Determinacy and Rewriting of Conjunctive Queries Using Views: A Progress Report

Alan Nash[1], Luc Segoufin[2], and Victor Vianu[1]*

[1] UC San Diego
[2] INRIA and Univ. Paris 11

Abstract. Suppose we are given a set of exact conjunctive views $\mathbf{V}$ and a conjunctive query Q. Suppose we wish to answer Q using $\mathbf{V}$, but the classical test for the existence of a conjunctive rewriting of Q using $\mathbf{V}$ answers negatively. What can we conclude: (i) there is no way Q can be answered using $\mathbf{V}$, or (ii) a more powerful rewriting language may be needed. This has been an open question, with conventional wisdom favoring (i). Surprisingly, we show that the right answer is actually (ii). That is, even if $\mathbf{V}$ provides enough information to answer Q, it may not be possible to rewrite Q in terms of $\mathbf{V}$ using just conjunctive queries – in fact, no monotonic language is sufficiently powerful. We also exhibit several well-behaved classes of conjunctive views and queries for which conjunctive rewritings remain sufficient. This continues a previous investigation of rewriting and its connection to semantic determinacy, for various query and view languages.

1 Introduction

The question of whether a given set $\mathbf{V}$ of views on a database can be used to answer another query Q arises in many different contexts. For instance, it is a central issue in data integration, semantic caching, security and privacy. The question can be formulated at several levels. The most general definition is information theoretic: $\mathbf{V}$ *determines* Q (which we denote $\mathbf{V} \twoheadrightarrow Q$) iff $\mathbf{V}(D_1) = \mathbf{V}(D_2) \rightarrow Q(D_1) = Q(D_2)$, for all database instances D_1 and D_2. Intuitively, determinacy says that $\mathbf{V}$ provides enough information to uniquely determine the answer to Q. However, it does not say that this can be done effectively, or using a particular query language. The next formulation is language specific: a query Q can be *rewritten* in terms of $\mathbf{V}$ using a rewriting language $\mathcal{R}$ iff there exists some query $R \in \mathcal{R}$ such that $Q(D) = R(\mathbf{V}(D))$ for all databases D.

What is the relationship between determinacy and rewriting? Suppose $\mathcal{R}$ is a rewriting language. Clearly, if Q can be rewritten in terms of $\mathbf{V}$ using some query $R \in \mathcal{R}$, then $\mathbf{V} \twoheadrightarrow Q$. The converse is generally not true. Given a view language $\mathcal{V}$ and query language $\mathcal{Q}$, if $\mathcal{R}$ can be used to rewrite a query Q in $\mathcal{Q}$ in terms of a set of views $\mathbf{V}$ in $\mathcal{V}$ whenever $\mathbf{V} \twoheadrightarrow Q$, we say that $\mathcal{R}$ is *complete* for $\mathcal{V}$-to-$\mathcal{Q}$ rewritings.

* Work supported in part by the NSF under grant number INT-0334764.

Query rewriting using views has been investigated in the context of data integration for some query languages, primarily conjunctive queries (CQs). However, the connection between rewriting and the semantic notion of determinacy has received little attention. For example, a classical algorithm allows to test whether a CQ query has a CQ rewriting using a set of exact CQ views. Suppose the algorithm says there is no such rewriting. Does this mean that the view does not determine the query, or could it be that CQs are just not powerful enough to rewrite Q in terms of $\mathbf{V}$? If so, what is the language needed for the rewriting?

In [14], a subset of the authors undertook a systematic investigation of these issues. They considered view languages $\mathcal{V}$ and query languages $\mathcal{Q}$ ranging from first-order logic (FO) to CQ and studied two main questions:

(i) is it decidable whether $\mathbf{V} \twoheadrightarrow Q$ for $\mathbf{V}$ in $\mathcal{V}$ and Q in $\mathcal{Q}$?
(ii) is $\mathcal{Q}$ complete for $\mathcal{V}$-to-$\mathcal{Q}$ rewritings? If not, how must $\mathcal{Q}$ be extended in order to express such rewritings?

As usual, all definitions and results come in two flavors: in the *unrestricted* case, databases can be finite or infinite. In the *finite* case, databases are assumed to be finite. The results of [14] concern languages ranging from FO to CQ, in both the unrestricted and finite cases. However, the main questions remained open for CQ, the simplest and most common language for defining views and queries. In the present paper we report some progress on this front. First, we settle in the negative the question of whether CQ is complete for CQ-to-CQ rewriting. In fact, no monotonic language can be complete for CQ-to-CQ rewriting. We then provide several classes of CQ views and queries for which CQ remains complete for rewriting, and for which determinacy is decidable. One such class consists of monadic CQ views and arbitrary CQ queries. Beyond monadic views, CQ can only remain complete for very limited classes of views. Indeed, we show that non-monotonic rewrite languages are required even for very simple CQ views whose patterns are trees, and that differ from simple paths by a single edge. We show that CQ remains complete for binary views consisting of a simple path.

Related work Answering queries using views arises in numerous contexts including data integration [15], query optimization and semantic caching [8], data warehousing, support of physical data independence by describing storage schemas as views [9], etc. The problem comes in several flavors, depending on assumptions on the views and their use. Mainly, the different settings vary along these dimensions:

(i) assumptions on the views: these may be *exact* (i.e. contain precisely the set of tuples in their definitions), or just *sound* (they provide only a subset of the tuples in the answer)
(ii) how the views are used: *query rewriting* requires reformulating the query in terms of the views, using some query language. One may require an *equivalent* rewriting, or just a *maximally contained* one. Another use of views is called *query answering.* This consists of finding all *certain* answers to a query given an instance of the view [1].

In our investigation, we focus on exact view definitions, and equivalent query rewritings, with the accompanying information-theoretic notion of determinacy. Results on equivalent query rewriting using exact views have focused primarily on CQs and UCQs (unions of CQs). It is shown in [12] that it is NP-complete whether a given (U)CQ query has an equivalent (U)CQ rewriting in terms of given (U)CQ views. Several polynomial-time special cases are identified for CQs in [7]. Answering queries using views in the presence of binding patterns is considered in [13]. Views and queries defined by CQs with arithmetic comparisons over dense orders are considered in [3], where it is shown that the existence of an equivalent rewriting using Datalog with comparisons is decidable.

The relation of rewriting to the information-theoretic notion of determinacy has received little attention. In [10, 11], Grumbach and Tininini consider the problem of computing an aggregate function using a given set of aggregate functions including count, average, sum, product, maximum. In particular, [11] introduces the notion of *subsumption* of a query by a view, which is identical to our notion of determinacy. Using this, they define completeness of a rewriting algorithm, and produce such an algorithm for simple aggregate functions on a single relation. Despite the similarity in flavor, none of the results transfer to the setting we consider.

