer by Q_2, i.e., $Q_1(D) \subseteq Q_2(D)$. 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 [9] 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. A CQ query Q is *minimized* if by deleting any subgoal we obtain a query which is not equivalent to Q. We denote by $\mathcal{V}(D)$ the result of computing the views on database D, i.e., $\mathcal{V}(D) = \bigcup_{V \in \mathcal{V}} V(D)$, where $V(D)$ contains atoms $v(t)$ for each answer t of view V.

Definition 1 (*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: $P(\mathcal{V}(D)) = Q(D)$.*

The *expansion* of a CQ query P on a set of CQ 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}. For conjunctive queries and views a conjunctive query P is an equivalent rewriting of query Q using $\mathcal{V}$ iff $P^{exp} \equiv Q$.

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 (*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 $Q(D_1) = Q(D_2)$.*

Let $\mathcal{L}$ be a query language or a set of queries (it will be clear from the context). We say that a subset $\mathcal{L}_1$ of $\mathcal{L}$ *contains almost all* queries in $\mathcal{L}$ if the following holds: Imagine $\mathcal{L}$ as a union of specific sets of queries, called *eq-sets* such that each eq-set contains exactly all queries in $\mathcal{L}$ that are equivalent to each other (i.e., every two queries in a particular eq-set are equivalent). Then $\mathcal{L}_1$ contains all queries in $\mathcal{L}$ except those queries contained in a finite number of eq-subsets.

Definition 3 ((*almost*) *complete language for rewriting*)**.** *Let $\mathcal{L}_Q$ and $\mathcal{L}_\mathcal{V}$ be query languages or sets of queries. Let $\mathcal{L}$ be query language.*

We say that $\mathcal{L}$ is complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewriting *if the following is true for any query Q in $\mathcal{L}_Q$ and any set of views $\mathcal{V}$ in $\mathcal{L}_\mathcal{V}$: If $\mathcal{V}$ determines Q then there is an equivalent rewriting in $\mathcal{L}$ of Q using $\mathcal{V}$. We say that $\mathcal{L}$* is complete for rewriting *if it is complete for $\mathcal{L}$ -to-$\mathcal{L}$ rewriting.*

We say that $\mathcal{L}$ is almost complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewriting *if there exists a subset $\mathcal{L}_{Q1}$ of $\mathcal{L}_Q$ which contains almost all queries in $\mathcal{L}_Q$ such that the following holds: $\mathcal{L}$ is complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_{Q1}$ rewriting. We say that $\mathcal{L}$* is almost complete for rewriting *if it is almost complete for $\mathcal{L}$ -to-$\mathcal{L}$ rewriting.*

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. The following proposition states some easy observations.

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

Canonical Rewriting. Let D_Q be the canonical database of Q. We compute the views on D_Q and get view instance $\mathcal{V}(D_Q)$ [3,2]. 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 1. We have the query $Q : q(X, Y) : -a(X, Z_1), a(Z_1, Z_2), b(Z_2, Y)$ and views $\mathcal{V}$: $V_1 : v_1(X, Z_2) : -a(X, Z_1), a(Z_1, Z_2)$ and $V_2 : v_2(X, Y) : -b(X, Y)$. 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 $\{v_1(x, z_2), v_2(z_2, y)\}$. Thus, R_c is: $q(X, Y) : -v_1(X, Z_2), v_2(Z_2, Y)$.

The following proposition can be used when we want to show that there is no equivalent CQ rewriting of a query using a set of views.

Proposition 2. *Let Q and $\mathcal{V}$ be conjunctive query and views and R_c be the canonical rewriting. If there is a conjunctive equivalent rewriting of Q using $\mathcal{V}$ then R_c is such a rewriting.*

2.3 Cases for Which CQ Is Complete for Rewriting

Theorem 1. *CQ is complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewriting in the case where $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_Q$ are subclasses of conjunctive queries in either of the following cases:*

1. *$\mathcal{L}_Q = CQ$ and $\mathcal{L}_\mathcal{V}$ contains only queries with no nondistinguished variables.*
2. *Binary base predicates, one view in the view set, $\mathcal{L}_Q$ contains only queries with one variable and $\mathcal{L}_\mathcal{V}$ contains only queries with one non-distinguished variable.*

3 Chain and Path Queries

In this section we consider chain and path queries and views.

