Theoretical Computer Science 412 (2011) 1005–1021

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Determinacy and query rewriting for conjunctive queries and views

Foto N. Afrati [*]

National Technical University of Athens, GR-157 73, Greece

ARTICLE INFO

Article history:
Received 16 April 2008
Received in revised form 26 October 2010
Accepted 12 December 2010
Communicated by D. Sannella

Keywords:
Determinacy
Views
Query rewriting
Query answering
Conjunctive queries
Chain queries
Path queries
First order logic

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 the 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 [22] has received considerable attention because of its relevance to many data-management applications, such as information integration [8,13,18,20,23,29], data warehousing [28,7], web-site design [16], and query optimization [12]. 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 [19].

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 $Q(D_1) = Q(D_2)$ [27]. 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 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.

* Tel.: +30 2102232097; fax: +30 2107722499.
E-mail addresses: afrati@softlab.ntua.gr, afrati@gmail.com.

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.031

The existence of rewritings depends 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 [25]. However there exist special cases for CQ queries and views where CQ is complete for rewriting [27,25].

In this paper we consider subclasses of CQs and investigate (a) decidability of determinacy, (b) special cases where CQ or first order logic (FO) 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. For chain queries and views, we show that FO is complete for rewriting.
4. We introduce the notion of a language being *almost* complete for rewriting.
5. 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.
6. Query rewriting 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 and queries Q_1, Q_2 in the language, if Q_1 is contained in Q_2 and the view set $\{Q_2\}$ 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.
7. 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.

Preliminary results of this paper appeared in [2].

1.1. Related work

In [27], the problem of determinacy is introduced and investigated for many languages including first order logic and fragments of second order logic. A considerable number of cases are resolved. A follow-up paper [25] continues in the same line of research. We briefly summarize the results in these two papers. FO (i.e., first order logic) is not complete for FO-to-FO rewriting; in fact, any language complete for FO-to-FO rewriting must express all computable queries. FO is not complete for ∃FO-to-FO rewriting but both ∃SO and ∀SO are complete for such rewriting. Datalog≠ is not complete for UCQ-to-UCQ rewriting (UCQ is short for the language of finite unions of conjunctive queries). This also holds for CQ≠-to-CQ rewriting. No monotonic language is complete for CQ-to-CQ rewriting, e.g., CQ≠ or UCQ or Datalog≠ will not do. Determinacy is undecidable for UCQ views and queries. Special classes of conjunctive queries and views are identified for which the language of conjunctive queries is complete and hence determinacy is decidable: (a) for Boolean views, (b) for monadic views, and (c) for single path-view.

In [21] a language is identified which is complete for rewriting for views and queries in the same language, it is the packed fragment of FO (PackedFO). PackedFO is a generalization of the guarded fragment of FO. The guarded CQs are exactly the acyclic CQs.

Determinacy and notions related to it are investigated in [9,10,17]. In [17] the notion of subsumption is introduced and used for the definition of complete rewritings. In [9,10] the concept of lossless view with respect to a query is introduced and investigated for regular path queries, both under the sound view assumption and under the exact view assumption. Losslessness under the exact view assumption is identical to determinacy.

There is a large amount of work on equivalent rewritings of queries using views. It includes [22], 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, and [14] where polynomial subcases were identified. In [26,5,15] 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, finding maximally contained rewritings is the best we can do given a rewriting language when there is no equivalent rewriting in the language, hence we want to obtain a rewriting (that uses only the views) that computes as many certain answers [1] as possible. In [24] 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 [9] and references therein. Answering queries using views for XPath query and view is investigated in [6].

2. Preliminaries and cases where CQ is complete

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 the predicate g_i defines a *base relation* (we use the same symbol for the predicate and the relation), and $\bar{X}_i$ is a vector where each component 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 also be viewed as a relational structure which we call a *canonical database of query Q* and denote by 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 [11] 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 containment mapping from Q_2 to Q_1 is defined by giving a mapping μ from the variables of Q_2 to the variables of Q_1. An example follows.

Example 1. The two queries are:

$$Q_1 : q_1(X', Y') : -a(X', Z_1'), b(Z_1', Z_1'), c(Z_1', Y'), a(X', Y')$$
$$Q_2 : q_2(X, Y) : -a(X, Z_1), b(Z_1, Z_2), c(Z_2, Y).$$

The containment mapping is given by the following mapping μ:

$$\{X \to X', Y \to Y', Z_1 \to Z_1', Z_2 \to Z_1'\}.$$

It is a containment mapping because the list of variables in the head of Q_2 is mapped to the list of variables to the head of Q_1 and each subgoal of Q_2 is mapped on a subgoal of Q_1 with the same predicate, i.e., $a(X, Z_1)$ is mapped to $a(X', Z_1')$, $b(Z_1, Z_2)$ is mapped to $b(Z_1', Z_1')$ and $c(Z_2, Y)$ is mapped to $c(Z_1', Y')$. This containment mapping proves that query Q_1 is contained in query Q_2. Notice that the subgoal $a(X', Y')$ of Q_1 is not a target of a subgoal of Q_2, i.e., it is not used to verify the containment mapping μ, or, in other words, it can be deleted from Q_1 and still the containment mapping will be valid.

A containment mapping from Q_2 to Q_1 induces several *subgoals mappings*, since, by definition, each subgoal of Q_2 must be mapped on a subgoal of Q_1. In the above example, the subgoal mapping is unique. A containment mapping from Q_2 to Q_1 is called *subgoals-onto* if there is an induced subgoal mapping such that each subgoal of Q_1 is a target of some subgoal of Q_2. The containment mapping in the above example is not subgoals-onto because subgoal $a(X', Y')$ of Q_1 is not a target.

A CQ query Q is *minimized* if by deleting any subgoal we obtain a query which is not equivalent to Q. We assume that all queries are minimized in the proofs here. 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 unfolding the view definitions, taking care to avoid unwanted variable bindings by renaming. I.e., existentially quantified variables (i.e., nondistinguished variables) in view definitions are renamed to 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*)**.** Suppose we have 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. 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 a 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 *almost complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewriting* if for every view set $\mathcal{V}_1$ from $\mathcal{L}_\mathcal{V}$ 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 $\mathcal{V}_1$-to-$\mathcal{L}_{Q1}$ 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 determines 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 a 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) *either $\mathcal{V}(D_Q) \neq \emptyset$ or $Q(D) = \emptyset$ for all D.* (e) *Let $\mathcal{V}' \subseteq \mathcal{V}$ contain exactly those views that use in their definitions only predicates that are used also in the definition of the query Q. Then $\mathcal{V}'$ determines Q.*

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 $Q(D_1) \neq Q(D_2)$, which is a contradiction.

To prove the third part, suppose p_i is the predicate name which appears in the query but does not appear in the view definitions. 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 argument, now D' is empty. Then again, on both D_Q and D' the views compute the same relations but the query does not.

Proof of (e). Suppose $\mathcal{V}$ determines Q. Let D_1 and D_2 be two databases such that $\mathcal{V}'(D_1) = \mathcal{V}'(D_2)$. We delete from both D_1 and D_2 all facts over predicates that are not used in Q, thus obtaining D_1' and D_2'. Clearly $Q(D_1') = Q(D_1)$ and $Q(D_2') = Q(D_2)$. Also, since views in $\mathcal{V}'$ do not use in their definition any of the predicates whose facts are deleted, we have $\mathcal{V}'(D_1) = \mathcal{V}'(D_2) = \mathcal{V}'(D_1') = \mathcal{V}'(D_2')$. Call $\mathcal{V}''$ the subset of the views not in $\mathcal{V}'$. Then $\mathcal{V}''(D_1') = \mathcal{V}''(D_2') = \emptyset$. Hence, $\mathcal{V}(D_1') = \mathcal{V}(D_2')$. Hence $Q(D_1') = Q(D_2')$. Therefore $Q(D_1) = Q(D_2)$. □

Canonical rewriting. Let D_Q be the canonical database of Q. We compute the views on D_Q and get view instance $\mathcal{V}(D_Q)$ [4,3]. 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 2. 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 a conjunctive query and views and R_c be the canonical rewriting. Then the following holds: If there is a conjunctive query that is an equivalent rewriting of Q using $\mathcal{V}$ then R_c is such a rewriting.*