In [5], the authors consider the notion of *lossless* view with respect to a query, in the context of regular path queries on semi-structured data. Losslessness is considered under the exact view assumption and under the sound view assumption. In the first case, losslessness is equivalent to determinacy and it remains open whether losslessness is decidable for regular path views and queries. In the second case, losslessness is shown to be decidable using automata-theoretic techniques. Again, these results have no bearing upon ours because of the differences in the settings and because we consider exact views.

Bancilhon and Spyratos [4] defined the notion of determinacy in the context of their investigation of view updates. In particular, they defined the notion of *view complement*. The complement of a view is another view so that together they uniquely determine the underlying database. Thus, a view and its complement determine the identity query on the database.

This paper is a direct follow-up of [14]. We briefly summarize the results of [14] for some key combinations of query and view languages. In the unrestricted case, FO turns out to be complete for FO-to-FO rewritings, as a consequence of Craig's Interpolation theorem [6]. Unfortunately this does not extend to the finite case: FO is no longer complete for FO-to-FO rewritings. In fact, any language complete for FO-to-FO rewritings must express all computable queries.

For views expressed in weaker languages, less powerful rewriting languages are needed. If views are expressed in $\exists$FO (existential FO), FO is still not complete for $\exists$FO-to-FO rewritings. However, both $\exists$SO and $\forall$SO (existential and universal second-order logic formulas) are complete for such rewritings. In fact $\exists$SO $\cap$ $\forall$SO is a lower bound, even if views are restricted to UCQs.

Consider UCQ views and queries. Similarly (but for different reasons), UCQ is not complete for UCQ-to-UCQ rewritings, nor are much more powerful languages such as Datalog $^{\neq}$. This also turns out to hold for CQ$^{\neq}$-to-CQ rewritings.

The results on determinacy are mostly negative: it is shown that determinacy is undecidable even for UCQ views and queries. The question is left open for CQs (although completeness of CQ and, as a corollary, decidability of determinacy, are erroneously claimed for CQs in the unrestricted case, as discussed below).

Organization After recalling some basic concepts and notation in Section 2, we show in Section 3 that CQs are not complete for CQ-to-CQ rewriting. For the unrestricted case, we exhibit an effective FO rewriting of Q in terms of $\mathbf{V}$, whenever $\mathbf{V}$ determines Q. For the finite case, the upper bound of $\exists SO \cap \forall SO$ shown in [14] remains the best available. In Section 4 we consider special cases of CQs for which determinacy is decidable and CQ remains complete as a rewriting language.

2 Basic concepts and notation

We begin with some basic definitions and notation. A database schema σ is a finite set of relation symbols with associated non-negative arities. A relation with arity zero is referred to as a *proposition*. A database instance D over σ associates a relation $D(R)$ of appropriate arity with values from some fixed infinite domain **dom** to each relation symbol R in σ (true/false for propositions). The domain of an instance D consists of the set of elements in **dom** occurring in D and is denoted $dom(D)$. The set of all instances over σ is denoted by $\mathcal{I}(\sigma)$. By default, all instances are assumed to be finite unless otherwise specified. Queries are defined as usual, as computable mappings from instances of an input schema to instances of an output schema that are generic, i.e. commute with isomorphisms of **dom** (e.g., see [2]). We assume familiarity with the query languages first-order logic (FO) and conjunctive queries (CQ).

Let σ and $\sigma_{\mathbf{V}}$ be database schemas. A *view* $\mathbf{V}$ from $\mathcal{I}(\sigma)$ to $\mathcal{I}(\sigma_{\mathbf{V}})$ is a set consisting of one query $Q_V : \mathcal{I}(\sigma) \to \mathcal{I}(V)$ for each $V \in \sigma_{\mathbf{V}}$. We refer to σ and $\sigma_{\mathbf{V}}$ as the input and output schemas of $\mathbf{V}$, respectively.

Consider a query Q over schema σ and a view $\mathbf{V}$ with input schema σ and output schema $\sigma_{\mathbf{V}}$. We say that $\mathbf{V}$ *determines* Q, denoted $\mathbf{V} \twoheadrightarrow Q$, iff for all $D_1, D_2 \in \mathcal{I}(\sigma)$, if $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ then $Q(D_1) = Q(D_2)$. Suppose $\mathbf{V} \twoheadrightarrow Q$ and let R be a query over $\mathcal{I}(\sigma_{\mathbf{V}})$. We say that Q can be *rewritten* in terms of $\mathbf{V}$ using R iff for each $D \in \mathcal{I}(\sigma)$, $Q(D) = R(\mathbf{V}(D))$. In other words, $Q = R \circ \mathbf{V}$. This is denoted by $Q \Rightarrow_{\mathbf{V}} R$. Note that several R's may satisfy this property, since such R's may behave differently on instances in $\mathcal{I}(\sigma_{\mathbf{V}})$ that are not in the image of $\mathbf{V}$.

Let $\mathcal{Q}$ be a query language and $\mathcal{V}$ a view language. A query language $\mathcal{R}$ is *complete* for $\mathcal{Q}$-to-$\mathcal{V}$ rewritings iff for every $Q \in \mathcal{Q}$ and $\mathbf{V} \in \mathcal{V}$ for which $\mathbf{V} \twoheadrightarrow Q$, there exists $R \in \mathcal{R}$ such that $Q \Rightarrow_{\mathbf{V}} R$.

In the unrestricted case, where database instances may be infinite, we will denote by $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$ and by $Q \overset{\infty}{\Rrightarrow}_{\mathbf{V}} R$ the fact that $\mathbf{V}$ determines Q and that R is a rewriting of Q using $\mathbf{V}$. Note that $\overset{\infty}{\twoheadrightarrow}$ implies $\twoheadrightarrow$ and $\overset{\infty}{\Rrightarrow}$ implies $\Rightarrow$ but that the converse does not generally hold [14].

3 CQ is not Complete for CQ-to-CQ Rewriting

In this section, we show that CQ is not complete for CQ-to-CQ rewriting. In fact, no monotonic language can be complete for CQ-to-CQ rewriting. We exhibit effective FO rewritings of CQ queries in terms of CQ views, whenever the views determine the query in the unrestricted case.

Before stating the main result of the section, we recall a test for checking whether a CQ query has a CQ rewriting in terms of a given set of CQ views. The test is based on the chase.

Let σ be a database schema and $Q(\bar{x})$ a CQ over σ with free variables $\bar{x}$. The *frozen body of* Q, denoted $[Q]$, is the instance over σ such that $(x_1, \ldots, x_k) \in R$ iff $R(x_1, \ldots, x_k)$ is an atom in Q. For a set $\mathbf{V}$ of CQs, $[\mathbf{V}]$ is the union of the $[Q]$'s for all $Q \in \mathbf{V}$. For a mapping α from variables to variables and constants, we denote by $\alpha([Q])$ the instance obtained by applying α to all variables in $[Q]$.