Definition 4. *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 simple path and the start and end nodes of this path are the arguments in the head. For example, this is a chain query: $q(X, Y) : -a(X, Z_1)$, $b(Z_1, Z_2), c(Z_2, Y)$.*

Path queries *are chain queries over a single binary relation. Path queries can be fully defined simply by the length of the path in the body (i.e., number of subgoals in the body). Hence we denote by P_k the path query of length k. We denote the language of all chain queries by CQ_{chain} and the language of all path queries by CQ_{path}.*

3.1 Chain Queries – Decidability

In the case of chain queries and views, we show that the following property fully characterizes cases where a set of views determine a query (Theorem 2), hence for this class determinacy is decidable.

Definition 5. *Let Q be a binary query over binary predicates. We say that Q is* disjoint *if the body of Q viewed as an undirected graph does not contain a (undirected) path from one head variable of Q to the other.*

Theorem 2. *Let query Q and views $\mathcal{V}$ be chain queries. Then the following hold:*

1. *$\mathcal{V}$ determines Q iff the canonical rewriting of Q using $\mathcal{V}$ is not disjoint.*
2. *First order logic is complete for CQ_{chain}-to-CQ_{chain} rewriting.*
3. *It is decidable whether a set of views determines a query.*

3.2 Path Queries – CQ Is Almost Complete for Rewriting

In this section we will prove the following theorem and we will also get a complete characterization for path queries and two path views as concerns CQ being complete for this class of queries and views.

Theorem 3. *CQ_{path} (and hence CQ) is almost complete for CQ_{path}-to-CQ_{path} rewriting. Hence CQ_{path} is almost complete for rewriting.*

The above theorem is a consequence of Lemma 2. In order to acquire some intuition we present first some intermediate results.

Theorem 4

1. *CQ_{path} (and hence CQ) is complete for $\{P_2, P_3\}$-to-CQ_{path} rewriting.*
2. *CQ_{path} (and hence CQ) is complete for $\{P_3, P_4\}$-to-CQ_{path1} rewriting, where CQ_{path1} is CQ_{path} after deleting P_5.*

*Proof (*of Part 1*).* The proof of part 1 is easy: The view set does not determine query P_1 for the following reason: Take a database which is empty and another database which contains a single tuple, then in both databases, the views compute the empty set while the query computes the empty set only in the former database. All other path queries have easy equivalent CQ_{path} rewritings. ⊣

It is interesting to note (as another counterexample that CQ is not complete for rewriting) that viewset $\{P_3, P_4\}$ determines the query P_5 because the following formula is a rewriting of $P_5(X, Y)$ (it is not a CQ however):

$$\phi(X,Y) : \exists Z[P_4(X,Z) \wedge \forall W((P_3(W,Z) \rightarrow P_4(W,Y))]$$

However there is no CQ rewriting of P_5 using $\{P_3, P_4\}$.

We generalize the result in Theorem 4 for two views P_k and P_{k+1}. The following theorem is a complete characterization of all path queries with respect to viewset $\{P_k, P_{k+1}\}$.

Theorem 5. *Let $\mathcal{QP}_{k+2}$ be the set of all path queries except the set of queries*

$$\mathcal{QPP}_{k+2} = \bigcup_{n=1}^{n=k-2} \{P_{nk+n+1}, P_{nk+n+2}, \ldots, P_{(n+1)k-1}\}$$

Then the following hold:

1. *CQ_{path} (and hence CQ) is complete for $\{P_k, P_{k+1}\}$-to-$\mathcal{QP}_{k+2}$ rewriting.*
2. *CQ is not complete for $\{P_k, P_{k+1}\}$-to-$\mathcal{QPP}_{k+2}$ rewriting.*

Proof. (Sketch) First we use Theorem 2 to prove that all path queries except queries $P_1, \ldots, P_{k-1}$ are determined by $\{P_k, P_{k+1}\}$. We only need to show that there is in the expansion of the canonical rewriting an undirected path from head variable X to head variable Y which ends in a forward edge. Inductively, for query P_m ($m \geq k$) there is such a directed path which ends in a forward edge. For query P_{m+1}, we augment the undirected path of P_m by taking a backward edge for P_k and then a forward edge for P_{k+1}.

