



HAL
open science

Views and Queries: Determinacy and Rewriting

Alan Nash, Luc Segoufin, Victor Vianu

► **To cite this version:**

Alan Nash, Luc Segoufin, Victor Vianu. Views and Queries: Determinacy and Rewriting. ACM Transactions on Database Systems, 2010, 35 (3), pp.1-41. 10.1145/1806907.1806913 . hal-03664876

HAL Id: hal-03664876

<https://hal.inria.fr/hal-03664876>

Submitted on 11 May 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Views and Queries: Determinacy and Rewriting

Alan Nash
Aleph One LLC

Luc Segoufin
INRIA, ENS-Cachan

Victor Vianu
U.C. San Diego

Abstract

We investigate the question of whether a query Q can be answered using a set \mathbf{V} of views. We first define the problem in information-theoretic terms: we say that \mathbf{V} determines Q if \mathbf{V} provides enough information to uniquely determine the answer to Q . Next, we look at the problem of rewriting Q in terms of \mathbf{V} using a specific language. Given a view language \mathcal{V} and query language \mathcal{Q} , we say that a rewriting language \mathcal{R} is complete for \mathcal{V} -to- \mathcal{Q} rewritings if every $Q \in \mathcal{Q}$ can be rewritten in terms of $\mathbf{V} \in \mathcal{V}$ using a query in \mathcal{R} , whenever \mathbf{V} determines Q . While query rewriting using views has been extensively investigated for some specific languages, the connection to the information-theoretic notion of determinacy, and the question of completeness of a rewriting language, have received little attention. In this paper we investigate systematically the notion of determinacy and its connection to rewriting. The results concern decidability of determinacy for various view and query languages, as well as the power required of complete rewriting languages. We consider languages ranging from first-order to conjunctive queries.

1 Introduction

The question of whether a given set of queries on a database can be used to answer another query arises in many different contexts. Recently, this has been a central issue in data integration, where the problem is framed in terms of query rewriting using views. In the exact local as view (LAV) flavor of the problem, data sources are described by views of a virtual global database. Queries against the global database are answered, if possible, by rewriting them in terms of the views specifying the sources. A similar problem arises in semantic caching: answers to some set of queries against a data source are cached, and one wishes to know if a newly arrived query can be answered using the cached information, without accessing the source. Yet another framework where the same problem arises (only in reverse) is security and privacy. Suppose access to some of the information in a database is provided by a set of public views, but answers to other queries are to be kept secret. This requires verifying that the disclosed views do *not* provide enough information to answer the secret queries.

The question of whether a query Q can be answered using a set \mathbf{V} of views can be formulated at several levels. The most general definition is information theoretic: \mathbf{V} *determines* Q (which we denote $\mathbf{V} \rightarrow Q$) iff $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ implies $Q(D_1) = Q(D_2)$, 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* using \mathbf{V} in a language \mathcal{R} iff there exists some query $R \in \mathcal{R}$ such that $Q(D) = R(\mathbf{V}(D))$ for all databases D . Let us denote this by $Q \Rightarrow_{\mathbf{V}} R$. As usual, there are two flavors to the above definitions, depending on whether database instances are unrestricted (finite or infinite) or restricted to be finite. The finite flavor is the default in all definitions, unless stated otherwise.

What is the relationship between determinacy and rewriting? Suppose \mathcal{R} is a rewriting language. Clearly, if $Q \Rightarrow_{\mathbf{V}} R$ for some $R \in \mathcal{R}$ then $\mathbf{V} \rightarrow Q$. 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} using \mathbf{V} in \mathcal{V} whenever $\mathbf{V} \rightarrow Q$, we say that \mathcal{R} is a *complete* rewriting language for \mathcal{V} -to- \mathcal{Q} rewritings. Clearly, a case of particular interest is when \mathcal{Q} itself is complete for \mathcal{V} -to- \mathcal{Q} rewritings, because then there is no need to extend the query language in order to take advantage of the available views.

Query rewriting using views has been investigated in the context of data integration for some query languages, primarily conjunctive queries (CQs). Determinacy, and its connection to rewriting, has not been investigated in the relational framework. For example, CQ rewritings received much attention in the LAV data integration context, but the question of whether CQ is complete as a rewriting language for CQ views and queries has not been addressed.

In this paper we undertake a systematic investigation of these issues. We consider view languages \mathcal{V} and query languages \mathcal{Q} ranging from first-order logic (FO) to CQ and study two main questions:

- (i) is it decidable whether $\mathbf{V} \rightarrow Q$ 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?

It is easily seen that determinacy becomes undecidable as soon as the query language \mathcal{Q} is powerful enough so that satisfiability of sentences in \mathcal{Q} becomes undecidable. The same holds if validity of sentences in \mathcal{V} is undecidable. Thus, (i) is moot for such languages, in particular for FO queries and views. However, determinacy is also undecidable for much weaker languages. Indeed, we show undecidability even for views and queries expressed as unions of conjunctive queries (UCQs). This is shown by a direct reduction of the word problem for finite monoids, known to be undecidable [25]. The question remains open for CQs, and appears to be quite challenging. Determinacy is shown to be decidable for special classes of CQs, such as Boolean or monadic CQs.

Before summarizing our results on question (ii), we mention two problems that are closely related, and that can be fruitfully used to gain insight into (ii). Suppose $\mathbf{V} \rightarrow Q$. Consider the query $Q_{\mathbf{V}}$ associating to each view answer $\mathbf{V}(D)$ the corresponding query answer $Q(D)$, where D is a database instance. In other words¹, $Q = Q_{\mathbf{V}} \circ \mathbf{V}$. To answer (ii), it is useful to understand the properties of the queries $Q_{\mathbf{V}}$, since this provides information on the rewriting language needed to express them. One useful piece of information is the complexity of computing the answer to $Q_{\mathbf{V}}$ given a view instance $\mathbf{V}(D)$, also known as the *query answering problem*. We occasionally consider the complexity of query answering as a tool for resolving (ii). Other useful information on $Q_{\mathbf{V}}$ concerns properties such as (non-)monotonicity, closure under extensions, etc. Again, we use such information to establish properties required of a language for rewriting Q using \mathbf{V} .

We consider again languages ranging from FO to CQ. We only mention the results for some key combinations of query and view languages, that imply the results for most other combinations. In the unrestricted case, FO turns out to be complete for FO-to-FO rewritings, as a consequence of Craig's Interpolation theorem [13]. Unfortunately this does not extend to the finite case: FO is no longer complete for FO-to-FO rewritings, and indeed the query answering problem for this case is Turing complete. In fact, we show that any language complete for FO-to-FO rewritings must express all computable queries.

¹For mappings f and g such that the image of g is included in the domain of f , the composition $f \circ g$ is the mapping defined by $f \circ g(x) = f(g(x))$.

For views expressed in weaker languages, less powerful rewriting languages are needed. If views are expressed in $\exists\text{FO}$ (existential FO), FO is still not complete for $\exists\text{FO}$ -to-FO rewritings. However, both $\exists\text{SO}$ and $\forall\text{SO}$ (existential and universal second-order logic formulas) are complete for such rewritings. In fact, this is a lower bound: we show that every language complete for $\exists\text{FO}$ -to-FO rewritings must be able to express all queries in $\exists\text{SO} \cap \forall\text{SO}$. The lower bound holds even if views are restricted to UCQs. The proof uses results on the expressive power of implicit definability [28, 22]. It turns out that FO does not become complete as a rewriting language even if queries are in CQ^\neg (CQ with safe negation). This uses the fact, shown by Gurevich, that there exist order-invariant queries defined by FO with access to an order on the domain that are not definable in FO without order (see Exercise 17.27 in [2]).