Proof. First observe that by construction of R_c there is a containment mapping from its expansion R_c^{exp} to Q. We will show that there is also a containment mapping from Q to R_c^{exp}. 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. D_Q (the canonical database of Q) is isomorphic to the body of Q. Hence, there is a homomorphism from R^{exp} to Q iff there is a homomorphism from R^{exp} to D_Q. Therefore, there is a homomorphism μ from R^{exp} to D_Q. This establishes that there is a homomorphism from R to R_c (by construction of R_c), hence there is a homomorphism from R^{exp} to R_c^{exp}. Since R is a rewriting equivalent to Q, there is a containment mapping from Q to R^{exp}. Hence there is a containment mapping from Q to R_c^{exp}. □

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, where $\mathcal{L}_{\mathcal{V}}$ and $\mathcal{L}_Q$ are subclasses of conjunctive queries in either of the following two cases:*

1. *$\mathcal{L}_Q = CQ$ and $\mathcal{L}_{\mathcal{V}}$ contains only full queries, i.e., $\mathcal{L}_{\mathcal{V}} = CQ_{full}$.*
2. *The following three conditions are satisfied:*
   - *We have one binary view in the view set.*
   - *$\mathcal{L}_Q$ contains only queries with one variable defined over binary base predicates, i.e., it is of the form $q(X) : -a_1(X, X), a_2(X, X), \ldots, a_n(X, X)$, where $a_1, \ldots, a_n$ are binary predicates (not necessarily distinct).*
   - *$\mathcal{L}_{\mathcal{V}}$ contains only binary queries with one non-distinguished variable, i.e., queries of the form $q(X, Y) : -b_1(\overline{X}_1), \ldots, b_m(\overline{X}_m)$, where $b_i, i = 1, \ldots, m$ are predicates (not necessarily distinct) and $\overline{X}_i, i = 1, \ldots, m$ is a vector (of any length) over variables X, Y, Z (i.e., Z is the only non-distinguished variable in the view definition).*

Proof. In the proofs below we assume that views and query are minimized. Case 1. (Full views) We will show that the canonical rewriting R_c is an equivalent rewriting. Since there are no nondistinguished variables in the 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 also defines a one-to-one containment mapping μ from R_c^{exp} to Q (this is possible because the views are full). Suppose μ is not subgoals-onto, hence there is a subgoal g of Q which is not a target of μ. Then consider Q' which is Q with g deleted. Because of μ and the construction of R_c (which included all view tuples in $\mathcal{V}(D_Q)$), we have $\mathcal{V}(D_Q) = \mathcal{V}(D_{Q'})$. This contradicts clause (a) of Proposition 1. Since μ is one-to-one and subgoals-onto, μ^{-1} is a containment mapping from Q to R_c^{exp}.

Case 2. (Query contains only one variable and is over binary predicates, single binary view with only one non-distinguished variable) Let query Q and view V be as in the statement of the theorem. According to Proposition 1(e), we may assume that either V does not determine Q or V is defined over binary predicates that are also used in the query definition. To see this, suppose not. Then take $\mathcal{V}'$ as in Proposition 1(e). In this case $\mathcal{V}'$ is empty. Hence, if D_1 is the canonical database D_Q and D_2 is empty, $\mathcal{V}'(D_1) = \mathcal{V}'(D_2) = \emptyset$, but $Q(D_1) \neq Q(D_2)$, which contradicts Proposition 1(e).

According to Proposition 1(d), $V(D_Q) \neq \emptyset$ (otherwise, $Q(D) = \emptyset$ for all D, hence Q can be trivially rewritten using any view set). Since D_Q contains only one constant, $V(D_Q)$ contains exactly one tuple $v(A, A)$. Hence the canonical rewriting is $R_c : q'(A) : -v(A, A)$. We will show that the expansion R_c^{exp} of R_c is equivalent to the query Q. It is clear that Q is contained in R_c^{exp} (by definition of R_c). We will prove in the rest of the proof that R_c^{exp} is contained in Q.

Suppose there is no containment mapping from Q to R_c^{exp}. Then, there is a base relation atom $r(X, X)$ in the body of the query such that there is no atom $r(A, A)$ in the body of R_c^{exp}—because otherwise a containment mapping would exist. Assuming the existence of such an atom $r(X, X)$ in Q, we will construct in the following two databases D_1, D_2 such that $V(D_1) = V(D_2)$ and $Q(D_1) \neq Q(D_2)$, thus arriving at a contradiction.

First we will construct five intermediate databases with convenient properties. We construct a database D over two constants c, d as follows. Let the distinguished variables of the view V be called X, Y and the non-distinguished variable be called Z. To construct D, take the body of the view definition and equate variable Z to Y. Then rename X to c and Y to d and add in D exactly all the facts generated by this procedure. For example, if the definition of V is: $v(X, Y) : -p_1(X, Z), p_2(X, X), p_3(X, Y), p_4(Z, Y)$ then D contains the facts $p_1(c, d), p_2(c, c), p_3(c, d)$ and $p_4(d, d)$. By construction and by assumption that there is no atom $r(A, A)$ in the body of R_c^{exp}, database instance D has the property: $V(D)$ contains tuple (c, d) and there is no tuple $r(c, c)$ in D. In a symmetric way, we construct database D' over two constants c', d' as follows. Take the body of the view definition and equate variable Z to X. Then rename Y to c' and X to d' and add in D' exactly all the facts generated by this procedure. For example, if the definition of V is as above then D' contains the facts $p_1(d', d'), p_2(d', d'), p_3(d', c')$ and $p_4(d', c')$. By construction and by assumption that there is no atom $r(A, A)$ in the body of R_c^{exp}, database instance D' has the property: $V(D')$ contains tuple (d', c') and there is no tuple $r(c', c')$ in D'.

The other three intermediate databases are constructed now. (i) D_0 is the database over one constant (unique up to isomorphism) which is a homomorphic image of the body of the view definition. Let a_0 be its single constant. I.e., D_0 is constructed by equating all three variables X, Y, Z of the view definition to a_0 and adding in D_0 all atoms in the body. (ii) D_Q is the canonical database of the query; let a_1 be its single constant. (iii) Finally we construct D_c^{exp}: 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). Note that A is the distinguished variable of R_c^{exp}.

Now we are ready to construct the two databases D_1 and D_2 using the five above constructed databases D, D', D_0, D_Q and D_c^{exp} as follows. D_1 is constructed by taking the union of D_c^{exp}, D, D' and D_0 and then by (i) equating c and c' to a and (ii) equating d, d' and a_0 to b (remember that a and b are the two constants of D_c^{exp}). D_2 is constructed by taking the union of D_1 with D_Q and by equating a_1 to a.

Now both $V(D_1)$ and $V(D_2)$ contain all four possible tuples, namely tuple (a, b) because tuple (c, d) was in $V(D)$, tuple (b, a) because tuple (d', c') was in $V(D')$ and tuples (a, a) and (b, b) because of D_0 and D_c^{exp}. Thus $V(D_1) = V(D_2)$. However, $Q(D_2)$ contains (a, a) and $Q(D_1)$ does not (because $r(a, a)$ is not a tuple of D_1 by construction), hence $Q(D_1) \neq Q(D_2)$, thus arriving at a contradiction. □

3. Chain and path queries

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

Definition 4. Given a set $\mathcal{P}$ of binary predicate names and a positive integer n, a *chain query* is defined to be a conjunctive query of the form

$$q(X_1, X_{n+1}) : -a_1(X_1, X_2), a_2(X_2, X_3), \ldots, a_i(X_i, X_{i+1}), a_{i+1}(X_{i+1}, X_{i+2}), \ldots, a_{n-1}(X_{n-1}, X_n), a_n(X_n, X_{n+1}).$$

where $a_i \in \mathcal{P}$, $i = 1, 2, \ldots, n$ and X_i, $i = 1, 2, \ldots, n + 1$ are all distinct variables.

Path queries are chain queries over a single binary relation. We denote the language of all chain queries by CQ_{chain} and the language of all path queries by CQ_{path}.