Then we use similar argument as in the case of the viewset $\{P_2, P_3\}$ to prove that none of the queries $P_1, \ldots, P_{k-1}$ are determined by $\{P_k, P_{k+1}\}$. Finally we prove that, for each path query in $\mathcal{QP}_{k+2}$, the canonical rewriting is an equivalent rewriting. ⊣

The following theorem is a corollary of Theorem 5 and Theorem 7 generalizes for any two views P_k, P_m. The proof of Theorem 7 is a consequence of Lemma 1.

Theorem 6. *CQ_{path} (and hence CQ) is almost complete for $\{P_k, P_{k+1}\}$-to-CQ_{path} rewriting.*

Theorem 7. *Let k, m be positive integers. Then, CQ_{path} (and hence CQ) is almost complete for $\{P_k, P_m\}$-to-CQ_{path} rewriting.*

Lemma 1. *Let P_n be a query and let viewset be $\{P_k, P_m\}$. Then the following hold.*

1. *If $n \geq km$ and the greatest common divisor of k and m divides n then there is a CQ_{path} equivalent rewriting of the query using $\{P_k, P_m\}$.*
2. *If the greatest common divisor of k and m does not divide n then $\{P_k, P_m\}$ does not determine the query.*

Finally the following lemma generalizes Lemma 1 for any number of views:

Lemma 2. *Let P_n be a query and let viewset be $\mathcal{V}$=$\{P_{k_1}, P_{k_2}, \ldots, P_{k_K}\}$. Then there is a positive integer n_0 which is a function only of $k_1, k_2, \ldots, k_K$ such that for any $n \geq n_0$ the following statements are equivalent.*

1. There is no equivalent rewriting in CQ of P_n using $\mathcal{V}$.
2. The canonical rewriting of P_n using $\mathcal{V}$ is disjoint.
3. $\mathcal{V}$ does not determine P_n.

4 Determinacy and Query Equivalence

The problem that we investigate in this paper relates determinacy to query rewriting. The simplest way to produce an equivalent rewriting of a query Q is when we have only one view and the view is equivalent to the query. Hence, a natural related problem is: If Q_1 is contained in Q_2 and Q_2 determines Q_1, 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 there is a counterexample:

Example 2. 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 and 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 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. 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) we are not allowed to rename the constants $x_1, \ldots, x_m$.

Now we introduce a new problem which relates determinacy to query equivalence:

Determinacy and query equivalence: Let Q_1, Q_2 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? If the answer is "yes" for any pair of queries Q_1, Q_2 where Q_1 belongs to CQ class CQ_1 and Q_2 belongs to CQ class CQ_2, then we say that *determinacy defines CQ_2-to-CQ_1 equivalence.*

This problem seems to be easier to resolve than the determinacy problem and Theorem 8 is formal evidence of that.

Theorem 8. *Let CQ_1, CQ_2 be subsets of the set of conjunctive queries. For the following two statements it holds: Statement (A) implies statement (B).*

A) CQ is complete for CQ_2-to-CQ_1 single view rewriting.
B) Determinacy defines CQ_2-to-CQ_1 equivalence.

In [22] it is proven part A of the above theorem for one path view. A consequence of it and Theorem 8 is the following:

Theorem 9. *Determinacy defines CQ_{path}-to-CQ equivalence.*

The *determinacy and query equivalence* question remains open. Theorem 10 settles a special case where we have replaced condition (b) with a stronger one. Theorem 11 is a consequence of Theorem 10.

Theorem 10. *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 database of Q_1. Then Q_1 and Q_2 are equivalent.*

Theorem 11. *Consider queries in either of the following cases: a) Q_1 has no self joins (i.e., each predicate name appears only once in the body) or b) Q_1 contains a single variable.*

Suppose CQ query Q_2 determines Q_1 and Q_1 is contained in Q_2. Then Q_1 and Q_2 are equivalent.