Recall that a tuple $\bar{c}$ is in $Q(D)$ iff there exists a homomorphism $h_{\bar{c}}$ from $[Q]$ to D such that $h_{\bar{c}}(\bar{x}) = \bar{c}$. In this case we say that $h_{\bar{c}}$ *witnesses* $\bar{c} \in Q(D)$, or that $\bar{c} \in Q(D)$ via $h_{\bar{c}}$.

Let $\mathbf{V}$ be a CQ view from $\mathcal{I}(\sigma)$ to $\mathcal{I}(\sigma_{\mathbf{V}})$. Let S be a database instance over $\mathcal{I}(\sigma_{\mathbf{V}})$ and C a set of elements. We define the $\mathbf{V}$-inverse of S relative to a domain C, denoted $\mathbf{V}_C^{-1}(S)$, as the instance D over σ defined as follows. Let V be a relation in $\sigma_{\mathbf{V}}$, with corresponding query $Q_V(\bar{x})$. For every tuple $\bar{c}$ belonging to V in S, we include in D the tuples of $\alpha_{\bar{c}}([Q_V])$ where $\alpha_{\bar{c}}(\bar{x}) = \bar{c}$ and $\alpha_{\bar{c}}$ maps every variable of $[Q_V]$ not in $\bar{x}$ to some new distinct value not in $dom(S) \cup C$. Thus, $\mathbf{V}_C^{-1}(S)$ is obtained as a *chase* of S in which all values introduced as witnesses are outside $dom(S)$ and C. To simplify, we usually assume that C consists of the entire domain of the instance when $\mathbf{V}^{-1}$ is applied, and omit specifying it explicitly. Thus, all witnesses introduced by an application of $\mathbf{V}^{-1}$ are new elements.

The following key facts are observed in [9] and [14]:

Proposition 1. *Let $Q(\bar{x})$ be a CQ and $S = \mathbf{V}([Q])$. Let $Q_{\mathbf{V}}(\bar{x})$ be the CQ over $\sigma_{\mathbf{V}}$ for which $[Q_{\mathbf{V}}] = S$. We have the following:*

(i) $Q_{\mathbf{V}} \circ \mathbf{V}$ *is equivalent to the CQ whose frozen body is* $\mathbf{V}^{-1}(S)$*;*
(ii) $Q \subseteq Q_{\mathbf{V}} \circ \mathbf{V}$*;*
(iii) *If* $\bar{x} \in Q(\mathbf{V}^{-1}(S))$ *then* $Q = Q_{\mathbf{V}} \circ \mathbf{V}$*. In particular,* $\mathbf{V} \twoheadrightarrow Q$*.*
(iv) *If Q has a CQ rewriting in terms of* $\mathbf{V}$*, then* $Q_{\mathbf{V}}$ *is such a rewriting.*

Note that Proposition 1 applies both to the finite and unrestricted cases. We can now show the following.

Theorem 1. *CQ is not complete for CQ-to-CQ rewriting.*

Proof. We exhibit a set of CQ views $\mathbf{V}$ and a CQ Q such that $\mathbf{V} \twoheadrightarrow Q$ but $\bar{x} \notin Q(\mathbf{V}^{-1}(S))$, where $S = \mathbf{V}([Q])$. By Proposition 1, this shows that Q has no CQ rewriting in terms of $\mathbf{V}$.

Let the database schema consist of a single binary relation R. Consider the set $\mathbf{V}$ consisting of the following three views (with corresponding graphical representations):

$$\begin{array}{ll} V_1(x,y) = \exists\alpha\exists\beta[R(\alpha,x) \wedge R(\alpha,\beta) \wedge R(\beta,y)] & x \leftarrow \alpha \rightarrow \beta \rightarrow y \\ V_2(x,y) = \exists\alpha[R(x,\alpha) \wedge R(\alpha,y)] & x \rightarrow \alpha \rightarrow y \\ V_3(x,y) = \exists\alpha\exists\beta[R(x,\alpha) \wedge R(\alpha,\beta) \wedge R(\beta,y)] & x \rightarrow \alpha \rightarrow \beta \rightarrow y \end{array}$$

Let $Q(x,y) = \exists a\exists b\exists c[R(a,x) \wedge R(a,b) \wedge R(b,c) \wedge R(c,y)]$.
We first show that $\mathbf{V} \twoheadrightarrow Q$. To do so, we prove that the formula

$$\varphi(x,y): \quad \exists d[V_1(x,d) \wedge \forall e(V_2(e,d) \rightarrow V_3(e,y)]$$

is a rewriting of Q using $\mathbf{V}$. In other words, for each database D over σ and $u, v \in dom(D)$, $\langle u,v\rangle \in Q(D)$ iff $\varphi(u,v)$ holds on the instance $V(D)$.

Suppose $\langle u,v\rangle \in Q(D)$ via a homomorphism h. Note that we also have $\langle x,c\rangle \in V_1([Q])$, via a homomorphism h'. It follows that $h \circ h'$ is a homomorphism from $[V_1]$ to D mapping $\langle x,c\rangle$ to $\langle u,h(c)\rangle$. We let $d = h(c)$ and from the above we have $\langle u,d\rangle \in V_1(D)$. Now because there is an edge $\langle c,y\rangle$ in $[Q]$ we have an edge $\langle h(c),v\rangle$ in D. Therefore for every e such that $\langle e,d\rangle \in V_2(D)$, $\langle e,v\rangle \in V_3(D)$. Thus, $\varphi(u,v)$ holds on $V(D)$. Conversely, suppose $\varphi(u,v)$ holds on $V(D)$. Then there exists d such that $\langle u,d\rangle \in V_1(D)$, so by definition of V_1 there exist α,β such that $R(\alpha,u) \wedge R(\alpha,\beta) \wedge R(\beta,d)$ holds in D. But then $\langle \alpha,d\rangle \in V_2(D)$ so by definition of φ, $\langle \alpha,v\rangle \in V_3(D)$. It follows that there exist b',c' such that $R(\alpha,b') \wedge R(b',c') \wedge R(c',v)$ holds in D. This together with the fact that $R(\alpha,u)$ holds in D, implies that $\langle u,v\rangle \in Q(D)$. Thus, $\varphi(x,y)$ is a rewriting of Q using $\mathbf{V}$, so $\mathbf{V} \twoheadrightarrow Q$.

In view of Proposition 1, it remains to show that $\langle x,y\rangle \notin Q(\mathbf{V}^{-1}(S))$, where $S = \mathbf{V}([Q])$. Clearly, $S = \mathbf{V}([Q])$ is the instance:

V_1	
	$x\ c$
	$b\ c$
	$c\ y$

V_2	
	$a\ c$
	$b\ y$

V_3	
	$a\ y$

and $\mathbf{V}^{-1}(S)$ is depicted in Figure 1.