Observe that the body of a chain query contains as subgoals a number of binary atoms which, if viewed as a labeled graph (since they are binary), form a simple directed path where the start and end nodes of this path are the arguments in the head. For example, the following is a chain query: $q(X, Y) : - a(X, Z_1), b(Z_1, Z_2), c(Z_2, Y)$, whereas the following two are *not* chain queries: $q'(X, Z_2) : -a(X, Z_1), b(Z_1, Z_2), c(Z_2, Y)$ and $q''(X, Y) : -a(X, Z_1), b(Z_1, Z_1), c(Z_1, Y)$. Query q' is not a chain query because the head variable Z_2 is not the end variable of the simple path in the body. Query q'' is not a chain query because the body is not a *simple* path from X to Y (there is a cycle from Z_1 to Z_1). A chain query can be fully described by a list of the subgoal predicate names, e.g., the above chain query is query $q : -abc$. In the following sections, sometimes we use this notation for chain queries since it is simpler and easier to read.

The above are not path queries because they use three predicate names, a, b, c. The following is a path query: $q_p(X, Y) : -b(X, Z_1), b(Z_1, Z_2), b(Z_2, Y)$. 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. Thus q_p above is a path query of length 3 and we refer to it by P_3.

In the rest of this section we show that determinacy is decidable for CQ_{chain} views and CQ_{chain} queries and we show that FO is complete for CQ_{chain}-to-CQ_{chain} rewriting (in Section 3.1). Then, in Section 3.2, we observe that although CQ is *not* complete for CQ_{path}-to-CQ_{path} rewriting, it only misses by a finite number of path queries for each view set. This observation justifies the introduction of the new notion of a language being *almost complete for rewriting* as in Definition 3. We prove that CQ is almost complete for CQ_{path}-to-CQ_{path} rewriting.

Before we continue, we give some useful definitions. Since we are going to use these definitions in other sections as well (where more general queries than chain queries are treated) we give them here in their most general form.

Definition 5 (*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.

Observe that, for chain queries, the connectivity graph is a simple path.

3.1. Chain queries — decidability

In this subsection, we show that the property below can be used to fully characterize cases where a set of views determine a query in the case we have views and a query which are chain queries. Hence, for this class determinacy is decidable.

Definition 6. Let Q be a binary query over binary predicates and having head variables X and Y. We say that Q is *disjoint* if, in its connectivity graph, there is no connected component which contains one subgoal that contains X and one subgoal that contains Y.

For the case of queries over binary predicates, it can be easily shown that the following is an equivalent definition of Definition 6. Hence for the rest of this section we use Definition 7.

Definition 7. Let Q be a binary query with head variables X and Y. 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 to the other.

Theorem 2 below is the main result of this subsection.

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.*

Theorem 2 is an immediate consequence of Lemmas 1 and 2.

Lemma 1. *Let query Q and views $\mathcal{V}$ be CQ chain queries. If the canonical rewriting of Q using $\mathcal{V}$ is disjoint then $\mathcal{V}$ does not determine Q.*

Proof. Suppose the canonical rewriting of Q using $\mathcal{V}$ is disjoint. We will construct two databases D' and D'' such that $\mathcal{V}(D') = \mathcal{V}(D'')$ and $Q(D') \neq Q(D'')$. We will first construct databases D_Q, D_1, D_2 and D_3 each of which is isomorphic to D_Q. Then we construct (a) D' to be the union of D_Q and D_1 and (b) D'' to be the union of D_2 and D_3.

Database D_Q is the canonical database of the query Q. Database D_1 is a copy of D_Q where all constants are replaced by fresh constants. For ease of reference, we assume that each constant of D_Q is replaced in D_1 by its primed version, i.e., c in D_Q is replaced in D_1 by c'.

We construct database D_2 as follows: Let G' be one of the connected components of R_c^{exp} (i.e., of the undirected graph of the body of R_c^{exp}) which contains the first head variable of R_c. By definition of R_c, there is a one-to-one homomorphism from the variables of R_c to the variables of Q such that it can be extended to a homomorphism h from R_c^{exp} to Q. Let h' be the restriction of h to the variables of G'. Let $Var(G')$ be the variables in the subgoals of Q that are targets of h'. Let G'' be the rest of the graph of the body of R_c^{exp}. Let h'' be the restriction of h to the variables of G''. Let $Var(G'')$ be the variables in the subgoals of Q that are targets of h''. By definition, $Var(G')$ and $Var(G'')$ are disjoint sets because G' and G'' are disjoint. Now

D_2 is a copy of D_Q which uses the corresponding constants as in D_Q except that for constants that are targets of h' (we may view h' as a homomorphism on D_Q too) we use their primed version (i.e., those constants are the same as the corresponding constants in D_1). Finally D_3 is the "reverse image" of D_2, i.e., it is a copy of D_2 where we have changed the primed constants to their non-primed version and the non-primed constants to their primed version.

We claim that $\mathcal{V}(D') = \mathcal{V}(D'')$. First observe that $\mathcal{V}(D_Q)$, $\mathcal{V}(D_i)$, $i = 1, 2, 3$ are pairwise isomorphic. Also $\mathcal{V}(D_Q)$ and $\mathcal{V}(D_1)$ are disjoint sets. Thus $\mathcal{V}(D') = \mathcal{V}(D_Q) \cup \mathcal{V}(D_1)$ consists of two copies of $\mathcal{V}(D_Q)$, one copy with facts over non-primed constants and one copy with facts over primed constants. Also $\mathcal{V}(D'') = \mathcal{V}(D_2) \cup \mathcal{V}(D_3)$. Again $\mathcal{V}(D_2)$ and $\mathcal{V}(D_3)$ are disjoint sets; because, by construction and by the observation about being isomorphic, if there is a $v_i(a, b)$ in $\mathcal{V}(D_2)$ then there is a $v_i(c, d)$ in $\mathcal{V}(D_3)$ such that c is the primed version of a, if a is non-primed or c is the non-primed version of a if a is primed, and similarly for b and d. It suffices to prove that neither $\mathcal{V}(D_2)$ nor $\mathcal{V}(D_3)$ contain any facts that use one primed and one non-primed constant. Suppose there was such a fact $v_i(c, c_1')$ in $\mathcal{V}(D_2)$. Then G' is not a connected component of the connectivity graph of R_c because it is not maximal, namely, we can add the view tuple corresponding to $v_i(c, c_1')$ and still be connected. We argue similarly if $v_i(c, c_1')$ in $\mathcal{V}(D_3)$. Thus $\mathcal{V}(D') = \mathcal{V}(D'')$. However $Q(D') \neq Q(D'')$ because (a) $Q(D')$ contains the tuple (a, b) (where a is the constant that replaces in D_Q the first head variable, X, of Q and b is the constant that replaces the second head variable, Y, of Q) and (b) $Q(D'')$ does not contain (a, b). The reason that $Q(D'')$ does not contain (a, b) is that, since G' and G'' are disjoint and one head variable is contained in G' and the other in G'', a appears in D_2 and b appears in D_3. However, Q is a chain query, hence connected, therefore there is no homomorphism from Q to D'' which will evaluate (a, b). □

Lemma 2. *Let query Q and views $\mathcal{V}$ be chain queries. Suppose the canonical rewriting of Q using $\mathcal{V}$ is not disjoint. Then there is an FO equivalent rewriting of Q using $\mathcal{V}$, hence $\mathcal{V}$ determines Q.*

Proof. We will prove that there exists a first order logic formula which is an equivalent rewriting of the query using the views. We will construct this formula inductively. As we will explain in detail in the following paragraphs, we use induction on the number of alternations between forward and backward edges along a path from X to Y where X and Y are the head variables in the canonical rewriting.

We define formally the *alternations* by defining the *alternation points* which mark an alternation. Let p be a (undirected) path from X to Y on the graph G_R which corresponds to the body of the canonical rewriting R. Let $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ be nodes of G_R (i.e., variables of R) on the path p such that the part of path p from X to a_1 corresponds to a maximal forward path starting at X in G_R (this means there are subgoals $q_{i_1}(X, z_1), q_{i_2}(z_1, z_2), \ldots, q_{i_s}(z_{s-1}, a_1)$ in R) and the part of path p from a_1 to b_1 corresponds to a maximal backward path starting at a_1 in G_R (this means there are subgoals $q_{j_1}(b_1, z_1), q_{j_2}(z_1, z_2), \ldots, q_{j_l}(z_{l-1}, a_1)$ in R) and the part of path p from b_1 to a_2 corresponds to a maximal forward path starting at b_1 in G_R, and the part of path p from a_2 to b_2 corresponds to a maximal backward path starting at a_2 in G_R and so on, up until the part of path p from b_n to Y corresponds to a maximal forward path starting at b_n in G_R (the last part is always a forward path). Then $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ are the alternation points of p.