5 Connectivity

In this section, we present a case where good behavior for determinacy can carry over to a broader class of queries. Specifically we relate determinacy to connectivity in the body of the query. The following example shows the intuition.

Example 3. We have query: $Q : Q(X) : -r(Y,X), s(Y,X), s_1(Z,Z_1), s_2(Z_1,Z)$ and views $\mathcal{V}$: $v_1(X,Y) : -r(Y,X)$ and $v_2(X,Y) : -s(Y,X), s_1(Z,Z_1), s_2(Z_1,Z)$. First observe that all variables contained in the last two subgoals of Q are not contained in any other subgoal of Q and neither they appear in the head of Q. In this case we say that subgoals $s_1(Z,Z_1), s_2(Z_1,Z)$ form a connected component (see definitions below). Moreover, let us consider the canonical rewriting (which happens to be an equivalent rewriting) of Q using these two views $R_1 : Q(X) : -v_1(X,Y), v_2(X,Y)$. Observe that none of the variables in the two last subgoals of the query appear in the rewriting (we conveniently retain the same names for the variables). In this case, we say in addition that the subgoals $s_1(Z,Z_1), s_2(Z_1,Z)$ form a semi-covered component wrto the views (see definition below). We conclude the observations on this example by noticing that the following query and views a) are simpler and b) can be used "instead" of the original query and views. Query $Q'(X) : -r(Y,X), s(Y,X)$ and views $\mathcal{V}$: $v'_1(X,Y) : -r(Y,X)$ and $v'_2(X,Y) : -s(Y,X)$. They were produced from the original query and views by a) deleting the semi-covered subgoals from the query and b) deleting an isomorphic copy of the semi-covered subgoals from view v_2 (see Lemma 3 for the feasibility of this). Then the canonical rewriting of Q' using $\mathcal{V}'$ is isomorphic to R_1, specifically it is: $R'_1 : Q'(X) : -v'_1(X,Y), v'_2(X,Y)$ and is again an equivalent rewriting. In this section, we make this observation formal, i.e., that in certain cases, we can reduce the original problem to a simpler one.

Definition 6 (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 query is *head-connected* if all subgoals containing head variables are contained in the same connected component.

Definition 7 (semi-covered component)**.** *Let Q and $\mathcal{V}$ be CQ query and views. Let G be a connected component of query Q. Suppose that every variable or constant 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 $\mathcal{V}$.

Lemma 3. *Let Q and $\mathcal{V}$ be conjunctive query and views. Suppose $\mathcal{V}$ determines Q. Let G_Q be a connected component of Q which is semi-covered wrto $\mathcal{V}$. Then there is a view in $\mathcal{V}$ which contains a connected component which is isomorphic to G_Q.*

As a consequence of Lemma 3, we can identify the semi-covered components of the query in the views definitions as well. Hence, we define the *semi-covered-free pair,* $(Q', \mathcal{V}')$, of a pair $(Q, \mathcal{V})$ of query and views: Q' results from Q by deleting all semi-covered components wrto $\mathcal{V}$ and each view in $\mathcal{V}'$ results from a view in $\mathcal{V}$ by deleting the components isomorphic to the semi-covered components of the query. Then the following holds:

Theorem 12. *Let CQ_1, CQ_2 be subsets of the set of conjunctive queries such that each query in either of them is head-connected. Let CQ_c be a conjunctive query language. Let CQ_{1f}, CQ_{2f} be subsets of the set of conjunctive queries such that for each query Q in CQ_1 (CQ_2 respectively) there is a query in CQ_{1f} (CQ_{2f}, respectively) which is produced from Q by deleting a connected component. Then the following holds:*

Language CQ_c is complete for CQ_1-to-CQ_2 rewriting iff it is complete for CQ_{1f}-to-CQ_{2f} rewriting.