It is easily checked that $\langle x,y\rangle$ does not belong to Q applied to the above instance. Thus, Q has no CQ rewriting in terms of $\mathbf{V}$. □

Remark 1. Note that Theorem 1 holds in the finite as well as the unrestricted case. This shows that Theorem 3.3 in [14], claiming that CQ is complete for CQ-to-CQ rewriting in the unrestricted case, is erroneous. Also, Theorem 3.7, claiming decidability of determinacy in the unrestricted case as a corollary, remains unproven. The source of the problem is Proposition 3.6, which is unfortunately false (the views and queries used in the above proof are a counterexample).

$$\begin{tikzcd}
\mathbf{a} \arrow[r] \arrow[d] & \beta_1 \arrow[r] & \gamma_1 \arrow[r] & \mathbf{y} \\
\beta_2 \arrow[dd] & & \gamma_3 \arrow[ur] & \gamma_4 \arrow[u] \\
& \beta_3 \arrow[ur] \arrow[dl] & & \\
\mathbf{c} & \beta_5 \arrow[l] & \alpha_5 \arrow[l] \arrow[r] & \mathbf{b} \arrow[uu] \\
\beta_6 \arrow[u] & & & \\
\alpha_6 \arrow[u] \arrow[r] & \mathbf{x} & &
\end{tikzcd}$$

Fig. 1. The graph of $\mathbf{V}^{-1}(S)$ (elements in $dom(S)$ are in boldface).

As a consequence of the proof of Theorem 1 we have:

Corollary 1. *No monotonic language is complete for CQ-to-CQ rewriting.*

Proof. Consider the set of CQ views $\mathbf{V}$ and the query Q used in the proof of Theorem 1. It was shown that $\mathbf{V} \twoheadrightarrow Q$. Consider the database instances $D_1 = [Q]$ and $D_2 = R$ where R is the relation depicted in figure 1. By construction, $\mathbf{V}(D_1) \subset \mathbf{V}(D_2)$. However, $\langle x, y \rangle \in Q(D_1)$ but $\langle x, y \rangle \notin Q(D_2)$, so $Q(D_1) \not\subseteq Q(D_2)$. Thus the mapping $Q_{\mathbf{V}}$ is non-monotonic. $\square$

FO rewriting in the unrestricted case For the set of views $\mathbf{V}$ and query Q above, we showed that Q has a simple FO rewriting in terms of $\mathbf{V}$. We next exhibit general FO rewritings of CQ queries in terms of CQ views, when these determine the query in the unrestricted case. First, we recall a characterization of determinacy in the unrestricted case, based on the chase. Let $\mathbf{V}$ be a set of CQ views and $Q(\bar{x})$ a CQ over the same database schema σ. We construct two infinite instances D_∞ and D'_∞ such that $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$ as follows. We first define inductively a sequence of instances $\{D_k, S_k, V_k, V'_k, S'_k, D'_k\}_{k \geq 0}$, constructed by a chase procedure. Specifically, D_k, D'_k are instances over the database schema σ, and S_k, V_k, V'_k, S'_k are instances over the view schema $\sigma_{\mathbf{V}}$. We will define $D_\infty = \bigcup_k D_k$ and $D'_\infty = \bigcup_k D'_k$. For the basis, $D_0 = [Q]$, $S_0 = V_0 = \mathbf{V}([Q])$, $S'_0 = V'_0 = \emptyset$, and $D'_0 = \mathbf{V}^{-1}(S_0)$. Inductively, $V'_{k+1} = \mathbf{V}(D'_k)$, $S'_{k+1} = V'_{k+1} - V_k$, $D_{k+1} = D_k \cup \mathbf{V}^{-1}(S'_{k+1})$, $V_{k+1} = \mathbf{V}(D_{k+1})$, $S_{k+1} = V_{k+1} - V'_{k+1}$, and $D'_{k+1} = D'_k \cup \mathbf{V}^{-1}(S_{k+1})$ (recall that new witnesses are introduced at every application of $\mathbf{V}^{-1}$).

Referring to the proof of Theorem 1, note that the instance depicted in Figure 1 coincides with D'_0.

The following properties are easily checked.

Proposition 2. *Let D_∞ and D'_∞ be constructed as above. The following hold:*

1. $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$.

2. *There exist homomorphisms from D_∞ into $[Q]$ and from D'_∞ into $[Q]$ that are the identity on $dom([Q])$.*
3. *For every pair of database instances I, I' such that $\mathbf{V}(I) = \mathbf{V}(I')$ and $\bar{a} \in Q(I)$, there exists a homomorphism from D'_∞ into I' mapping the free variables $\bar{x}$ of Q to $\bar{a}$.*

As a consequence of the above we have the following.

Proposition 3. *Let $\mathbf{V}$ be a set of CQ views and $Q(\bar{x})$ a CQ over the same database schema, where $\bar{x}$ are the free variables of Q. Then $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$ iff $\bar{x} \in Q(D'_\infty)$.*

Proof. Suppose $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$. By (1) of Proposition 2, $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$. It follows that $Q(D_\infty) = Q(D'_\infty)$. But $[Q] = D_0 \subset D_\infty$, so $\bar{x} \in Q(D_\infty)$. It follows that $\bar{x} \in Q(D'_\infty)$. Conversely, suppose that $\bar{x} \in Q(D'_\infty)$. Let I, I' be database instances such that $\mathbf{V}(I) = \mathbf{V}(I')$. We have to show that $Q(I) = Q(I')$. By symmetry, it is enough to show that $Q(I) \subseteq Q(J)$. Let $\bar{a} \in Q(I)$. By (3) of Proposition 2, there exists a homomorphism from D'_∞ into I' mapping $\bar{x}$ to $\bar{a}$. It follows by (2) of Proposition 2 that there is a homomorphism from Q into I' mapping $\bar{x}$ to $\bar{a}$, so $\bar{a} \in Q(I')$. □

We are now ready to show that FO is complete for CQ-to-CQ rewriting in the unrestricted case. Let $\mathbf{V}$ be a set of CQ views and $Q(\bar{x})$ be a CQ with free variables $\bar{x}$, both over database schema σ. Recall the sequence $\{D_k, S_k, V_k, V'_k, S'_k, D'_k\}_{k \geq 0}$ defined above for each $\mathbf{V}$ and Q. For each finite instance S over $\sigma_{\mathbf{V}}$, let ϕ_S be the conjunction of the literals $R(t)$ such that $R \in \sigma_{\mathbf{V}}$ and $t \in S(R)$. Let $k > 0$ be fixed. We define a sequence of formulas $\{\varphi_i^k\}_{i=0}^k$ by induction on i, backward from k. First, let

$$\varphi_k^k = \exists \bar{\alpha}_k \phi_{S_k},$$

where $\bar{\alpha}_k$ are the elements in $dom(S_k) - dom(V'_k)$.