The case of CQs is of particular interest in the context of answering queries using views, and has been intensively studied. The conventional wisdom has been that CQs must be complete for CQ-to-CQ rewritings. Surprisingly, we show that this is not the case. We do this by exhibiting a set of CQ views \mathbf{V} and query Q such that $\mathbf{V} \twoheadrightarrow Q$ but $Q_{\mathbf{V}}$ is non-monotonic. Thus, no monotonic language can be complete for CQ-to-CQ rewritings. The same is shown for CQ^\neq and UCQs.

Given the above, it is natural to ask whether CQ (and similarly CQ^\neq and UCQ) remain complete in the case when the mapping $Q_{\mathbf{V}}$ is monotonic. We answer this in the affirmative, for CQ as well as CQ^\neq and UCQ.

What is then the rewriting power needed for CQ-to-CQ rewritings? A plausible candidate for a complete language for CQ-to-CQ rewritings would be FO. We are able to show that for all CQ views \mathbf{V} and query Q for which \mathbf{V} determines Q in the *unrestricted* case, there is an effective FO rewriting of Q using \mathbf{V} . The question remains open in the finite case, for which the best known upper bounds remains $\exists\text{SO} \cap \forall\text{SO}$.

Finally, we provide several special 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. Another class consists of Boolean CQ views and queries. Beyond these, 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 a binary view consisting of one simple path.

Related work Answering queries using views arises in numerous contexts including data integration [38], query optimization and semantic caching [16], data warehousing [5], support of physical data independence by describing storage schemas as views [17, 37, 39], 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 using the views, in 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. We also consider the complexity of the

query answering problem, but only in the case when the view determines the query (so the certain and possible answers coincide). Results on equivalent query rewriting using exact views have focused primarily on CQs and UCQs. It is shown in [29] 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 [14]. Answering queries using views in the presence of binding patterns is considered in [34]. Views and queries defined by CQs with arithmetic comparisons over dense orders are considered in [4], where it is shown that the existence of an equivalent rewriting using Datalog with comparisons is decidable. The problem for recursive queries is considered in [18], where it is shown that it is undecidable if a Datalog query can be rewritten using some Datalog program in terms of a set of CQ views. Answering queries using views in semi-structured databases represented by directed labeled graphs is considered in [9, 10, 11]. Here the views and queries are defined by regular path expressions, possibly including inverse edge navigation. In [9] it is shown that the exact rewriting problem is 2EXPSPACE-complete.

The relation of rewriting to the information-theoretic notion of determinacy has received little attention. In [23, 24], 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, [24] 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 [11], the authors consider the notion of *lossless* view with respect to a query, in the context of regular path queries on semi-structured data. A set of views is lossless with respect to a query if for every database, the views are sufficient to answer the query. 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.

In the context of view updates, Bancilhon and Spyratos introduced the notion of *view complement* [6]. The complement of a view is another view so that together they uniquely determine the underlying database. This information-theoretic notion is a special kind of determinacy: a view and its complement determine the identity query on the database.

Suppose a set of views \mathbf{V} does *not* determine a query Q . In this case one is typically interested to compute from a view extent E the *certain* answers to Q . For exact views, the set of certain answers is $\text{cert}_Q(E) = \cap\{Q(D) \mid \mathbf{V}(D) = E\}$. The data complexity of computing $\text{cert}_Q(E)$ from E has been studied by Abiteboul and Dutschka [1] for various query and view languages (for both exact and sound views). The rewriting problem for exact views is to find a query R in some rewriting language \mathcal{R} , such that for every extent E of \mathbf{V} , $R(E) = \text{cert}_Q(E)$. If such R exists for every $\mathbf{V} \in \mathcal{V}$ and $Q \in \mathcal{Q}$, let us say that \mathcal{R} is complete for \mathcal{V} -to- \mathcal{Q} rewritings of certain answers. Clearly, if $V \rightarrow Q$ then for every database D , $Q(D) = \text{cert}_Q(\mathbf{V}(D))$. Thus, for exact views, every rewriting language that is complete for \mathcal{V} -to- \mathcal{Q} rewritings of certain answers must also be complete for \mathcal{V} -to- \mathcal{Q} rewritings according to our definition. In particular, our lower bounds on the expressive power of languages complete for \mathcal{V} -to- \mathcal{Q} rewritings still hold for languages complete for \mathcal{V} -to- \mathcal{Q} rewritings of certain answers. Also, upper bounds shown in [1] on the data complexity of computing certain answers for exact views also apply to the data complexity of computing $Q_{\mathbf{V}}$ in our framework.

For sound views, $\text{cert}_Q(E) = \cap\{Q(D) \mid E \subseteq \mathbf{V}(D)\}$. If $\mathbf{V} \rightarrow Q$, it is easily seen that $\text{cert}_Q(E) = Q_{\mathbf{V}}(E)$ for all E in the image of \mathbf{V} iff $Q_{\mathbf{V}}$ is monotonic, i.e. $\mathbf{V}(D_1) \subseteq \mathbf{V}(D_2)$ implies $Q(D_1) \subseteq Q(D_2)$. In this case, the upper bounds shown in [1] on the data complexity of computing cert_Q for sound views also apply to the data complexity of computing $Q_{\mathbf{V}}$. Likewise, if a language \mathcal{R} can be used to rewrite cert_Q in terms of \mathbf{V} for sound views, and $Q_{\mathbf{V}}$ is monotonic, then \mathcal{R} can be used to rewrite Q in terms of \mathbf{V} in our sense.

In a broader historical context, the topic of this paper is related to the notion of *implicit definability* in logic, first introduced by Tarski [36]. The analog of completeness of a language for view-to-query rewritings is the notion of *completeness for definitions* of a logical language, also defined in [36]. In particular, FO was shown to be complete for definitions by Beth [7] and later by Craig [15], as an application of his Interpolation Theorem. Maarten Marx provides additional historical perspective on problems related to determinacy and rewriting [30].

Recent work following up on the results of [35, 31] has sought to identify additional classes of views and queries that are well behaved with respect to determinacy and rewriting. In [3], Foto Afrati considers the case of *path queries* asking for nodes connected by a path of given length in a binary relation representing a directed graph. It is shown there that determinacy is decidable for path views and queries, and FO rewritings of the path query in terms of the path views can be effectively computed (see Remark 5.3). Taking a different approach, Maarten Marx considers in [30] syntactic restrictions FO and UCQ yielding *packed* FO and UCQ, and shows for these fragments decidability of determinacy and completeness for rewritings (see Remarks 3.11, 5.23).

The present paper is the extended version of the conference publications [35, 31].

Organization After introducing some basic concepts in Section 2, we discuss determinacy and rewriting for FO queries and views in Section 3. We then proceed with unions of conjunctive queries and views in Section 4, and conjunctive queries and views in Section 5. We end with brief conclusions.

2 Preliminaries

We begin with some basic definitions and notation. Unless otherwise indicated, we follow standard terminology (see [2] and [27]).