We consider the graph G_Q which is the simple path defined by the body of query Q from one distinguished variable (X) to the other (Y). For ease of reference, we keep node names on G_Q the same as the variable names in the body of Q. The canonical rewriting is not disjoint, hence there is an undirected path p in G_R from X to Y. We consider undirected path p in G_R and we define the alternation points $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ as above. By the definition of R_c, there is a one-to-one homomorphism from the variables of R_c to the variables of Q such that it can be extended to a homomorphism μ from R_c^{exp} to Q. For ease of reference, we refer herein to $\mu(a_i)$ as a_i and $\mu(b_i)$ as b_i respectively; the meaning will be clear from the context.

The following is an example and will be our running example for this proof.

Example 3. Consider the chain query $Q : -c_1c_2bcdef$ and the seven chain views $\mathcal{V}$: $v_1 : -c_1c_2bcd$, $v_2 : -bcd$, $v_3 : -bc$, $v_4 : -c$, $v_5 : -cde$, $v_6 : -c_2bcde$ and $v_7 : -c_1bcdef$.

Then D_Q is a simple path with labels c_1c_2bcdef and suppose its nodes are $1, 2, 3, 4, 5, 6, 7, 8$, e.g., the edge from node 1 to node 2 in D_Q is labeled a_1, and the edge from node 6 to node 7 is labeled e. Then $\mathcal{V}(D_Q) = \{v_1(1, 6), v_2(3, 6), v_3(3, 5), v_4(4, 5), v_5(4, 7), v_6(2, 7), v_7(2, 8), \}$. Hence, G_R has the edges: $\{(1, 6), (3, 6), (3, 5), (4, 5), (4, 7), (2, 7), (2, 8), \}$. Then the alternation points are $a_1 = 6, b_1 = 3, a_2 = 5, b_2 = 4, a_3 = 7, b_3 = 2$. And a path p from X to Y in G_R is $p = 1, 6, 3, 5, 4, 7, 2, 8$.

In order to keep things simple, in this example there is only one path and also each forward or backward sub-path contains only one view; in general there may exist several paths from X to Y and we pick one arbitrarily; also each of the sub-paths may contain several views.

Claim. *Consider path p as described above and its alternation points $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$. We say that a point d_1 is closer to X than point d_2 if d_1 is closer to X than d_2 on the simple path G_Q. Then, there exists i such that either of the following happens: (We assume that $X = b_0$ and $Y = a_{n+1}$)*

1. *For the alternation points $b_{i-1}, a_i, b_i, a_{i+1}$, the following holds: b_{i-1} is closer to X than b_i, and a_i is closer to X than a_{i+1}.*
2. *For the alternation points $a_i, b_i, a_{i+1}, b_{i+1}$, the following holds: a_{i+1} is closer to X than a_i, and b_{i+1} is closer to X than b_i.*

Proof of Claim. It is done by contradiction. Suppose there is no such i as in the statement of the claim. Then the following happens: a_2 is closer to X than a_1 because, otherwise, condition 1 of the Claim holds (notice that by construction we have that b_1 is closer to X than a_1). b_1 is closer to X than b_2 because, otherwise, condition 2 of the Claim holds. Thus both a_2 and b_2 are closer to X than a_1 and farther from X than b_1.

Exactly the same argument shows that both a_3 and b_3 are closer to X than a_2 and farther from X than b_2 and so on, and, for any n, a_{n+1} is closer to X than a_n. This leads to a contradiction, since we know that $Y = a_{n+1}$ is not closer to X than a_n. □

Example 4. We continue from Example 3. In this example, there is a i as in the Claim that satisfies the first clause of the claim and it is $i = 2$. In more detail: In this example we have: $a_1 = 6$, $b_1 = 3$, $a_2 = 5$, $b_2 = 4$, $a_3 = 7$ and $b_3 = 2$. Indeed, $b_{i-1} = b_1 = 3$ is closer to node 1 than $b_i = b_2 = 4$ and $a_i = a_2 = 5$ is closer to node 1 than $a_{i+1} = a_3 = 7$.

Instead of the Claim above we can use the following Alternative Claim and again prove our result in a similar way.

Before we state the alternative claim, we need a definition: Along the path p let b_i be the left-most alternation point. Along the sub-path of p from the beginning of p to b_i, let a_j be the right-most alternation point. An example of these two points follows.

Example 5. Continued from Example 3. In this example, along the path $p = 1, 6, 3, 5, 4, 7, 2, 8$, b_i, as defined above, is node 2, and a_j, as defined above, is node 7.

Alternative Claim. *Let b_i and a_j be the points defined as above. Then the following three chain queries each have the property that its canonical rewriting with respect to viewset $\mathcal{V}$ is not disjoint:*

- *The chain query from X (the first head variable) to a_j.*
- *The chain query from a_j to b_i.*
- *The chain query from b_i to Y (the second head variable)*

To complete the proof we prove first the following lemma:

Lemma 3. *Suppose chain conjunctive query $Q(X, Y)$ and chain views $\mathcal{V}$. Suppose there is a path p in the graph of the canonical rewriting with two alternation points, a_1 and b_1. Then, the following first order formula is an equivalent rewriting of Q:*

$$\phi(X, Y) : \exists Z[R_2(X, Z) \wedge \forall W((R_3(W, Z) \rightarrow R_4(W, Y))]$$

where R_2, R_3 and R_4 are chain queries over view subgoals that represent the paths from X to a_1, from b_1 to a_1 and from b_1 to Y respectively.

Proof. Before we proceed with the proof, we make more specific the definitions of R_2, R_3, R_4. I.e., $R_2(W, Z)$ will be as

$$\exists c_1, c_2, \ldots, c_{j-1}[v_{i_1}(W, c_1) \wedge v_{i_2}(c_1, c_2) \wedge \cdots \wedge v_{i_j}(c_{j-1}, Z)]$$

where $v_{i_1}(W, c_1), v_{i_2}(c_1, c_2), \ldots, v_{i_j}(c_{j-1}, Z)$ is isomorphic to the subgoals in the body of R that describe the path from X to a_1.

Similarly, $R_3(W, Z)$ will be as

$$\exists c_1, c_2, \ldots, c_{j'-1}[v_{i_1}(W, c_1) \wedge v_{i_2}(c_1, c_2) \wedge \cdots \wedge v_{i'_j}(c_{j'-1}, Z)]$$

where $v_{i_1}(W, c_1), v_{i_2}(c_1, c_2), \ldots, v_{i'_j}(c_{j'-1}, Z)$ is isomorphic to the subgoals in the body of R that describe the path from b_1 to a_1.

And finally, $R_4(W, A)$ will be as

$$\exists c_1, c_2, \ldots, c_{j''-1}[v_{i_1}(W, c_1) \wedge v_{i_2}(c_1, c_2) \wedge \cdots \wedge v_{i_{j''}}(c_{j''-1}, Z)]$$

where $v_{i_1}(W, c_1), v_{i_2}(c_1, c_2), \ldots, v_{i_{j''}}(c_{j''-1}, Z)$ is isomorphic to the subgoals in the body of R that describe the path from b_1 to a_2.

For the one direction, suppose $(a, b) \in Q(D)$ for database D. Then by definition of Q, there exists variable a_1 in the body of Q and a homomorphism h from the variables of Q to the constants of D such that $(h(a), h(a_1)) \in R_2(D)$. Consider the formula ϕ and let $Z = h(a_1)$. Indeed $R_2(h(a), h(a_1))$ holds on $\mathcal{V}(D)$ as is requited by ϕ. Let $W = w$ and suppose $R_3(w, h(a_1))$ holds on $\mathcal{V}(D)$. By definition of R_2 and Q, $R_4(h(a_1), h(Y))$ holds on $\mathcal{V}(D)$, hence $R_4(h(a_1), b)$ holds on $\mathcal{V}(D)$. Hence $\phi(a, b)$ holds on $\mathcal{V}(D)$.