For $0 \leq i < k$ let

$$\varphi_i^k = \exists \bar{\alpha}_i [\phi_{S_i} \wedge \forall \bar{\alpha}'_{i+1} (\phi_{S'_{i+1}} \rightarrow \varphi_{i+1}^k)]$$

where $\bar{\alpha}_i$ are the elements in $dom(S_i) - dom(V'_i)$, and $\bar{\alpha}'_{i+1}$ are the elements in $dom(S'_{i+1}) - dom(V_i)$.

We can now show the following (proof omitted).

Theorem 2. *Let $\mathbf{V}$ be a set of CQ views and Q be a CQ, both over database schema σ. If $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$, then there exists $k \geq 0$ such that $Q \overset{\infty}{\Rightarrow}_{\mathbf{V}} \varphi_0^k$.*

Remark. In [14] we showed, using Craig's Interpolation Theorem, that for any set $\mathbf{V}$ of FO views and FO query Q such that $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$ there exists an FO rewriting of Q using $\mathbf{V}$. However, this result is not constructive, as the interpolation theorem itself is not constructive. On the other hand, Theorem 2 provides a constructive algorithm to obtain the rewriting formula. Indeed, assume that $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$

and $\mathbf{V}$ and Q are CQs. Then we know that there is a k such that $\bar{x} \in Q(D'_k)$. Therefore, k can be computed by generating the D'_i until $\bar{x} \in Q(D'_i)$. Once we have k, the formula φ_0^k is easy to compute. Note that Theorem 2 works only for CQ views and queries, while the interpolation argument applies to all FO views and queries. □

Note that the complexity of the formula φ_0^k provided by Theorem 2 for rewriting Q in terms of $\mathbf{V}$ when $\mathbf{V} \overset{\infty}{\twoheadrightarrow} Q$ is intimately related to the minimum k for which $\bar{x} \in Q(D'_k)$. Indeed, φ_0^k has exactly k quantifier alternations. It is therefore of interest to know whether there might be a constant bound on k. In addition to yielding a simpler rewriting with a fixed number of quantifier alternations, this would also provide a decision procedure for unrestricted determinacy: just test that $\bar{x} \in Q(D'_k)$. Recall that it remains open whether determinacy is decidable. Unfortunately, there is no constant bound on k. This is shown below by a generalization of the example used in the proof of Theorem 2.

Example 1. The following shows that for each $k > 0$ there exists a set of CQ views $\mathbf{V}$ and a CQ query $Q(\bar{x})$ such that $\bar{x} \in Q(D'_k)$ but $\bar{x} \notin Q(D'_i)$ for $i < k$.

Let $P_n(x, y)$ be the CQ on a binary relation R stating that there is a path of length n from x to y in R. Consider the set $\mathbf{V}$ of views consisting of

$$\begin{aligned} V_1(x,y) &= \exists \alpha[R(\alpha,x) \wedge P_{k+1}(\alpha,y)], \\ V_2(x,y) &= P_{k+1}(x,y), \\ V_3(x,y) &= P_{k+2}(x,y). \end{aligned}$$

Let $\quad Q(x,y) = \exists a[R(a,x) \wedge P_{2k+1}(a,y)]$.

It can be shown that $\langle x, y\rangle \in Q(D'_k)$, but $\langle x, y\rangle \notin Q(D'_i)$ for $i < k$. Note that the example in the proof of Theorem 1 is a special case of the above with $k = 1$.

4 Well-Behaved Classes of CQ Views and Queries

We next consider restricted classes of views and queries for which CQ remains complete for rewriting. As a consequence, determinacy for these classes is decidable.

4.1 Monadic Views

In this section we consider the special case when the views are monadic. A *monadic* conjunctive query (MCQ) is a CQ whose result is unary. We show the following.

Theorem 3. *(i) CQ is complete for MCQ-to-CQ rewriting.*
(ii) Determinacy is decidable for MCQ views and CQ queries.

Note that (ii) is an immediate consequence of (i), in view of Proposition 1.

Towards proving (i), we first note that it is enough to consider monadic queries. Indeed, we show that for MCQ views, determinacy of arbitrary CQ queries can be reduced to determinacy of MCQ queries.

Proposition 4. *Let* $\mathbf{V}$ *be a set of MCQ views and* $Q(x_1, \ldots, x_k)$ *a CQ query with free variables* $x_1, \ldots, x_k$*, over the same database schema* σ*. Let* $Q_i(x_i)$ *be the MCQ with free variable* x_i*, obtained by quantifying existentially in* Q *all* x_j *for* $j \neq i$*.*

(i) if $\mathbf{V} \twoheadrightarrow Q$ *then* Q *is equivalent to* $\bigwedge Q_i(x_i)$*;*
(ii) if $Q = \bigwedge Q_i(x_i)$ *then* $\mathbf{V} \twoheadrightarrow Q$ *iff* $\mathbf{V} \twoheadrightarrow Q_i(x_i)$ *for* $1 \leq i \leq k$*.*

Proof. Consider (i). Suppose $\mathbf{V} \twoheadrightarrow Q$ and consider $[Q]$. We show that for $i \neq j$, no tuple t_i of $[Q]$ containing x_i is connected to a tuple t_j of $[Q]$ containing x_j in the Gaifman graph of $[Q]$. This clearly proves (i).

Consider instances A and B over σ defined as follows. Let D consist of k elements $\{a_1, \ldots, a_k\}$. Let $A = \{R(a, \ldots, a) \mid R \in \sigma, a \in D\}$, and let B contain, for each $R \in \sigma$ of arity r, the cross-product D^r. Clearly, $\mathbf{V}(A) = \mathbf{V}(B)$ (all views return D in both cases), so $Q(A) = Q(B)$. But $Q(B) = D^k$, so $Q(A) = D^k$. In particular, $\langle a_1, \ldots, a_k \rangle \in Q(A)$. This is only possible if there is no path in the Gaifman graph of $[Q]$ from a tuple containing x_i to one containing x_j for $i \neq j$. Part (ii) is obvious. $\square$

In view of Proposition 4, it is enough to prove that CQ is complete for MCQ-to-MCQ rewriting. As a warm-up, let us consider first the case when $\mathbf{V}$ consists of a single view $V(x)$, and the database schema is one binary relation. Let $Q(x)$ be an MCQ, and suppose that $V \twoheadrightarrow Q$. We show that Q is equivalent to V. Since Q_V is a query, $Q \subseteq V$ (Q cannot introduce domain elements not in V). It remains to show that $V \subseteq Q$. If $Q(D) \neq \emptyset$ it follows by genericity of Q_V that $V(D) \subseteq Q(D)$; however, it is conceivable that $Q(D) = \emptyset$ but $V(D) \neq \emptyset$. Let $A = [V]$ and $B = \{(v, v) \mid v \in V([V])\}$. Clearly, $V(A) = V(B)$. It follows that $Q(A) = Q(B)$. Clearly, $x \in Q(B)$, since $x \in V([V])$ and $(x, x) \in B$. But then $x \in Q(A) = Q([V])$, so there exists a homomorphism from $[Q]$ to $[V]$ fixing x. It follows that $V \subseteq Q$, which completes the proof.