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*. We assume fixed an infinite domain \mathbf{dom} of values. A database instance D over σ associates a relation $D(R)$ of appropriate arity with values from \mathbf{dom} to each relation symbol R in σ (true/false for propositions). The active domain of an instance D consists of the set of elements in \mathbf{dom} occurring in D and is denoted $\text{adom}(D)$. 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. In terms of classical logic, a database instance is a relational structure whose universe is the active domain of the instance. In some proofs, we will need to apply known results of model theory that apply to classical relational structures, whose universe may strictly include the active domain. In this case, we will explicitly use the term *relational structure*. 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 \mathbf{dom} (e.g., see [2]). We assume familiarity with the query languages in Figure 1. As usual for two queries q and q' , we denote by $q \subseteq q'$ the fact that over all instances D , $q(D)$ is always a subset of $q'(D)$. As usual,

NOTATION	LANGUAGE
FO	first-order logic over relations
\exists FO	existential FO
\exists SO	existential second-order logic
\forall SO	universal second-order logic
CQ	conjunctive queries without $=$, \neq , constants
UCQ	unions of conjunctive queries
(U)CQ $^=$	(U)CQs extended with $=$, \neq
(U)CQ $^\neq$	

Figure 1: Query languages used in the paper

unless otherwise specified, constant values from **dom** may be used in queries. These are not part of the schema, and are always interpreted as themselves. This differs from constants in classical logic, which are part of the vocabulary and are interpreted as arbitrary values from the universe of the structure. Given a query φ , we define its active domain $adom(\varphi)$ to be the set of constant values used in φ . Given a database instance D , we denote $adom(D, \varphi) = adom(D) \cup adom(\varphi)$. The semantics of φ over an instance D is defined with respect to the universe $adom(D, \varphi)$. In particular, if φ is a first-order logic formula, all variables (free and quantified) range over $adom(D, \varphi)$. If φ is in second-order logic, its first-order variables range over $adom(D, \varphi)$ and its second-order variables range over relations over $adom(D, \varphi)$. The semantics for relational structures is defined as usual with respect to the specified universe.

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 $\mathcal{I}(\sigma) \rightarrow \mathcal{I}(\{V\})$ 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 $\mathbf{V} \rightarrow Q$, iff for all $D_1, D_2 \in \mathcal{I}(\sigma)$, if $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ then $Q(D_1) = Q(D_2)$. Let R be a query over $\mathcal{I}(\sigma_{\mathbf{V}})$. We say that R is a *rewriting* of Q using \mathbf{V} 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} . The notions of determinacy and rewriting are related as there is a query R such that $Q \Rightarrow_{\mathbf{V}} R$ iff $\mathbf{V} \rightarrow Q$.

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

The notions of determinacy and rewriting come in two flavors: finite and unrestricted, depending on whether instances are restricted to be finite. We always assume the finite variant unless explicitly stated otherwise.

We next present some basic but useful observations about determinacy and rewriting.

Proposition 2.1 *Let \mathcal{Q} and \mathcal{V} be languages such that satisfiability of sentences in \mathcal{Q} is undecidable or validity of sentences in \mathcal{V} is undecidable. Then it is undecidable whether $\mathbf{V} \rightarrow Q$, where Q is in \mathcal{Q} and \mathbf{V} is a set of views defined in \mathcal{V} .*

Proof: Suppose first that satisfiability of sentences in \mathcal{Q} is undecidable. Let φ be a sentence in \mathcal{Q}

over database schema σ . Consider the database schema $\sigma \cup \{R\}$ where R is unary. Let \mathbf{V} be empty and $Q = \varphi \wedge R(x)$. Clearly, $\mathbf{V} \rightarrow Q$ iff φ is unsatisfiable.

Next, suppose validity of sentences in \mathcal{V} is undecidable. Let φ be a sentence in \mathcal{V} over schema σ . Consider the database schema $\sigma \cup \{R\}$ with R unary, and \mathbf{V} consisting of the view $\varphi \wedge R(x)$. Let Q consist of the query $R(x)$. Clearly, $\mathbf{V} \rightarrow Q$ iff φ is valid. \square

Corollary 2.2 *If \mathcal{V} is FO or \mathcal{Q} is FO, it is undecidable whether $\mathbf{V} \rightarrow Q$ for views \mathbf{V} in \mathcal{V} and queries Q in \mathcal{Q} .*

In looking for fragments of FO for which determinacy might be decidable, it is tempting to reduce determinacy to satisfiability testing. This can be done as follows. Let σ be a database schema and \mathbf{V} and Q be views and a query over σ . Let σ_1 and σ_2 be two disjoint copies of σ , and \mathbf{V}_i, Q_i ($i = 1, 2$) versions of \mathbf{V} and Q operating on σ_1 and σ_2 . Consider the FO sentence φ over $\sigma_1 \cup \sigma_2$:

$$\forall \bar{x}(\mathbf{V}_1(\bar{x}) \leftrightarrow \mathbf{V}_2(\bar{x})) \wedge \exists \bar{y}(Q_1(\bar{y}) \wedge \neg Q_2(\bar{y})).$$

Clearly, $\mathbf{V} \rightarrow Q$ iff φ is not finitely satisfiable. Unfortunately, even for CQ views and queries, φ does not belong to an FO fragment known to have a decidable satisfiability problem [8]. Thus, we need to take advantage of the finer structure of the query languages we consider in order to settle decidability of determinacy.

Finally, suppose $\mathbf{V} \rightarrow Q$. Let then $Q_{\mathbf{V}}$ be the mapping associating to every instance $\mathbf{V}(D)$ in the image of \mathbf{V} the corresponding $Q(D)$. Note that $Q_{\mathbf{V}}$ is defined only on the image of \mathbf{V} . Therefore when we say “ $Q_{\mathbf{V}}$ is computable” we mean that for any instance S in the image of \mathbf{V} , $Q_{\mathbf{V}}(S)$ is computable. Similarly, genericity of $Q_{\mathbf{V}}$ only applies to the image of \mathbf{V} .

Proposition 2.3 *Suppose \mathbf{V} and Q are computable views and queries without constants, over database schema σ , and $\mathbf{V} \rightarrow Q$. Let $Q_{\mathbf{V}}$ be the mapping associating to every instance in the image of \mathbf{V} the corresponding value of Q . Then $Q_{\mathbf{V}}$ is generic and computable. In particular, for all $D \in \mathcal{I}(\sigma)$: (i) $\text{adom}(Q(D)) \subseteq \text{adom}(\mathbf{V}(D))$, and (ii) every permutation of \mathbf{dom} that is an automorphism of $\mathbf{V}(D)$ is also an automorphism of $Q(D)$.*

Proof: Let f be a permutation of \mathbf{dom} . Let $S = \mathbf{V}(D)$. Then $Q_{\mathbf{V}}(f(S)) = Q_{\mathbf{V}}(f(\mathbf{V}(D))) = Q_{\mathbf{V}}(\mathbf{V}(f(D))) = Q(f(D)) = f(Q(D)) = f(Q_{\mathbf{V}}(\mathbf{V}(D))) = f(Q_{\mathbf{V}}(S))$. Thus, $Q_{\mathbf{V}}$ is generic, and (i) and (ii) are immediate consequences. To compute $Q_{\mathbf{V}}$ on input S in the image of \mathbf{V} , find some D over σ for which $Q(D) = S$, which is guaranteed to exist. Then compute $Q(D)$. \square

Remark 2.4 *Suppose \mathbf{V} and Q are as in Proposition 2.3 and $Q_{\mathbf{V}}$ halts on every instance in the image of \mathbf{V} . One may wonder if $Q_{\mathbf{V}}$ can be extended to a computable, halting query on all of $\mathcal{I}(\sigma_{\mathbf{V}})$. A simple recursion-theoretic argument shows that this is not the case. In fact, there exist \mathbf{V} and Q both in FO for which $\mathbf{V} \rightarrow Q$ but $Q_{\mathbf{V}}$ cannot be extended to a halting computable query on all of $\mathcal{I}(\sigma_{\mathbf{V}})$.*

3 FO Queries and Views

In this section we investigate determinacy and rewriting for FO queries and views. Determinacy is clearly undecidable for FO in view of Corollary 2.2. We therefore focus on rewriting. Is FO complete

for FO-to-FO rewritings? The answer turns out to be radically different in the unrestricted and the finite cases.