For the other direction, suppose $\phi(a, b)$ holds on $\mathcal{V}(D)$ and suppose the witness for Z is c. Then, by definition of ϕ, $R_2(a, c)$ holds on $\mathcal{V}(D)$. By definition of R_2, there is a w such that $(a, w) \in Q_a(D)$ where Q_a is a chain query that reads the directed path from a to b_1 on Q. Hence $R_3(w, c)$ holds on $\mathcal{V}(D)$. Hence, according to ϕ, $R_4(w, b)$ holds on $\mathcal{V}(D)$. Hence $(w, b) \in R_4^{exp}(D)$. Since $(a, w) \in Q_a(D)$ and $(w, b) \in R_4^{exp}(D)$, we deduce that $(a, b) \in Q(D)$. □

Example 6. We continue from Example 4. According to Lemma 3, we can consider the chain query $Q' : -bcde$ and produce the following equivalent rewriting of Q' using the views v_3, v_4 and v_5:

$$\phi(X, Y) : \exists Z[v_3(X, Z) \wedge \forall W((v_4(W, Z) \rightarrow v_5(W, Y))].$$

Now the rest of the proof is almost a straightforward consequence of the claim and Lemma 3. It is proven inductively on the number of alternation points, namely, we prove the following inductive hypothesis:

Inductive hypothesis: Suppose chain conjunctive query $Q(X,Y)$ and chain views $\mathcal{V}$. Suppose there is a path p in the graph of the canonical rewriting with n alternation points. Then, there is a FO rewriting equivalent to Q.

Base case: This comes from Lemma 3.

Inductive case: Suppose the following holds: For any chain conjunctive query $Q_1(X,Y)$ and chain views $\mathcal{V}_1$, if there is a path p in the graph of the canonical rewriting with k alternation points then there is an FO rewriting equivalent to Q_1. Suppose query Q and views $\mathcal{V}$ such that there is a path p in the graph of the canonical rewriting with $k+1$ alternation points. Then, according to the Claim above, there is a j such that one of the clause of the Claim holds. Thus we have two cases (actually the argument for the two cases are very similar, but we have them separately here for clarity):

If clause 1 holds for j, then we consider the query Q'' which is the chain conjunctive query with body isomorphic to the path from b_{i-1} to a_{i+1}. According to Lemma 3, there is an equivalent FO rewriting of Q'' using $\mathcal{V}$. Now we construct view set $\mathcal{V}_1$ which is $\mathcal{V} \cup \{Q''\}$. For query Q and view set $\mathcal{V}_1$, it can be easily shown that there is a path with $2(i-1)$ alternation points. Thus, by the inductive hypothesis, we can construct an equivalent FO rewriting ϕ_1 of Q using the view set $\mathcal{V}_1$. In order to obtain an equivalent rewriting of Q using the original view set $\mathcal{V}$, we replace in ϕ_1 the query Q'' by its equivalent FO rewriting which uses $\mathcal{V}$.

If clause 2 holds, then we consider the query Q'' which is the chain conjunctive query with body isomorphic to the path from a_i to b_{i+1}. According to Lemma 3, there is an equivalent FO rewriting of Q'' using $\mathcal{V}$. Now we construct view set $\mathcal{V}_1$ which is $\mathcal{V} \cup \{Q''\}$. For query Q and view set $\mathcal{V}_1$, it can be easily shown that there is a path with $2(i-1)$ alternation points. Thus, by the inductive hypothesis, we can construct an equivalent FO rewriting ϕ_1 of Q using the view set $\mathcal{V}_1$. In order to obtain an equivalent rewriting of Q using the original view set $\mathcal{V}$, we replace in ϕ_1 the query Q'' by its equivalent FO rewriting which uses $\mathcal{V}$.

Example 7. We continue from Example 6. We consider now a new view set $\mathcal{V}'$ which is $\mathcal{V}$ union $\{Q'\}$. Now we compute again $\mathcal{V}'(D_Q)$ and construct a new graph G'_R, and it is easy to see that a path p' in G'_R from X to Y is $p' = 6, 3, 7, 2$ (actually p' comes from p after we delete nodes 5 and 4, as now there is a view $Q'(3,7)$). Now we consider query $Q'' : -c_2bcde$ and using Lemma 3, we produce the following equivalent rewriting of Q'' using v_1, v_2 and Q':

$$\phi'(X,Y) : \exists Z[v_1(X,Z) \wedge \forall W((v_2(W,Z) \rightarrow Q'(W,Y))].$$

This rewriting uses Q' as a view but we have ϕ (from Example 6) which is an equivalent rewriting of Q' using $\mathcal{V}$ and we can replace ϕ for Q' in ϕ' thus producing an equivalent rewriting of Q'' using $\mathcal{V}$.

Finally we produce an equivalent rewriting of Q based on Lemma 3 and using the views Q'', v_6 and v_7 which is the following:

$$\phi''(X,Y) : \exists Z[Q''(X,Z) \wedge \forall W((v_6(W,Z) \rightarrow v_7(W,Y))].$$

Concluding, an equivalent rewriting of Q using $\mathcal{V}$ is ϕ'' where we have replaced Q'' by its equivalent rewriting ϕ' after we have replaced in ϕ' the equivalent rewriting ϕ of Q'.

We include another example where clause 2 of the claim is satisfied just to demonstrate how this case is also possible. We don't go into detail however, since the analysis is very similar to the running example of this proof.

Example 8. Consider the chain query $Q : -c_1c_2bcde$ and the chain views $\mathcal{V}$: $v_1 : -c_1c_2cd$, $v_2 : -cd$, $v_3 : -c$, $v_4 : -c_2c$ and $v_5 : -c_2cde$.

If we want to use the Alternative Claim, then we argue in a similar fashion, as follows: By induction each of the three queries in the Alternative Claim can be rewritten in terms of views in $\mathcal{V}$. Then, using these three queries as the view set we can rewrite the original query appropriately. □

Proof of Theorem 2. Let R_c be the canonical rewriting of Q with respect to $\mathcal{V}$.

1. Lemma 1 proves that if $\mathcal{V}$ determines Q then the canonical rewriting R_c is not disjoint. Lemma 2 proves that if the canonical rewriting R_c is not disjoint then there is an FO equivalent rewriting of Q using $\mathcal{V}$, hence $\mathcal{V}$ determines Q.
2. If $\mathcal{V}$ determines Q then R_c is not disjoint according to Lemma 1. Lemma 2 gives an FO equivalent rewriting of Q using $\mathcal{V}$.
3. We check R_c. If R_c is disjoint then $\mathcal{V}$ does not determine Q. Otherwise $\mathcal{V}$ determines Q. □

Open problem: In the proof of Lemma 2, we may use FO formulas of arbitrary quantifier alternation length. Thus an open problem of independent theoretical interest is: Are formulas of arbitrary quantifier alternation length necessary in order to express an equivalent rewriting in FO language of a chain query using a set of chain views? Our conjecture is that the answer to this question is "yes".

3.2. Path queries—CQ is almost complete for rewriting

In this subsection we will prove the following theorem:

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

The above theorem is a consequence of Lemma 4.

Lemma 4. *Let P_n be a query and let view set 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.*

Proof. First, (3) implies (1) because the existence of an equivalent rewriting proves determinacy. Also (2) implies (3) because of Theorem 2. Thus, it remains to be proven that (1) implies (2). We will prove, equivalently, that if the canonical rewriting is not disjoint then there is a CQ equivalent rewriting.

Because the canonical rewriting is not disjoint, by definition, there is an undirected path from its start node to its end node. Therefore $\Sigma_{i=1}^{K} x_i k_i = n$ has an integer solution and hence the greatest common divisor (GCD for short) of $k_1, \ldots, k_K$ divides n (to see that this holds, divide both sides of $\Sigma_{i=1}^{K} x_i k_i = n$ by the GCD; then the left hand side is an integer, hence the right hand side is an integer too, hence the GCD divides n). We want to prove that for $n > \sum_{i \neq q; i,q=1,\ldots,K} k_i k_q$, if the GCD of $k_1, \ldots, k_K$ divides n then $\Sigma_{i=1}^{K} x_i k_i = n$ has a nonnegative integer solution. We prove it inductively on the number of k_i's.