Unfortunately, the above approach for single views does not easily extend to multiple monadic views. Indeed, suppose we have two views V_1, V_2. The stumbling block in extending the proof is the construction of two instances A and B such that $V_1(A) = V_1(B)$ *and* $V_2(A) = V_2(B)$, and forcing the existence of a homomorphism showing that Q can be rewritten using V_1, V_2. As we shall see, the multiple views case requires a more involved construction.

Given two instances D and D' over the same schema, we say that D' *retracts* to D if $D \subseteq D'$ and there is a homomorphism from D' to D fixing the domain of D.

Let $\mathbf{V}$ be a set of MCQ views and $Q(x)$ an MCQ query with free variable x, both over database schema σ, such that $\mathbf{V} \twoheadrightarrow Q$. We wish to show that Q has a CQ rewriting in terms of $\mathbf{V}$. The idea of the proof is the following. Recall, from Proposition 1 (iii), that Q has a CQ rewriting in terms of $\mathbf{V}$ iff $Q_{\mathbf{V}}$ is such a rewriting, where $Q_{\mathbf{V}}$ is the CQ over $\sigma_{\mathbf{V}}$ whose body is $\mathbf{V}([Q])$. This in turn holds iff there is a homomorphism from $[Q]$ into $\mathbf{V}^{-1}(\mathbf{V}([Q]))$ fixing x. To show that this is the case, consider $A_0 = [Q]$ and $B_0 = \mathbf{V}^{-1}(\mathbf{V}([Q]))$.

We show the following key lemma.

Lemma 1. *There exist finite instances A and B over σ such that A retracts to A_0, B retracts to B_0, and $\mathbf{V}(A) = \mathbf{V}(B)$.*

Note that the lemma suffices to conclude the proof of Part (i) of Theorem 3. Indeed, $\mathbf{V}(A) = \mathbf{V}(B)$ implies that $Q(A) = Q(B)$. But $x \in Q(A)$ (since A contains $[Q]$) so $x \in Q(B)$. Since B retracts to B_0 and $x \in dom(B_0)$, this implies that $x \in Q(B_0)$ so there is a homomorphism fixing x from $[Q]$ to $B_0 = \mathbf{V}^{-1}(\mathbf{V}([Q]))$. This establishes (i) of Theorem 3.

Let us fix $\mathbf{V} = \{V_1, \dots, V_k\}$ where the V_i are MCQs over database schema σ. Consider an instance D over the schema $\sigma_{\mathbf{V}}$ of the image of $\mathbf{V}$. For each element a in the domain of D, we denote its *type* in $\mathbf{V}(D)$ as the set $\tau(D, a) = \{i \mid a \in V_i(D)\}$. More precisely, a *type* S of $\sigma_{\mathbf{V}}$ is a subset of $[k] = \{1, \dots, k\}$. A type S is *realized* in D if $\tau(D, c) = S$ for some $c \in dom(D)$. A type S is *realizable* if it is realized in some D. We also denote by $\#(S, D)$ the number of elements $c \in dom(D)$ for which $\tau(D, c) = S$. Note that two instances over $\sigma_{\mathbf{V}}$ are isomorphic iff for all types S, the number of elements of type S in the two instances is the same. Also, if we construct instances A and B over σ such that $\mathbf{V}(A)$ and $\mathbf{V}(B)$ are isomorphic and x has the same type in $\mathbf{V}(A)$ and $\mathbf{V}(B)$, then there are instances A' and B' isomorphic to A and B by isomorphisms preserving x, such that $\mathbf{V}(A') = \mathbf{V}(B')$. Thus, to establish Lemma 1, it is enough to prove the following variant.

Lemma 2. *There exist finite instances A and B over σ such that A retracts to A_0, B retracts to B_0, x has the same type in $\mathbf{V}(A)$ and $\mathbf{V}(B)$, and for each type S, the number of elements of type S in $\mathbf{V}(A)$ is the same as the number of elements of type S in $\mathbf{V}(B)$.*

The proof of the lemma requires some technical development, to which the rest of the section is dedicated. For conciseness, we introduce the following notation. Let A and B be structures, with $a \in dom(A)$ and $b \in dom(B)$. We write $A \to B$ to mean that there is a homomorphism h from A to B, and $Aa \circ\!\!\rightarrow Bb$ if furthermore $h(a) = b$. We also write $Aa \leftarrow\!\circ\!\!\rightarrow Bb$ if $Aa \circ\!\!\rightarrow Bb$ and $Bb \circ\!\!\rightarrow Aa$.

We assume wlog that for all i, V_i is minimized, and that no two V_i are equivalent. We also use the following notation. For each type S, let V_S be the minimized body of the query $\bigwedge_{i \in S} V_i(x_S)$, where x_S is a fresh variable. For technical reasons, we enforce that the V_S for distinct S's have disjoint domains. By slight abuse of notation, we use V_S to denote both the query $\bigwedge_{i \in S} V_i(x_S)$ and its body, as needed.

We construct instances A and B satisfying the requirements of Lemma 2. Consider first A_0 and B_0. Recall that, by construction, B_0 retracts to A_0 and for all $V \in \mathbf{V}$, $V(B_0)$ and $V(A_0)$ agree on $dom(A_0)$. In particular, all elements in A_0, including x, have the same type in $\mathbf{V}(A_0)$ and $\mathbf{V}(B_0)$. Note that, if A retracts to A_0 and B retracts to B_0, then x also has the same type in $\mathbf{V}(A)$ and $\mathbf{V}(B)$.

Observe that we can assume wlog that the body of each view V_i is connected. Otherwise, let W_i be the MCQ whose body is the connected component of V_i containing x_i. Suppose A retracts to A_0 and B retracts to B_0. Recall that B_0

retracts to A_0. If it is not the case that $V_i \rightarrow A_0$ then $V_i(A) = V_i(B) = \emptyset$, and for each S such that $i \in S$, $\#(S, A) = \#(S, B) = 0$. Thus, V_i can be eliminated from $\mathbf{V}$ in our construction. Otherwise, $V_i \rightarrow A_0$, so $V_i(A) = W_i(A)$ and $V_i(B) = W_i(B)$. Thus, V_i can be replaced by W_i. In view of the above, we henceforth assume that all views in $\mathbf{V}$ have connected bodies.