We begin by considering the unrestricted variant. Recall that in the unrestricted case, database instances are allowed to be arbitrary (finite or infinite). We denote the fact that \mathbf{V} determines Q for unrestricted instances by $\mathbf{V} \xrightarrow{\infty} Q$.

Theorem 3.1 below follows from the work of Beth and Craig [7, 15]. We include a direct proof here for the sake of self containment.

Theorem 3.1 *In the unrestricted case, FO is complete for FO-to-FO rewritings.*

Proof: The general idea of the proof is simple, if $\mathbf{V} \xrightarrow{\infty} Q$ then $Q_{\mathbf{V}}$ does not depend on the database D but only on $\mathbf{V}(D)$. In other words $Q_{\mathbf{V}}$ is invariant under the choice of D , assuming $\mathbf{V}(D)$ remains fixed. This property can be specified in FO. Then, using Craig interpolation theorem, all reference to D can be removed from the specification above, yielding a first-order formula which turns out to be the desired FO rewriting.

In developing the details, some care is needed because Craig's interpolation theorem applies to relational structures while the notion of determinacy involves database instances. Recall from Section 2 that a relational structure explicitly provides a universe that may be larger than the active domain of the same structure viewed as a database instance. In order to cope with this technical difference, we will explicitly restrict all quantifications to the active domain in the formula expressing that \mathbf{V} determines Q , and use an extension of Craig's interpolation theorem that yields an interpolant that also quantifies over the active domain.

In this proof we will consider extensions of relational schemas with a finite set of constant symbols. If the schema contains constants, an instance over the schema provides values to the constants in addition to the relations. For FO sentences φ, ψ over the same schema, we use the notation $\varphi \models \psi$ to mean that every structure (finite or infinite) satisfying φ also satisfies ψ . We also use the notation $\exists(\forall)\bar{x} \in U$ for a unary symbol U and vector of variables $\bar{x} = x_1 \dots x_m$ as shorthand for $\exists(\forall)x_1 \in U \dots \exists(\forall)x_m \in U$.

Let σ be the database schema, $\sigma_{\mathbf{V}} = \{V_1, \dots, V_k\}$ be the output schema of \mathbf{V} , and ϕ_1, \dots, ϕ_k be the corresponding queries defining \mathbf{V} . Suppose $\mathbf{V} \xrightarrow{\infty} Q$ where \mathbf{V} and Q are FO views and queries over schema σ .

Let σ_1, σ_2 be disjoint copies of σ . Let n be the number of free variables of Q and let \bar{c} be n constant symbols. Let U_1, U_2, U_V be three new unary symbols. We now consider relational structures over the schema $\tau = \sigma_1 \cup \sigma_2 \cup \sigma_{\mathbf{V}} \cup \{\bar{c}, U_1, U_2, U_V\}$. The intended meaning of a structure over τ is that its projection over σ_1 and σ_2 form two database instances having the same view which is the projection of the structure over $\sigma_{\mathbf{V}}$. Moreover U_1, U_2 and U_V are respectively the active domain of relations occurring in σ_1, σ_2 and $\sigma_{\mathbf{V}}$.

For $i = 1$ or $i = 2$ we denote by Q^i the query obtained from Q by replacing every symbol from σ with its copy in σ_i , by enforcing that every free variable belongs to U_i and by restricting each quantification to range over U_i . For all $j \leq k$ we define similarly copies ϕ_j^1 and ϕ_j^2 of ϕ_j by restricting the range of quantification in the appropriate way. Finally we denote by α_{adom}^i the formula stating that the elements of U_i appear in some relation of σ_i .

For $i = 1$ or 2 , consider the following formula which will be denoted by $\mathbf{V}^i = \sigma_{\mathbf{V}}$ in the sequel:

$$\bigwedge_{j=1, \dots, k} \forall \bar{x} \in U_V \quad \phi_j^i(\bar{x}) \leftrightarrow V_j(\bar{x}) \quad \wedge$$

$$\bigwedge_{j=1, \dots, k} \forall \bar{x} \in U_i \phi_j^i(\bar{x}) \rightarrow \bar{x} \in U_V \quad \wedge \quad \forall x \in U_V \exists \bar{y} \in U_i \bigvee_{j=1, \dots, k} \phi_j^i(\bar{y}, x)$$

This formula says that (i) when restricted to the elements of U_V , V_j is indeed the j th view of the database i , (ii) the elements of U_V contains at least the elements of the view of the restriction of the database i to elements of U_i , and (iii) U_V contains only elements of the view of database i restricted to elements U_i . Altogether this says that the restriction of V_j to elements of U_V is exactly the j th view of the restriction to elements of U_i of the database i .

The following result is now a simple consequence of the fact that $\mathbf{V}(D)$ determines Q .

Claim 3.2

$$(\alpha_{\text{adom}}^1 \wedge \mathbf{V}^1 = \sigma_{\mathbf{V}} \wedge Q^1(\bar{c})) \models [(\alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}) \rightarrow Q^2(\bar{c})] \quad (1)$$

Proof: Let \mathcal{D} be a relational structure over τ such that $\mathcal{D} \models (\alpha_{\text{adom}}^1 \wedge \mathbf{V}^1 = \sigma_{\mathbf{V}} \wedge Q^1(\bar{c}))$. We need to show that $\mathcal{D} \models [(\alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}) \rightarrow Q^2(\bar{c})]$. For this we further assume that $\mathcal{D} \models \alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}$ and we show that $\mathcal{D} \models Q^2(\bar{c})$.

For $i = 1$ or $i = 2$, let D^i be the database instance over σ constructed from \mathcal{D} by keeping only the restriction of the relations of σ^i to the elements of U_i . Let S be the database instance constructed from \mathcal{D} by keeping only the restriction of the relations of $\sigma_{\mathbf{V}}$ to the elements of U_V .