Inductive hypothesis: For any positive integers $k_1, \ldots, k_j$ and for any positive integer m, if the GCD of $k_1, \ldots, k_j$ divides m and

$$m > \sum_{i \neq q; i,q=1,\ldots,j} k_i k_q \tag{1}$$

then the equation $\Sigma_{i=1}^{j} x_i k_i = m$ has a nonnegative integer solution $x_1^0, \ldots, x_j^0$ such that, for any d in $\{1, 2, \ldots, j\}$, the following holds:

$$x_d^0 k_d > m - \sum_{i \neq q; i,q=1,\ldots,j} k_i k_q. \tag{2}$$

Proof of the inductive hypothesis:

Base step. In this step we need to show that the equation $a_1 x_1 + a_2 x_2 = b_0$, where a_1, a_2, b_0 are positive integers, and $b_0 > a_1 a_2$ has a nonnegative integer solution x_1^0, x_2^0 such that $a_1 x_1^0 > b_0 - a_1 a_2$. This is a Diophantine equation and it is a known fact.

Inductive step. Suppose that the inductive hypothesis holds for $j \to j$. Then, we will prove that it also holds for $j \to j+1$. I.e., we will prove that for any positive integers $k_1, \ldots, k_{j+1}$ and any positive integer p such that the GCD of $k_1, \ldots, k_{j+1}$ divides p and

$$p > \sum_{i \neq q; i,q=1,\ldots,j+1} k_i k_q \tag{3}$$

then the equation $\Sigma_{i=1}^{j+1} x_i k_i = p$ has a nonnegative integer solution $x_1^0, \ldots, x_{j+1}^0$ such that, for any d in $\{1, 2, \ldots, j+1\}$, the following holds:

$$x_d^0 k_d > p - \sum_{i \neq q; i,q=1,\ldots,j+1} k_i k_q. \tag{4}$$

Without loss of generality suppose $d = 1$. From here on, we suppose $k_1 = a_1$ and $k_2 = a_2$. We rewrite the equation as: $a_1 x_1 + a_2 x_2 = p - \Sigma_{i=3}^{j+1} x_i k_i$. Let the GCD of a_1, a_2 be b. Then we rewrite: $(a_1/b)x_1 + (a_2/b)x_2 = (p - \Sigma_{i=3}^{j+1} x_i k_i)/b$. Now, we look for solutions of: $p = \Sigma_{i=3}^{j+1} x_i k_i + bx$. There are solutions because the GCD of $k_3, \ldots, k_{j+1}, b$ is equal to the GCD of $k_1, \ldots, k_{j+1}$ (to prove observe that any common divisor of a_1, a_2 must divide b) hence it divides p. Also

$$p > a_1 a_2 + \sum_{i \neq q; i,q=3,\ldots,j+1} k_i k_q + b \sum_{i=3,\ldots,j+1} k_i \tag{5}$$

(because $b < a_1$ and $b < a_2$). Hence, according to the inductive hypothesis, $p = \Sigma_{i=3}^{j+1} x_i k_i + bx$ has a nonnegative integer solution $x_3^0, \ldots, x_{j+1}^0, x^0$ such that

$$bx^0 > p - \sum_{i \neq q; i,q=3,\ldots,j+1} k_i k_q - b \sum_{i=3,\ldots,j+1} k_i > a_1 a_2. \tag{6}$$

The last inequality holds because

$$p > a_1a_2 + \sum_{i\neq q;i,q=3,\ldots,j+1} k_ik_q + b\sum_{i=3,\ldots,j+1} k_i \tag{7}$$

Then we have $(p - \Sigma_{i=3}^{j+1}x_i^0k_i)/b = x^0$. Now it suffices to prove that the equation $a_1x_1 + a_2x_2 = bx^0$ has a nonnegative integer solution x_1^0, x_2^0 such that $a_1x_1^0 > bx^0 - a_1a_2$. Observe that $bx^0 > a_1a_2$. Also the GCD of a_1, a_2 (i.e., b) divides bx^0. Hence $a_1x_1 + a_2x_2 = bx^0$ has a nonnegative integer solution x_1^0, x_2^0 such that $a_1x_1^0 > bx^0 - a_1a_2$. Hence $a_1x_1^0 + a_2x_2^0 = bx^0$ and by replacing bx^0 we have that $a_1x_1^0 + a_2x_2^0 = p - \Sigma_{i=3}^{j+1}x_i^0k_i$. Finally we need to show that x_1^0 satisfies the inequality required in the inductive hypothesis. Indeed, since $a_1x_1^0 > bx^0 - a_1a_2$ we get that

$$a_1x_1^0 > bx^0 - a_1a_2 > p - \sum_{i\neq q;i,q=3,\ldots,j+1} k_ik_q - b\sum_{i=3,\ldots,j+1} k_i - a_1a_2 > p - \sum_{i\neq q;i,q=1,\ldots,j+1} k_ik_q. \quad \square \tag{8}$$

An interesting special case of the above, where a CQ query is determined by a set of CQ views but there is no equivalent CQ rewriting is stated in the following proposition. This is one of the simplest examples where a path query is defined by a set of path views but there is no equivalent CQ rewriting.

Proposition 3. *The view set $\{P_3, P_4\}$ determines the query P_5. There is no CQ rewriting of P_5 using $\{P_3, P_4\}$.*

Proof. Let $P_5(X, Y) : -e(X, Z_1), e(Z_1, Z_2), e(Z_2, Z_3), e(Z_3, Z_4), e(Z_4, Y)$.

The following formula is an equivalent rewriting of P_5 (it is not a CQ however):

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

The proof that it is an equivalent rewriting is similar to the proof of Lemma 3. To see that there is no rewriting, take the canonical rewriting which is as follows:

$$R_c(X, Y) : -P_3(X, Z_3), P_3(Z_1, Z_4), P_3(Z_2, Y), P_4(X, Z_4), P_4(Z_1, Y).$$

The expansion of R_c is not a query equivalent to P_5. $\square$

4. Single view. Query equivalence

4.1. Single view case

The following defines a language to be *s-complete* when it is complete for any view set (from a specific query language) that contains a single view.

Definition 8 (*(Almost)* s-Complete Language for Rewriting)**.** Let $\mathcal{L}_Q$ and $\mathcal{L}_\mathcal{V}$ be query languages or sets of queries. Let $\mathcal{L}$ be a query language.

We say that $\mathcal{L}$ is *s-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 view V in $\mathcal{L}_\mathcal{V}$: If V determines Q then there is an equivalent rewriting in $\mathcal{L}$ of Q using V. We say that $\mathcal{L}$ *is s-complete for rewriting* if it is complete for $\mathcal{L}$ -to-$\mathcal{L}$ rewriting.

We say that $\mathcal{L}$ is *almost s-complete for $\mathcal{L}_\mathcal{V}$-to-$\mathcal{L}_Q$ rewriting* if for every view V_1 from $\mathcal{L}_\mathcal{V}$ 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 V_1-to-$\mathcal{L}_{Q1}$ rewriting. We say that *$\mathcal{L}$ is almost s-complete for rewriting* if it is almost s-complete for $\mathcal{L}$ -to-$\mathcal{L}$ rewriting.

In Proposition 3, we have an example of a path query and a view set which contains two path views for which the views determine the query but there is no CQ equivalent rewriting. In Theorem 1(2), we have presented a simple case where CQ is complete for rewriting when we have one view in the view set. In [25], it is proven that CQ is complete for all CQ queries in the special case where we have a single path view in the view set. I.e., CQ is s-complete for CQ$_{path}$-to-CQ rewriting. The following extends this result to chain query language.

Chain Query Language. The following is an easy consequence of Theorem 2.