To construct A and B satisfying the requirements of the lemma, we augment A_0 and B_0 with special instances that we call ***bricks***. Let Δ be the set of realizable types S in $[k]$ for which $\#(S, A_0) \neq \#(S, B_0)$, and let $\Delta = \{S_1, \ldots, S_r\}$. For each $i \in [r]$ and $n_i > 0$, let $V_{S_i}^{n_i}$ be the instance constructed from V_{S_i} by replacing x_{S_i} with $n_i + 1$ copies (x_{S_i}, m) of x_{S_i}, $0 \leq m \leq n_i$, connected to the other elements in V_{S_i} in the same way as x_{S_i}. Clearly, $V_{S_i} x_{S_i} \longleftarrow\!\circ\!\longrightarrow V_{S_i}^{n_i}(x_{S_i}, m)$ for $0 \leq m \leq n_i$. A brick Z consists of the disjoint union of the $V_{S_i}^{n_i}$, $1 \leq i \leq r$ for some choice of the n_i. Furthermore, we ensure that $dom(Z)$ is disjoint from $dom(A_0)$ and $dom(B_0)$. Let Z_0 denote the brick where each $n_i = 0$, i.e. Z_0 is isomorphic to $V_{S_1} \cup \ldots \cup V_{S_r}$.

Lemma 3. *Let $\Delta = \{S_1, \ldots, S_r\}$ and Z be a brick. Then for each $i \in [r]$, $\tau(Z, (x_{S_i}, m)) = S_i$, $0 \leq m \leq n_i$. In particular, $\#(S_i, Z) = \#(S_i, Z_0) + n_i$, $1 \leq i \leq r$.*

Proof. Consider $i \in [r]$, and fix m, $0 \leq m \leq n_i$. Clearly, $(x_{S_i}, m) \in V_{S_i}(Z)$, so $S_i \subseteq \tau(Z, (x_{S_i}, m))$. Suppose $j \in \tau(Z, (x_{S_i}, m))$. Then $(x_{S_i}, m) \in V_j(Z)$. Since V_j is connected and $V_{S_i}^{n_i}$ is disjoint from the rest of Z, $(x_{S_i}, m) \in V_j(V_{S_i}^{n_i})$. As noted earlier, $V_{S_i}^{n_i}(x_{S_i}, m) \circ\!\!\longrightarrow V_{S_i} x_{S_i}$. Thus, $x_{S_i} \in V_j(V_{S_i})$, so $V_{S_i} \subseteq V_j$. Since S_i is realizable, this implies that $j \in S_i$. Thus, $\tau(Z, (x_{S_i}, m)) = S_i$. Also, it is easily seen that for each $v \in dom(V_{S_i})$ where $v \neq x_{S_i}$, $\tau(v, Z_0) = \tau(v, Z)$. As a consequence, $\#(S_i, Z) = \#(S_i, Z_0) + n_i$. □

For each $S_i \in \Delta$ let $\alpha_i = \#(S_i, A_0)$ and $\beta_i = \#(S_i, B_0)$. Let Z_A be the brick for which $n_j = \beta_j$, $1 \leq j \leq r$, and Z_B be the brick for which $n_j = \alpha_j$, $1 \leq j \leq r$. Then the following holds.

Lemma 4. *Let $\Delta = \{S_1, \ldots, S_r\}$. Let Z_A and Z_B be the bricks constructed as above, $A = A_0 \cup Z_A$, and $B = B_0 \cup Z_B$. The following hold:*

(i) A retracts to A_0 and B retracts to B_0;
(ii) $\#(S, A) = \#(S, B)$ for all realizable types in $[k]$.

Proof. Consider (i). Let $i \in [r]$. Since $\#(S_i, A_0) \neq \#(S_i, B_0)$, $V_{S_i} \rightarrow A_0$ or $V_{S_i} \rightarrow B_0$, so $V_{S_i} \rightarrow A_0$. Since $V_{S_i}^{n_1} \rightarrow V_{S_i}$, it follows that $V_{S_i}^{n_i} \rightarrow A_0$ for all $i \in [r]$. It follows that $Z_A \rightarrow A_0$ so A retracts to A_0. Similarly, since $V_{S_i} \rightarrow A_0$, $a \in V_{S_i}(A_0)$ for some $a \in dom(A_0)$. By construction, $a \in V_{S_i}(B_0)$, so $V_{S_i} \rightarrow B_0$ for all $i \in [r]$. Similarly to the above, $Z_B \rightarrow B_0$ so B retracts to B_0.

Next consider (ii). Since A retracts to A_0, $\tau(A, a) = \tau(A_0, a)$ for every $a \in dom(A_0)$. Since A_0 and Z_A are disjoint and all view bodies are connected, $\tau(A, a) = \tau(Z_A, a)$ for every $a \in dom(Z_A)$. Thus, $\#(S, A) = \#(S, A_0) + \#(S, Z_A)$ for every type S. Similarly, $\#(S, B) = \#(S, B_0) + \#(S, Z_B)$ for every type S. Let S be a realizable type. Suppose first $S \notin \Delta$, so $\#(S, A_0) = \#(S, B_0)$. It remains to show that $\#(S, Z_A) = \#(S, Z_B)$. By Lemma 3, $\tau(Z_A, (x_i, m)) = S_i$ for each $S_i \in \Delta$. It follows that $\{a \mid a \in Z_A, \tau(Z_A, a) = S\} = \{a \mid a \in Z_0, \tau(Z_0, a) = S\}$. Thus, $\#(S, Z_A) = \#(S, Z_0)$. Similarly, $\#(S, Z_B) = \#(S, Z_0)$. Thus, $\#(S, Z_A) = \#(S, Z_B)$ so $\#(S, A) = \#(S, B)$.

If $S_i \in \Delta$, let $z_i = \#(S_i, Z_0)$. Recall that $\alpha_i = \#(S_i, A_0)$ and $\beta_i = \#(S_i, B_0)$. By the above, $\#(S_i, A) = \#(S_i, A_0) + \#(S, Z_A)$. By Lemma 3, $\#(S_i, Z_A) = \#(S_i, Z_0) + n_i = z_i + \beta_i$, so $\#(S_i, A) = \alpha_i + z_i + \beta_i$. Similarly, $\#(S_i, B) = \#(S_i, B_0) + \#(S_i, Z_B) = \beta_i + z_i + \alpha_i$. It follows that $\#(S_i, A) = \#(S_i, B)$, proving (ii).

This concludes the proof of Lemma 2 and that of Theorem 3.

4.2 Path Views

We have seen above that CQ is complete for MCQ-to-CQ rewriting. Unfortunately, extending this result beyond monadic views is possible only in a very limited way. Recall the example used in the proof of Theorem 1. It shows that even very simple binary views render CQ incomplete. Indeed, the views used there are trees, and differ from simple paths by just a single edge. In this section we show that CQ is complete (and therefore determinacy is decidable) in the case where the database schema is a binary relation R, and $\mathbf{V}$ consists of a single view

$$P_k(x, y) = \exists x_1 \ldots \exists x_{k-1} R(x, x_1) \wedge R(x_1, x_2) \wedge \ldots \wedge R(x_{k-1}, y)$$

where $k \geq 2$, providing the nodes connected by a path of length k. Note that the case where the view in a path of length 1 is trivial since then the view provides the entire database.