By definition of $\mathbf{V}^1 = \sigma_{\mathbf{V}}$ and $\mathbf{V}^2 = \sigma_{\mathbf{V}}$ we have $\mathbf{V}(D^1) = S = \mathbf{V}(D^2)$. Let \bar{a} be the interpretation in \mathcal{D} of the constants \bar{c} . As the free variables of Q^1 were restricted to elements of U_1 , from $\mathcal{D} \models Q^1(\bar{c})$ we have $D^1 \models Q(\bar{a})$. Hence, because $\mathbf{V} \stackrel{\infty}{\approx} Q$ we must have $D^2 \models Q(\bar{a})$. By construction of Q^2 this implies that $\mathcal{D} \models Q^2(\bar{c})$ as required. \square

Notice now that the formula of the left-hand side of (1) contains only symbols from $\sigma_1 \cup \sigma_{\mathbf{V}} \cup \{\bar{c}, U_1, U_v\}$ and all quantifications are restricted to elements either in U_1 or in U_V , while the formula of the right-hand side contains only symbols from $\sigma_2 \cup \sigma_{\mathbf{V}} \cup \{\bar{c}, U_2, U_V\}$ and all quantifications are restricted to elements either in U_2 or in U_V . Hence, by the relativized Craig's Interpolation Theorem proved by Martin Otto [32] there exists a sentence $\theta(\bar{c})$ that uses only symbols from $\sigma_{\mathbf{V}} \cup \{\bar{c}, U_V\}$ where all quantifications are restricted to elements of U_V and such that:

$$(\alpha_{\text{adom}}^1 \wedge \mathbf{V}^1 = \sigma_{\mathbf{V}} \wedge Q^1(\bar{c})) \models \theta(\bar{c}) \quad (2)$$

$$\theta(\bar{c}) \models [(\alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}) \rightarrow Q^2(\bar{c})] \quad (3)$$

Note first that as quantifications in θ are restricted to elements of U_V we can assume that θ does not contain any atom of the form $U_V(y)$ where y is a quantified variable (otherwise we replace it by TRUE). Let now Θ be the formula obtained from θ by ignoring the restriction of the quantifications to U_V , by replacing \bar{c} with free variables \bar{x} and by replacing all terms of the form $U_V(c)$ where c is a constant by TRUE.

We now show that Θ is a rewriting of Q using \mathbf{V} .

Let D be a database instance over σ and let S be $\mathbf{V}(D)$. We need to show that for all tuples \bar{a} of elements of D we have $\bar{a} \in Q(D)$ iff $\bar{a} \in \Theta(S)$.

Let \mathcal{D} be the relational structure whose universe is the active domain of D and whose relations of σ_1 , σ_2 and $\sigma_{\mathbf{V}}$ are respectively the corresponding relations of D and S , where \bar{c} is interpreted as \bar{a} and where U_1, U_2 and U_V are the active domains of relations respectively in σ_1 , σ_2 and $\sigma_{\mathbf{V}}$.

Assume first that $\bar{a} \in Q(D)$. By construction of \mathcal{D} and because of the restrictions on the quantifications we have $\mathcal{D} \models (\alpha_{\text{adom}}^1 \wedge \mathbf{V}^1 = \sigma_{\mathbf{V}} \wedge Q^1(\bar{c}))$. From (2) we have $\mathcal{D} \models \theta(\bar{c})$. By construction of Θ , this implies that $\bar{a} \in \Theta(S)$.

Assume now that $\bar{a} \in \Theta(S)$. By construction of \mathcal{D} and Θ this implies that $\mathcal{D} \models \theta(\bar{c})$. Hence from (3) we have $\mathcal{D} \models [(\alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}) \rightarrow Q^2(\bar{c})]$. Now by construction $\mathcal{D} \models \alpha_{\text{adom}}^2 \wedge \mathbf{V}^2 = \sigma_{\mathbf{V}}$. Therefore $\mathcal{D} \models Q^2(\bar{c})$. As Q^2 restricts its quantification to U_2 which is the active domain of D this implies $\bar{a} \in Q(D)$. \square

The proof of Theorem 3.1 relies crucially on Craig’s Interpolation Theorem. This fundamental result in model theory holds for unrestricted instances but fails in the finite case [19]. Indeed, Theorem 3.1 does not hold in the finite case, as shown next.

Example 3.3 Let σ be a database schema and σ_{\leq} the extension of σ with a binary relation “ \leq ”. Let $\varphi(\leq)$ be an FO sentence over σ_{\leq} . Let ψ be the FO sentence checking that \leq is a linear order. Now consider the set of views \mathbf{V} with $\sigma_{\mathbf{V}} = \sigma \cup \{R_{\psi}\}$, where R_{ψ} is zero-ary. \mathbf{V} returns the value of ψ (in R_{ψ}) and the content of R for each $R \in \sigma$. Let Q_{φ} be the query $\psi \wedge \varphi(\leq)$. It is easy to verify that if φ is order invariant (i.e. its answer is independent of the choice of the order relation \leq) then $\mathbf{V} \twoheadrightarrow Q_{\varphi}$. If FO is complete for FO-to-FO rewritings in the finite, then there exists an FO query θ over $\sigma_{\mathbf{V}} = \sigma \cup \{R_{\psi}\}$ such that Q_{φ} is equivalent to θ . On ordered finite instances, this means that $\varphi(\leq)$ is equivalent to $\theta(\text{true}/R_{\psi})$, obtained by replacing the proposition R_{ψ} with true in θ . However, Gurevich has shown that there exist order-invariant FO queries $\varphi(\leq)$ expressing queries that are not finitely definable in FO (see Exercise 17.27 in [2]). This is a contradiction.

The following characterizes the expressive power needed for completeness wrt FO-to-FO rewriting in the finite case.

Theorem 3.4 *If \mathcal{R} is complete for FO-to-FO rewritings (for finite instances) then \mathcal{R} expresses all computable queries.*

Proof: Let M be an arbitrary Turing machine expressing a total, computable generic query q whose inputs and outputs are graphs (the argument can be easily extended to arbitrary schemas). More precisely, consider a directed graph G with sets of nodes $\text{adom}(G)$, and \leq a total order on the nodes. We consider a standard encoding $\text{enc}_{\leq}(G)$ of G as a string in $\{0, 1\}^*$ of length $|\text{adom}(G)|^2$ whose $\langle i, j \rangle$ -th position in lexicographic order is 1 iff $\langle a_i, a_j \rangle \in E$ where a_i and a_j have rank i resp. j with respect to the order \leq . M computes q iff M on input $\text{enc}_{\leq}(G)$ halts with output $\text{enc}_{\leq}(q(G))$, for every total order \leq on $\text{adom}(G)$. Consider the database schema $\sigma = \{R_1, R_2, \leq, T\}$ where R_1, R_2 and \leq are binary, and T is ternary. The intended meaning is that R_1 is the input graph, \leq is a total order over some set $D \supseteq \text{adom}(R_1)$ with $\text{adom}(R_1)$ as initial elements, and T represents a halting computation of M on input $\text{enc}_{\leq}(R_1)$, with output $\text{enc}_{\leq}(R_2)$. More specifically, $T(i, j, c)$ holds if in the i -th configuration of M , the content of the j -th tape cell is c (the position of the head and the state are encoded in tape symbols). In the encoding, i is represented by the element of rank i in the ordering \leq . Using standard techniques (see for instance [2]) one can construct an FO sentence φ_M over σ stating that \leq is indeed a total order including $\text{adom}(G)$ as initial elements, and that T is a correct representation of a halting computation of M on input $\text{enc}_{\leq}(R_1)$, with output $\text{enc}_{\leq}(R_2)$. Let \mathbf{V} be the view with $\sigma_{\mathbf{V}} = \{S_1\}$ defined by $Q_{S_1} = \varphi_M \wedge R_1(x, y)$, and Q be the query $\varphi_M \wedge R_2(x, y)$.