Theorem 4. *CQ$_{chain}$ (and hence CQ) is s-complete for CQ$_{chain}$-to-CQ$_{chain}$ rewriting.*

Proof. Let $\mathcal{V}$ be a set of chain views that determine a chain query Q. According to Theorem 2, this yields that the canonical rewriting is not disjoint. However, since we have only one chain view, the existence of an undirected path joining two nodes yields the existence of a directed path joining those two nodes. The directed path however yields directly an equivalent rewriting which is a CQ. $\square$

4.2. Determinacy and query equivalence

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 9. 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

$$\begin{aligned} R^{exp} : q_1'(X, Y, Z, W, A, B) : &-r(Y, X), s(Y, X), r(Z, W), s(Z, Z_1'), s(Z_1', Z_2'), s(Z_2', W), s(A, A_1'), s(A_1', A_1'), s(A_1', B), \\ &r(Y_1, X_1), s(Y_1, X_1), r(Z_1, W_1), s(Z_1, Z_1''), s(Z_1'', Z_2''), s(Z_2'', W_1), s(Z, A_1''), s(A_1'', A_1''), s(A_1'', W). \end{aligned}$$

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}:

$$\begin{aligned} \mu_1 : \{&X \to X, Y \to Y, Z \to Z, W \to W, A \to A, B \to B, Z_1' \to Z_1, Z_2' \to Z_1, A_1' \to A_1, \\ &A_1'' \to A_1, X_1 \to X, Y_1 \to Y, Z_1 \to Z, W_1 \to W, Z_1'' \to Z_1, Z_2'' \to Z_1\} \\ \mu_2 : \{&X \to X, Y \to Y, Z \to Z, W \to W, A \to A, B \to B, A_1 \to A_1', Z_1 \to A_1''\}. \end{aligned}$$

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 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: $Q(D_3) \subseteq_s Q(D_2)$.

In this paper whenever we use the notation $Q(D_1) \subseteq_s Q(D_2)$, D_1 and D_2 are canonical databases of queries Q_1 and Q_2. In this case, we add more conditions to the above definition that relate to specific constants in D_1 and D_2 (actually these constants are related to the head variables of queries Q_1 and Q_2 as we will explain shortly). Although we need incorporate these constants in the notation, we will keep (slightly abusively) the same notation here since it is clear which constants we mean. Thus, 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 the 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 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? 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 5 is formal evidence of that.

Theorem 5. *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 s-complete for CQ_2-to-CQ_1 rewriting.*
(B) *Determinacy defines CQ_2-to-CQ_1 equivalence.*

Proof. Let Q_1 and Q_2 be queries such that Q_2 determines Q_1 and all the assumptions of the "determinacy and query equivalence" problem (stated above) are satisfied. Remember that these assumptions include that $Q_2(D_1) \subseteq_s Q_2(D_2)$ and 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.

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 2). 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 the following: Observe that the atoms in the body of R_c and $Q_2(D_1)$ are isomorphic because they are both found by applying the same homomorphisms from Q_2 to Q_1 (or its isomorphic D_1). For the same reason, if we take the canonical rewriting R_{c2} of Q_2 using Q_2 (notice that this rewriting does not necessarily contain only a single atom), then the atoms in the body of R_{c2} and $Q_2(D_2)$ are isomorphic. Since $Q_2(D_1) \subseteq_s Q_2(D_2)$, this implies that there is a one-to-one containment mapping from R_c to R_{c2}. This mapping can be extended (since both R_c and R_{c2} are computed by using as a view the query Q_2) to a containment mapping from R_c^{exp} to R_{c2}^{exp}. Since R_{c2}^{exp}, by construction, is equivalent to Q_2, this means that there is a containment mapping from R_c^{exp} to Q_2. □

In [25] it is proven that CQ is s-complete for CQ_{path}-to-CQ rewriting. A consequence of it and Theorem 5 is the following:

Theorem 6. *Determinacy defines CQ_{path}-to-CQ equivalence.*

The *determinacy and query equivalence* question remains open. Theorem 7 settles a special case where, in the definition of the "determinacy defines equivalence" problem, we have replaced condition (b) with a stronger one. Note that Theorem 7 provides an alternative proof of Theorem 6.

Theorem 7. *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 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.*

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. We assume, wlog, that both queries have no repeated variables in the head.

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 5. *Let Q_1 and Q_2 be CQ queries as in the statement of Theorem 7. Suppose Q_1 and Q_2 are not equivalent. Then the following holds:*

There is a CQ 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) *there exists a containment mapping h from Q_1' to Q_1 such that it is identity for all variables of Q_1' except that $h(X_1) = h(X_2) = X_1$.*

Proof. The proof of the lemma has two parts. In the first part we prove that any containment mapping μ_i from Q_2 to Q_1 is subgoals-onto. In the second part, we prove that if a query Q_1 is contained in a query Q_2, Q_1 and Q_2 are not equivalent and also any containment mapping μ_i from Q_2 to Q_1 is subgoals-onto then, the consequent clause of the lemma is true.

First we prove that since Q_2 determines Q_1, any containment mapping μ_i from Q_2 to Q_1 is subgoals-onto. Suppose not. Then, take any μ_i and take database D' to be the canonical database of Q_1 and database D'' to be the canonical database of Q_1 except a subgoal that is not in $\mu_i(Q_2)$. Since Q_1 is minimized, we have that the frozen head tuple t_0 belongs in $Q_1(D')$ but t_0 does not belong to $Q_1(D'')$. Hence $Q_1(D') \neq Q_1(D'')$. Since $Q_2(D')$ contains only one tuple and $Q_2(D'') \subseteq Q_2(D')$, and μ_i can be used to obtain the head tuple of Q_2 when applied on both D' and D'', we have that $Q_2(D'') = Q_2(D')$. However, since we proved that $Q_1(D') \neq Q_1(D'')$, this is a contradiction to the determinacy assumption.

Let h_1 be a homomorphism (containment mapping) 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 and has the properties as in the statement of the lemma. Observe that, during the construction, we conveniently keep the names of the variables in Q_1 and Q_1' (except X_2 which appears only in Q_1').

Since Q_1 and Q_2 are not equivalent and h_1 is subgoals-onto, there are variables X_1, X_2 of Q_2 such that $h_1(X_1) = h_1(X_2)$.[1] Then h is the following: It is the same as h_1 except that for h we have $h(X_1) \neq h(X_2)$. 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. □

[1] If h_1 was not subgoals-onto, then h_1 could be one-to-one and still the two queries not be equivalent because non-equivalence would be guaranteed by the existence of a subgoal not in $h_1(Q_2)$.

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 5 with respect to Q_1, namely, 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 (this is so because we assumed that there are no repeated variables in the head of the queries).

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. □

Theorem 8 is a consequence of Theorem 7.

Theorem 8. *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 10. 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 do 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 with respect to 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 6 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.

A query is *head-connected* if all subgoals containing head variables are contained in the same connected component of the query (recall Definition 5).

Definition 9 (*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 the subgoals that form the nodes of 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 *wrt* $\mathcal{V}$.

Lemma 6. *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 wrt* $\mathcal{V}$*. Then there is a view in* $\mathcal{V}$ *which contains a connected component which is isomorphic to* G_Q*.*

We need the definition:

Definition 10 (*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.

Proof of Lemma 6.

Proof. Because of Proposition 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 $Q(D_Q) \neq Q(D_1)$ (because Q is minimized). □

As a consequence of Lemma 6, 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 wrt $\mathcal{V}$ 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 9. *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 proof of Theorem 9 is a immediate consequence of the following:

Proposition 4. 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. 1. 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 $Q'(D_i) = Q(D_i')$, $i = 1, 2$. 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).

Now it is easy to show that on D_1' and D_2', views $\mathcal{V}$ compute the same 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.

2. The same equivalent rewriting R where we replace the views from $\mathcal{V}'$ with their counterpart in $\mathcal{V}$. Let R^{exp} be the expansion of R using $\mathcal{V}$ and let R^{exp1} be the expansion of R using $\mathcal{V}'$. Since R is an equivalent rewriting of Q' using $\mathcal{V}'$, R^{exp1} and Q' are equivalent. Hence there are containment mappings appropriately. These mappings can be used to define mappings between Q and R^{exp}: keep the mapping the same (assuming the names of the variables are retained as necessary and possible) and add an isomorphism between the semi-covered components. □

The following is a corollary of Theorem 9 and results from Section 3:

Theorem 10. *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}.*

Then it holds: CQ_{path} (and hence CQ) is almost complete for CQ_{apath}-to-CQ_{apath} rewriting.