We show the following.

Theorem 4. *Let Q be a CQ query over R and $k \geq 2$.*
(i) If $P_k \twoheadrightarrow Q$ then Q has a CQ rewriting in terms of P_k.
(ii) Given a CQ query Q, it is decidable whether $P_k \twoheadrightarrow Q$.

Proof. In view of (iv) of Proposition 1, the existence of a CQ rewriting is decidable, so (ii) follows from (i).

Consider (i). Let Q be a CQ and suppose $P_k \twoheadrightarrow Q$. We show that there is a CQ rewriting of Q using P_k. To this end, we construct two finite instances I and J such that $P_k(I) = P_k(J)$, J consists of several disjoint copies of D'_0, and $\bar{x} \in Q(I)$. This implies that $\bar{x} \in Q(D'_0)$ and (i) follows from Proposition 1. The construction of I and J is done using a careful modification of the chase procedure.

Consider the beginning of the chase sequence as defined in Section 3. Let $D_0 = [Q]$, $S_0 = V(D_0)$, $D'_0 = V^{-1}(S_0)$, $V'_1 = V(D'_0)$, and $S'_1 = V'_1 - S_0$.

Lemma 5. *S_0 and S'_1 have disjoint domains.*

Proof. Suppose that $\langle a, b\rangle \in P_k(D'_0)$ for some $a \in dom(S_0)$ (the case when $b \in dom(S_0)$ is similar). We show that b cannot be in $dom(D'_0) - dom(S_0)$. Since $\langle a, b\rangle \in P_k(D'_0)$, there is a path of length k from a to b. Note that D'_0 has no cycles of length less than k. It follows that either $a = b$ or $d(a, b) = k$. In the first case we are done. If $d(a, b) = k$, by construction of D'_0, b can only be in $dom(S_0)$. □

Next, let us construct a special binary relation M as follows. The domain of M consists of $dom(S'_1)$ and, for each $\alpha \in dom(S'_1)$, new distinct elements $x^\alpha_1, \ldots, x^\alpha_{k-1}$. M has the following edges:

$$\text{for each } \alpha \in dom(S'_1), \text{ the edges} \qquad \alpha \to x^\alpha_1 \to \ldots \to x^\alpha_{k-1};$$

$$\text{and for each } \langle \alpha, \beta\rangle \in S'_1, \text{ an edge} \qquad x^\alpha_{k-1} \to \beta.$$

The following is easily shown by construction of M.

Lemma 6. *Let M be constructed as above. Then $P_k(M)$ consists of k disjoint copies of S'_1. Specifically, $\langle \alpha, \beta\rangle \in S'_1$ iff $\langle \alpha, \beta\rangle \in P_k(M)$ and $\langle x^\alpha_i, x^\beta_i\rangle \in P_k(M)$ for $1 \le i \le k-1$.*

We now use M to construct our desired instances I and J. Let I consist of k disjoint copies of $[Q]$ together with M, and J consist of k disjoint copies of D'_0. Then by Lemma 5, respectively Lemma 6, $P_k(J)$ and $P_k(I)$ both consist of k disjoint copies of S_0 and k disjoint copies of S'_1. By appropriately renaming elements as needed, we obtain $P_k(I) = P_k(J)$. Since $P_k \twoheadrightarrow Q$, $Q(I) = Q(J)$. But $\bar{x} \in Q(I)$ since I contains $[Q]$. It follows that $\bar{x} \in Q(J)$. Since J retracts to D'_0 it follows that $\bar{x} \in Q(D'_0)$ so by Proposition 1, Q has a CQ rewriting in terms of P_k. This completes the proof of Theorem 4. □

5 Conclusion

Several important questions remain unresolved for conjunctive queries. First, decidability of determinacy remains open in both the finite and unrestricted cases. In fact, it remains open whether unrestricted and finite determinacy coincide. Note that if the latter holds, this implies decidability of determinacy. Indeed, then determinacy would be r.e. (using the chase procedure) and co-r.e. (because failure of finite determinacy is witnessed by finite instances).

If it turns out that finite and infinite determinacy are distinct for CQs, then it may be the case that unrestricted determinacy is decidable, while finite determinacy is not. Also, we can obtain FO rewritings whenever **V** determines Q in the unrestricted case, while the best complete language in the finite case remains $\exists$SO $\cap$ $\forall$SO. Since unrestricted determinacy implies finite determinacy, an algorithm for testing unrestricted determinacy could be used in practice as a sound but incomplete algorithm for testing finite determinacy: all positive answers would imply finite determinacy, but the algorithm could return false negatives. When the algorithm accepts, we would also have a guaranteed FO rewriting. Thus, the unrestricted case may turn out to be of practical interest if finite determinacy is undecidable, or to obtain FO rewritings.

While we exibited some classes of CQ views for which CQ remains complete as a rewriting language, we do not yet have a complete characterization of such well-behaved classes. Note that a slight increase in the power of the rewrite language, such as using UCQ or $\mathrm{CQ}^{\neq}$ instead of CQ, does not help. Indeed, a consequence of our results here and in [14] is that there is gap, in the following sense: if CQ is not sufficient to rewrite Q in terms of $\mathbf{V}$, then a non-monotonic language is needed for the rewriting.

Acknowledgments. The authors would like to thank Alin Deutsch and Sergey Melnik for useful discussions related to the material of this paper.

References

1. S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. PODS 1998, 254-263.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley, 1995.
3. F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. PODS 2002, 209-220.
4. F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.* 6(4): 557-575, 1981.
5. D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Lossless regular views. PODS 2002, 247-258.
6. C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, 1977.
7. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. ICDT 1997, 56-70.
8. S. Dar, M.J. Franklin, B. Jonsson, D. Srivastava and M. Tan. Semantic data caching and replacement. VLDB 1996, 330-341.
9. A. Deutsch, L. Popa, and V. Tannen. Physical data independence, constraints and optimization with universal plans. VLDB 1999, 459-470.
10. S. Grumbach, M. Rafanelli, and L. Tininini. Querying aggregate data. PODS 1999, 174-184.
11. S. Grumbach and L. Tininini. On the content of materialized aggregate views. PODS 2000, 47-57.
12. Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. PODS 1995, 95–104.
13. A. Rajaraman, Y. Sagiv, and J.D. Ullman. Answering queries using templates with binding patterns. PODS 1995, 105-112.
14. L. Segoufin and V. Vianu. Views and queries: determinacy and rewriting. PODS 2005, 49-60.
15. J.D. Ullman. Information integration using logical views. ICDT 1997, 19-40.