We claim that $\mathbf{V} \rightarrow Q$ and $Q = q \circ \mathbf{V}$. Indeed if \mathbf{V} is empty then either φ_M is false or R_1 is empty. In both cases Q is empty (note that by genericity $q(\emptyset) = \emptyset$) and so $Q = q \circ \mathbf{V}$. If \mathbf{V} is not empty then φ_M holds and \mathbf{V} returns R_1 . As φ_M holds, Q equals $R_2 = q(R_1)$. Thus, $Q = q \circ \mathbf{V}$. \square

In terms of complexity, Theorem 3.4 says that there is no complexity bound for the query answering problem for FO views and queries. It is clearly of interest to identify cases for which one can show bounds on the complexity of query answering. We next show that for views restricted to \exists FO, the complexity of query answering is $\mathbf{NP} \cap \text{co-}\mathbf{NP}$. The bound is tight, even when the views are further restricted to UCQs. This will be shown using a logical characterization of $\mathbf{NP} \cap \text{co-}\mathbf{NP}$ by means of implicit definability [28, 22].

Recall that Fagin's Theorem shows that \exists SO expresses \mathbf{NP} , and \forall SO expresses $\text{co-}\mathbf{NP}$ (see [27]). Thus, to show the $\mathbf{NP} \cap \text{co-}\mathbf{NP}$ complexity bound it is sufficient to prove the completeness of \exists SO and \forall SO for \exists FO-to-FO rewritings. We do this next.

Theorem 3.5 *\exists SO and \forall SO are both complete for \exists FO-to-FO rewritings.*

Proof: Let \mathbf{V} be a view from $\mathcal{I}(\sigma)$ to $\mathcal{I}(\sigma_{\mathbf{V}})$ defined in \exists FO. Let S be an instance over $\mathcal{I}(\sigma_{\mathbf{V}})$ in the image of \mathbf{V} . We start with the following lemma showing that among the database instances D such that $\mathbf{V}(D) = S$ there is one of size polynomial in $|S|$.

Lemma 3.6 *Let \mathbf{V} be defined in \exists FO and k be the maximum number of variables and constants in a view definition of \mathbf{V} , in prenex form. If $S \in \mathcal{I}(\sigma_{\mathbf{V}})$ and S is in the image of \mathbf{V} then there exists $D \in \mathcal{I}(\sigma)$ such that $\mathbf{V}(D) = S$ and $|\text{adom}(D)| \leq k|\text{adom}(S)|^k$.*

Proof: Let S, \mathbf{V} be as in the statement. Let $D' \in \mathcal{I}(\sigma)$ be such that $\mathbf{V}(D') = S$. Consider $V \in \mathbf{V}$ with corresponding view Q_V and let \bar{c} be a tuple in $S(V)$ (its arity is at most k). Each such \bar{c} is witnessed by an assignment $\theta_{\bar{c}}$ extending \bar{c} to the existentially quantified variables of Q_V . Let A be the set of all elements of D' occurring in $\theta_{\bar{c}}$ for some tuple \bar{c} of S . Let D be the restriction of D' to A . By construction, each assignment $\theta_{\bar{c}}$ still witnesses \bar{c} . Therefore, $S \subseteq \mathbf{V}(D)$. Because \mathbf{V} is in \exists FO, \mathbf{V} is closed under extension, so $\mathbf{V}(D) \subseteq \mathbf{V}(D') = S$. Altogether $\mathbf{V}(D) = S$ and $\text{adom}(D) = A$ has size bounded by $k|\text{adom}(S)|^k$. \square

Assume now that Q is in FO and that $\mathbf{V} \rightarrow Q$. Recall that $Q_{\mathbf{V}}$ is the mapping associating to every instance S in the image of \mathbf{V} the corresponding value of Q . Recall that, by Proposition 2.3, $\text{adom}(Q_{\mathbf{V}}(S)) \subseteq \text{adom}(S)$. By Lemma 3.6 a non-deterministic polynomial algorithm for checking whether a tuple \bar{c} over $\text{adom}(S)$ is in $Q_{\mathbf{V}}(S)$ goes as follows: guess a database instance D over σ of size polynomial in $|S|$, check that $\mathbf{V}(D) = S$ and check that $\bar{c} \in Q(D)$. Also from Lemma 3.6 we have the following universal polynomial algorithm: For all database instances D of size polynomial in $|S|$, if $\mathbf{V}(D) = S$, check that $\bar{c} \in Q(D)$. The first algorithm is in \mathbf{NP} while the second is in $\text{co-}\mathbf{NP}$. By Fagin's theorem this implies that $\bar{c} \in Q_{\mathbf{V}}(S)$ can be expressed by both \exists SO and \forall SO formulas $\varphi(\bar{c})$. By universality of constants, it follows that there are \exists SO and \forall SO formulas $\varphi(\bar{x})$ defining $Q_{\mathbf{V}}$. \square

We next show that Theorem 3.5 is tight, i.e. every rewriting language complete for \exists FO-to-FO rewritings must be able to express all properties in \exists SO \cap \forall SO. In fact, the lower bound holds even for UCQ-to-FO rewritings.

Theorem 3.7 *Let τ be a schema. For every query q on instances over τ , definable in $\exists SO \cap \forall SO$, there exists a set \mathbf{V} of UCQ views and an FO sentence Q such that $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ defines q .*

Proof: We use a result of [28] (see also [22]) on the expressive power of implicit definability over finite instances. We briefly recall the result and related definitions. Let τ be a database schema. Let q be a query over τ returning a k -ary relation. Such a query q is said to be implicitly definable over τ if there exists a schema $\tau' = \tau \cup \{T, \bar{S}\}$, where T is k -ary and \bar{S} are new relation symbols, and a FO(τ') sentence $\varphi(T, \bar{S})$ such that (i) for all $D \in \mathcal{I}(\tau)$ there exists a sequence \bar{S} of relation over $\text{adom}(D)$ such that $D \models \varphi(q(D), \bar{S})$ and (ii) for all relations T and \bar{S} over $\text{adom}(D)$ we have $D \models \varphi(T, \bar{S})$ implies $T = q(D)$.

The set of queries implicitly definable over τ is denoted by $\text{GIMP}(\tau)$. We use the following known result [28, 22]: $\text{GIMP}(\tau)$ consists of all queries whose data complexity is $\mathbf{NP} \cap \text{co-NP}$. In view of Fagin's Theorem, this yields:

Theorem 3.8 [28, 22] *$\text{GIMP}(\tau)$ consists of all queries over τ expressible in $\exists SO \cap \forall SO$.*

Thus, to establish Theorem 3.7 it is enough to show that every language complete for UCQ-to-FO rewritings must express every query in $\text{GIMP}(\tau)$.

Let τ be a schema and q a query in $\text{GIMP}(\tau)$. Thus, there exists an FO sentence $\varphi(T, \bar{S})$ over $\tau' = \tau \cup \{T, \bar{S}\}$ such that q is implicitly defined over τ by $\varphi(T, \bar{S})$. We may assume wlog that $\varphi(T, \bar{S})$ uses only \wedge, \neg, \exists .