6. Conclusion

In this paper we considered query and views that are conjunctive queries (CQ) or fragments of it and investigated the problem of decidability of determinacy and of a language being complete for rewriting. We first showed special cases where CQ is complete for rewriting. Then we identified two fragments of CQ with good behavior, namely chain queries and path queries. For chain query and views we showed that FO is complete for rewriting and determinacy is decidable. For path query and views, we observed that CQ is not complete for rewriting (there are simple counterexamples) but it only misses by a finite set of queries for each view set. Thus we defined the notion of a language being *almost* complete for rewriting and showed that CQ is almost complete for rewriting for path query and views. We explained how query equivalence is related to determinacy in the special case where we have one view in the view set. We showed that there are fragments of CQ for which determinacy does define equivalence. Finally, we showed how to extend results about determinacy to broader classes of queries using connectivity properties of the body of the query.

It is not easy to see how to extend the results in Theorem 1, although it will not be surprising if similar techniques work for slightly broader query languages than the ones considered in case 2 of Theorem 1. The results in Section 5 may be extended either towards identifying more connectivity properties that simplify determinacy related problems or towards using connectivity to extend significantly known results. The problem about whether determinacy defines query equivalence remains open for CQ queries, while a special case is solved here.

We believe that the results in Section 3 can be significantly extended to capture the case where the views are general CQ queries, or, on the more conservative side, when the views are acyclic queries. Towards this direction, we first observe that if we consider views to be *extended chain* queries (i.e., binary queries with body as in chain queries, only that we may export in the head any pair of variables) then Lemmas 1 and 2 can be probably easily extended. We provide now some intuition about how to extend those lemmas even for CQ views. First we need to state and prove a proposition which will "exclude" certain views in a similar way Proposition 1(e) excludes views with predicates not used in the query. The excluded views should be such that they "cannot produce" an extended chain query. Or, in other words, that an FO or a CQ equivalent rewriting will only use excluded views. A not excluded view will have the following property: Let us consider view $V(X, Y, \bar{Z})$ over binary predicates, where there are two head variables X and Y such that there is a path from one variable to the other (if the body is viewed as a graph). For such a view, we can produce an extended chain view $Q'(X, Y)$ as follows: $Q'(X, Y) :- \exists \bar{Z} V(X, Y, \bar{Z})$. This intuition leads us to conjecture the following extension of Theorem 2:

Table 1
Summary of results for complete languages for rewriting obtained in this paper and most related main results in the literature.

Rewriting lang.	View lang.	Query lang.	Complete	Reference
FO	CQ_{chain}	CQ_{chain}	Yes	Theorem 2
FO	CQ	CQ_{chain}	Open	Conjecture 1
FO	∃FO	FO	No	[27]
∃SO∩∀SO	FO	FO	Yes	[27]
Datalog$^{\neq}$	UCQ	UCQ	No	[27]
CQ	CQ	CQ	No	[25]
CQ	$CQ_{Boolean}$	CQ	Yes	[25]
CQ	$CQ_{Monadic}$	CQ	Yes	[25]
CQ	CQ_{full}	CQ	Yes	Theorem 1

Table 2
Summary of results for almost complete languages for rewriting.

Rewriting lang.	View lang.	Query lang.	Al. complete	Reference
CQ_{path}	CQ_{path}	CQ_{path}	Yes	Theorem 3
CQ_{path}	CQ_{apath}	CQ_{apath}	Yes	Theorem 10
CQ	CQ	CQ_{path}	Open	Conjecture 2

Table 3
Summary of results when we have only one view in the view set. The asterisk (∗) denotes the one case in this table where we have two views in the view set.

Rewriting lang.	View lang.	Query lang.	S-Complete	Reference
CQ	CQ_{path}	CQ	Yes	[25]
CQ	CQ	CQ_{path}	Open	
CQ	CQ_{path}	CQ_{path}	No∗	Proposition 3
CQ_{chain}	CQ_{chain}	CQ_{chain}	Yes	Theorem 4
CQ	CQ_{3var}	CQ_{1var}	Yes	Theorem 1

Conjecture 1. *Let query Q be a chain query and $\mathcal{V}$ be CQ views. 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-to-CQ_{chain} rewriting.*
3. *It is decidable whether a set of views determines a query.*

If the above conjecture is true then we believe that it will be rather easy to derive the following extension of Theorem 3:

Conjecture 2. *CQ is almost complete for CQ-to-CQ_{path} rewriting.*

In Tables 1–3 we summarize the results in this paper and related results from [27,25]. Finally another interesting issue is to investigate the same questions in the unrestricted case, namely in the case database instances are not restricted to be finite. This is investigated in [27,25] and the relation between the restricted (finite) and the unrestricted case is shown in certain cases, whereas many open problems remain to be solved that are discussed therein. In Table 3, we also mention a problem for the single view case which is open but it will not be surprising if CQ was complete for rewriting.

References

[1] Serge Abiteboul, Oliver M. Duschka, Complexity of answering queries using materialized views, in: PODS, 1998.
[2] Foto N. Afrati, Rewriting conjunctive queries determined by views, in: MFCS, 2007.
[3] Foto N. Afrati, Chen Li, Jeffrey D. Ullman, Generating efficient plans using views, in: SIGMOD, 2001.
[4] Foto N. Afrati, Chen Li, Jeffrey D. Ullman, Using views to generate efficient evaluation plans for queries, JCSS 73 (5) (2007) 703–724.
[5] Foto N. Afrati, Chen Li, Prasenjit Mitra, Rewriting queries using views in the presence of arithmetic comparisons, Theoret. Comput. Sci. 368 (1–2) (2006).
[6] Foto N. Afrati, Rada Chirkova, Manolis Gergatsoulis, Benny Kimelfeld, Vassia Pavlaki, Yehoshua Sagiv, On rewriting XPath queries using views, in: EDBT, 2009.
[7] Sanjay Agrawal, Surajit Chaudhuri, Vivek R. Narasayya, Automated selection of materialized views and indexes in sql databases, in: Proc. of VLDB, 2000.
[8] Roberto J. Bayardo Jr., et al. Infosleuth: semantic integration of information in open and dynamic environments (experience paper), in: SIGMOD, 1997.
[9] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi, Lossless regular views, in: PODS, 2002.
[10] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi, View-based query processing: on the relationship between rewriting, answering and losslessness, in: ICDT, 2005.
[11] Ashok K. Chandra, Philip M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: STOC, 1977.
[12] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, Kyuseok Shim, Optimizing queries with materialized views, in: ICDE, 1995.
[13] Sudarshan S. Chawathe, et al. The TSIMMIS project: integration of heterogeneous information sources, in: IPSJ, 1994.
[14] C. Chekuri, A. Rajaraman, Conjunctive query containment revisited, in: ICDT, 1997.
[15] Oliver M. Duschka, Michael R. Genesereth, Answering recursive queries using views, in: PODS, 1997.

[16] Daniela Florescu, Alon Levy, Dan Suciu, Khaled Yagoub, Optimization of run-time management of data intensive web-sites, in: Proc. of VLDB, 1999.
[17] Stéphane Grumbach, Leonardo Tininini, On the content of materialized aggregate views, in: PODS, 2000.
[18] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, Jun Yang, Optimizing queries across diverse data sources, in: Proc. of VLDB, 1997.
[19] Alon Y. Halevy, Answering queries using views: a survey, VLDB J. 10 (4) (2001).
[20] Zachary Ives, Daniela Florescu, Marc Friedman, Alon Levy, Dan Weld, An adaptive query execution engine for data integration, in: SIGMOD, 1999.
[21] Maarten Marx, Queries determined by views: pack your views, in: PODS, 2007.
[22] Alon Levy, Alberto O. Mendelzon, Yehoshua Sagiv, Divesh Srivastava, Answering queries using views, in: PODS, 1995.
[23] Alon Levy, Anand Rajaraman, Joann J. Ordille, Querying heterogeneous information sources using source descriptions, in: Proc. of VLDB, 1996.
[24] Chen Li, Mayank Bawa, Jeff Ullman, Minimizing view sets without losing query-answering power, in: ICDT, 2001.
[25] Alan Nash, Luc Segoufin, Victor Vianu, Determinacy and rewriting of conjunctive queries using views: A progress report, in: ICDT, 2007.
[26] Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Answering queries using templates with binding patterns, in: PODS, 1995.
[27] Luc Segoufin, Victor Vianu, Views and queries: Determinacy and rewriting, in: PODS, 2005.
[28] Dimitri Theodoratos, Timos Sellis, Data warehouse configuration, in: Proc. of VLDB, 1997.
[29] Jeffrey D. Ullman, Information integration using logical views, in: ICDT, 1997.