The following is a corollary of Theorem 12 and results from Section 3:

Theorem 13. *Let P_k^a be a query with two variables in the head whose body contains i) a path on binary predicate r from one head variable to the other and ii) additional subgoals on predicates distinct from r and using variables distinct from the variables that are used to define the path. We call the language of such queries CQ_{apath}.*

Suppose we have query Q and views $\mathcal{V}$ that are in CQ_{apath}. Then it holds: CQ_{path} (and hence CQ) is almost complete for CQ_{apath}-to-CQ_{apath} rewriting.

6 Conclusion

The case about finding well behaved subclasses of conjunctive queries is of interest and is far from closed. We include some suggestions that are close to the research presented in this paper. For chain queries, we don't have a full characterization as concerns subclasses for which CQ is complete. We don't know whether determinacy defines CQ-to-CQ equivalence. Decidability of determinacy for conjunctive queries remains open.

Acknowledgments. Many thanks to Jeff Ullman for insightful discussions and for providing Example 2. Thanks also to the anonymous reviewers for their very useful comments.

References

1. Abiteboul, S., Duschka, O.M.: Complexity of answering queries using materialized views. In: PODS (1998)
2. Afrati, F., Li, C., Ullman, J.D.: Generating efficient plans using views. In: SIGMOD (2001)

3. Afrati, F., Li, C., Ullman, J.D.: Using views to generate efficient evaluation plans for queries. JCSS, to appear
4. Afrati, F.N., Li, C., Mitra, P.: Rewriting queries using views in the presence of arithmetic comparisons. Theor. Comput. Sci. 368(1-2) (2006)
5. Agrawal, S., Chaudhuri, S., Narasayya, V.R.: Automated selection of materialized views and indexes in sql databases. In: Proc. of VLDB (2000)
6. Bayardo Jr., R.J., et al.: Infosleuth: Semantic integration of information in open and dynamic environments (experience paper). In: SIGMOD (1997)
7. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Lossless regular views. In: PODS, ACM, New York (2002)
8. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: View-based query query processing: On the relationship between rewriting, answering and losslessness. In: International Conference on Database Theory (ICDT) (2005)
9. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: STOC (1977)
10. Chaudhuri, S., Krishnamurthy, R., Potamianos, S., Shim, K.: Optimizing queries with materialized views. In: ICDE (1995)
11. Chawathe, S.S., et al.: The TSIMMIS project: Integration of heterogeneous information sources. In: IPSJ (1994)
12. Chekuri, C., Rajaraman, A.: Conjunctive query containment revisited. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, Springer, Heidelberg (1996)
13. Duschka, O.M., Genesereth, M.R.: Answering recursive queries using views. In: PODS (1997)
14. Florescu, D., Levy, A., Suciu, D., Yagoub, K.: Optimization of run-time management of data intensive web-sites. In: Proc. of VLDB (1999)
15. Grumbach, S., Tininini, L.: On the content of materialized aggregate views. In: PODS (2000)
16. Haas, L.M., Kossmann, D., Wimmers, E.L., Yang, J.: Optimizing queries across diverse data sources. In: Proc. of VLDB (1997)
17. Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal 10(4)
18. Ives, Z., Florescu, D., Friedman, M., Levy, A., Weld, D.: An adaptive query execution engine for data integration. In: SIGMOD (1999)
19. Levy, A., Mendelzon, A., Sagiv, Y., Srivastava, D.: Answering queries using views. In: PODS (1995)
20. Levy, A., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: Proc. of VLDB (1996)
21. Li, C., Bawa, M., Ullman, J.: Minimizing view sets without losing query-answering power. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, Springer, Heidelberg (2000)
22. Nash, A., Segoufin, L., Vianu, V.: Determinacy and rewriting of conjunctive queries using views: A progress report. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, Springer, Heidelberg (2006)
23. Rajaraman, A., Sagiv, Y., Ullman, J.D.: Answering queries using templates with binding patterns. In: PODS (1995)
24. Segoufin, L., Vianu, V.: Views and queries: Determinacy and rewriting. In: PODS, ACM Press, New York (2005)
25. Theodoratos, D., Sellis, T.: Data warehouse configuration. In: Proc. of VLDB (1997)
26. Ullman, J.D.: Information integration using logical views. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, Springer, Heidelberg (1996)