We construct Q and \mathbf{V} as follows. We first augment τ' with some new relation symbols. For each subformula $\theta(\bar{x})$ of φ with n free variables consider two relation symbols R_θ and \bar{R}_θ of arity n . In particular, R_φ is a proposition. Let σ be the set of such relations, and $\tau'' = \tau' \cup \sigma$. Given $D \in \mathcal{I}(\tau')$, the intent is for R_θ to contain $\theta(D(\tau'))$ and for \bar{R}_θ to be the complement of R_θ (for technical reasons, \bar{R}_θ is needed even when $-\theta$ is not a subformula of φ). The interest of the auxiliary relations is that φ can be checked by verifying that each R_θ has the expected content using a straightforward structural induction on θ . This is done by checking simple connections between the relations in σ :

- (1) $\bar{R}_\theta = \text{adom}(D)^k - R_\theta, R_{-\theta} = \bar{R}_\theta,$
- (2) $R_{\theta_1 \wedge \theta_2}(\bar{x}, \bar{y}, \bar{z}) = R_{\theta_1}(\bar{x}, \bar{y}) \wedge R_{\theta_2}(\bar{y}, \bar{z}),$ and
- (3) $R_{\exists x \theta(x, \bar{y})}(\bar{y}) = \exists x R_{\theta(x, \bar{y})}(x, \bar{y}).$

Let ψ be an FO(τ'') formula which checks that all relations $R_\theta, \bar{R}_\theta \in \sigma$ satisfy (1)-(3) above. Let Q be the query $\psi \wedge \varphi(T, \bar{S}) \wedge T(\bar{x})$. Thus, for $D \in \mathcal{I}(\tau'')$ satisfying φ and for which additionally the relations of $D(\sigma)$ satisfy (1)-(3), $Q(D)$ returns $D(T) = q(D(\tau))$.

We now define a set \mathbf{V} of UCQ views. On input $D \in \mathcal{I}(\tau'')$, a first subset \mathbf{V}_τ of \mathbf{V} simply returns $D(\tau)$. In particular, \mathbf{V}_τ provides the active domain A of D . A second set of views, \mathbf{V}_σ , allows verifying whether the relations in σ satisfy (1)-(3) without providing any information on $D(T)$. Specifically, \mathbf{V}_σ contains the following:

- (i) Views allowing to check (1). To verify that \bar{R}_θ is the complement of R_θ , we use two views: the first is defined by $R_\theta(\bar{x}) \wedge \bar{R}_\theta(\bar{x})$ and the second $R_\theta(\bar{x}) \vee \bar{R}_\theta(\bar{x})$. If k is the arity of R_θ , (1) holds iff the first view returns \emptyset and the second returns A^k .

- (ii) Views allowing to check (2). Let $\theta(\bar{x}, \bar{y}, \bar{z}) = \theta_1(\bar{x}, \bar{y}) \wedge \theta_2(\bar{y}, \bar{z})$, where $\bar{x}, \bar{y}, \bar{z}$ are disjoint sequences of free variables. We use three views. The first is defined by $R_{\theta_1}(\bar{x}, \bar{y}) \wedge R_{\theta_2}(\bar{y}, \bar{z}) \wedge \bar{R}_{\theta}(\bar{x}, \bar{y}, \bar{z})$, the second by $R_{\theta}(\bar{x}, \bar{y}, \bar{z}) \wedge \bar{R}_{\theta_1}(\bar{x}, \bar{y})$, and the third by $R_{\theta}(\bar{x}, \bar{y}, \bar{z}) \wedge \bar{R}_{\theta_2}(\bar{y}, \bar{z})$. Note that (2) holds for θ iff the three views return the empty set.
- (iii) Views allowing to check (3). Let $\theta(\bar{y}) = \exists x \theta_1(x, \bar{y})$. We use two views. The first is defined by $\exists x R_{\theta_1}(x, \bar{y}) \wedge \bar{R}_{\theta}(\bar{y})$, and the second by $\exists x R_{\theta_1}(x, \bar{y}) \vee \bar{R}_{\theta}(\bar{y})$. If k is the arity of θ then (3) holds iff the first view is empty while the second is A^k .

Finally, \mathbf{V} contains a view V_{φ} that returns the value of R_{φ} , which coincides with that of φ . We now claim that $\mathbf{V} \rightarrow Q$. Consider $D \in \mathcal{I}(\tau'')$. As described above, the views \mathbf{V}_{σ} provide enough information to determine if the relations in σ satisfy (1)-(3). In particular, this determines the value of ψ . If ψ is false then $Q(D) = \emptyset$. If ψ is true, V_{φ} provides the value of φ . If φ is false then $Q(D) = \emptyset$. If φ is true then $Q(D)$ returns $D(T) = q(D(\tau))$ which is uniquely determined by $D(\tau)$ by definition of GIMP. Since \mathbf{V}_{τ} provides $D(\tau)$, it follows that $\mathbf{V} \rightarrow Q$.

Now consider $Q_{\mathbf{V}}$. By definition, $Q_{\mathbf{V}}$ computes $q(D(\tau))$ on instances $D \in \mathcal{I}(\sigma_{\mathbf{V}})$ extending $D(\tau)$ to $\sigma_{\mathbf{V}}$ with relations \emptyset or $\text{adom}(D(\tau))^k$ for the view relations in \mathbf{V}_{σ} corresponding to the case when φ is satisfied by the pre-image of the view, as described in (i)-(iii), and with *true* for V_{φ} , corresponding to the case when the pre-image satisfies φ . This extension is trivial, and easily expressible from $D(\tau)$, hence $Q_{\mathbf{V}}$ has the same complexity as q . \square

Note that Theorem 3.7 requires only UCQ views and therefore Theorem 3.5 is also tight when views are restricted to UCQs. As an immediate consequence of Theorem 3.7 any language not capturing $\mathbf{NP} \cap \text{co-NP}$ fails to be complete for UCQ-to-FO rewritings. As this is the case for Datalog $^{\neg}$ and fixpoint logic² FO+LFP [27] we have:

Corollary 3.9 *Datalog $^{\neg}$ and FO+LFP are not complete for UCQ-to-FO rewritings.*

We can show that FO is not a complete rewriting language even if the views are restricted to CQ $^{\neg}$, where CQ $^{\neg}$ is CQ extended with safe negation.

Proposition 3.10 *FO is not complete for CQ $^{\neg}$ -to-FO rewritings.*

Proof: We revisit Example 3.3. We use a strict linear order $<$ rather than \leq . As in the example, we use schemas σ and $\sigma_{<}$, an FO sentence ψ checking that $<$ is a strict total order, and the FO query $Q_{\varphi} = \psi \wedge \varphi(<)$ for some order-invariant FO($\sigma_{<}$) sentence φ which is not FO-definable over σ alone.

Let \mathbf{V} consist of the following views:

1. $x < y \wedge y < x$
2. $x < y \wedge y < z \wedge \neg(x < z)$,
3. for each pair of relations $R_1, R_2 \in \sigma$ and appropriate i, j , one view $R_1(\dots, x_i, \dots) \wedge R_2(\dots, x_j, \dots) \wedge \neg(x_i < x_j) \wedge \neg(x_j < x_i)$
4. for each $R \in \sigma$, one view returning R .

²FO+LFP is FO extended with a least fixpoint operator, see [19]. For the definitions of Datalog variants, see [2].

We claim that $\mathbf{V} \rightarrow Q_\varphi$. First, note that ψ holds (i.e. $<$ is a strict total order on the domain) iff views (1) and (2) are empty and the view (3) contains all the tuples (a, a) for a in $\text{adom}(R)$. Indeed, this ensures antisymmetry (1), transitivity (2), and totality (3). If any of these views is not empty, then ψ is false so Q_φ is false. Otherwise, Q_φ returns the value of $\varphi(<)$ on the relations in $\sigma_{<}$, which by the order invariance of $\varphi(<)$ depends only on the relations in σ . These are provided by the views (5). Thus, $\mathbf{V} \rightarrow Q_\varphi$ and $Q_{\mathbf{V}}$ allows defining $\varphi(<)$ using just the relations in σ , which cannot be done in FO. \square

Remark 3.11 *In work subsequent to [35, 31], Maarten Marx identifies a well-behaved fragment of FO with respect to determinacy and rewriting [30]. The fragment, called packed FO (PFO), is an extension of the well-known guarded fragment [21]. In a nutshell, guarded FO is restricted by allowing only quantifiers of the form $\exists \bar{x}(R(\bar{x}, \bar{y}) \wedge \varphi(\bar{x}, \bar{y}))$ where $R(\bar{x}, \bar{y})$ is an atom, and $\varphi(\bar{x}, \bar{y})$ is itself a guarded formula with free variables \bar{x}, \bar{y} . In terms of the relational algebra, guarded FO corresponds to the semi-join algebra [26]. Packed FO relaxes the guarded restriction by allowing as guards conjunctions of atoms so that each pair of variables occurs together in some conjunct. The results in [30] show that infinite and finite determinacy coincide for PFO, determinacy is decidable in 2-EXPTIME, and PFO is complete for PFO-to-PFO rewritings. Similar results are obtained for the packed fragment of UCQ, see Remark 5.23.*

4 UCQ Views and Queries

We have seen that determinacy is undecidable for FO views and queries, and indeed for any query language with undecidable satisfiability problem and any view language with undecidable validity problem. However, determinacy remains undecidable for much weaker languages. We show next that this holds even for UCQs. The undecidability result is quite strong, as it holds for a fixed database schema, a fixed view, and UCQs with no constants.

Theorem 4.1 *There exists a database schema σ and a fixed view \mathbf{V} over σ defined by UCQs, for which it is undecidable, given a UCQ query Q over σ , whether $\mathbf{V} \rightarrow Q$.*

Proof: The proof is by reduction from the word problem for finite monoids: Consider a finite set H of equations of the form $x \cdot y = z$, where x, y, z are symbols. Let F be an equation of the form $x = y$. Given a monoid (M, \circ) , the symbols are interpreted as elements in M and \cdot as the operation \circ of the monoid. Given H and F as above, the word problem for finite monoids asks whether H implies F over all finite monoids. This is known to be undecidable [25].

For the purpose of this proof, we will not work directly on monoids but rather on *monoidal* operations. Let X be a finite set. A function $f : X \times X \rightarrow X$ is said to be *monoidal* if it is complete (total and onto) and defines an associative operation. The operation of a monoid is always monoidal due to the presence of an identity element. It is immediate to extend Gurevich's undecidability result to monoidal functions by augmenting a monoidal function, if needed, with an identity element.

We construct a view \mathbf{V} and, given H and F as above, a query $Q_{H,F}$ such that $\mathbf{V} \rightarrow Q_{H,F}$ iff H implies F over all finite monoidal functions. We proceed in two steps. We construct \mathbf{V} and $Q_{H,F}$ in UCQ^- , then we show how equality can be removed.

Consider the database schema $\sigma = \{R, p_1, p_2\}$ where R is ternary and p_1, p_2 are zero-ary (propositions). We intend to represent $x \cdot y = z$ by $R(x, y, z)$.

A ternary relation R is *monoidal* if it is the graph of a monoidal function. That is, R is the graph of a (i) complete (ii) function which is (iii) associative. We construct a view \mathbf{V} which essentially checks that R is monoidal.

In order to do this we check that

- (i) $\{x \mid \exists y, z R(x, y, z)\} = \{y \mid \exists x, z R(x, y, z)\} = \{z \mid \exists x, y R(x, y, z)\}$,
- (ii) $\{(z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z')\} = \{(z, z') \mid z = z'\}$,
- (iii) $\{(w, w') \mid \exists x, y, z, u, v R(x, y, u) \wedge R(u, z, w) \wedge R(y, z, v) \wedge R(x, v, w')\} = \{(w, w') \mid w = w'\}$.

This is encoded in the view \mathbf{V} as follows.

Let $\phi_1(x, y, z)$ be $R(x, y, z)$, ϕ_2 be $p_1 \vee p_2$, ϕ_3 be $p_1 \wedge p_2$, and, for each equation α of the form $S = T$ in the list above let ϕ_α be $(p_1 \wedge S) \vee (p_2 \wedge T)$. Let \mathbf{V} consist of the queries ϕ_1, ϕ_2, ϕ_3 defining the views V_1, V_2, V_3 and ϕ_α defining the view V_α for each of the three equations α above.

We thus have: If D_1 and D_2 are such that $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ and exactly one of p_1, p_2 is true in D_1 and the other in D_2 , then R is monoidal. Indeed, for each equation α above, $V_\alpha(D_1) = V_\alpha(D_2)$ together with the assumption on p_1 and p_2 imply that R satisfies α . Note that (i) guarantees that R is total and onto, (ii) says that R is a function, and (iii) ensures associativity. Thus R is monoidal.

We now define $Q_{H,F}$. Given H and $F = \{x = y\}$, with x, y occurring in H , we set $\psi_{H,F}(x, y)$ to $\exists \bar{u} \bigwedge_{u_1 \cdot u_2 = u_3 \in H} R(u_1, u_2, u_3)$ (in the formula all the variables are quantified except for x and y). Let $Q_{H,F}(x, y)$ be $(p_1 \wedge p_2) \vee (p_1 \wedge \psi_{H,F}(x, y) \wedge x = y) \vee (p_2 \wedge \psi_{H,F}(x, y))$.

We claim that $\mathbf{V} \rightarrow Q_{H,F}$ iff H implies F over all finite monoidal functions.

Assume first that H implies F on all finite monoidal functions. Consider D_1 and D_2 such that $\mathbf{V}(D_1) = \mathbf{V}(D_2)$. If V_3 is true then $Q_{H,F}(D_1) = Q_{H,F}(D_2) = \text{adom}(R) \times \text{adom}(R)$. If V_3 and V_2 are false then $Q_{H,F}(D_1) = Q_{H,F}(D_2) = \emptyset$. In the remaining case exactly one of p_1, p_2 is true in D_1 and in D_2 . If it is the same for D_1 and D_2 then we immediately have $Q_{H,F}(D_1) = Q_{H,F}(D_2)$. Otherwise we have that D_1 and D_2 agree on R (by V_1) and R is monoidal (by the remark above). Since H implies F on monoidal functions, $\psi_{H,F}(x, y) \Rightarrow x = y$. This yields $Q_{H,F}(D_1) = Q_{H,F}(D_2)$.

Assume now that $\mathbf{V} \rightarrow Q_{H,F}$. Let R be a monoidal graph. Consider the extension D_1 of R with p_1 true and p_2 false, and the extension D_2 of R with p_1 false and p_2 true. We have $D_1 \models p_1 \wedge \neg p_2$, $D_2 \models \neg p_1 \wedge p_2$ and $\mathbf{V}(D_1) = \mathbf{V}(D_2)$. Therefore $Q_{H,F}(D_1) = Q_{H,F}(D_2)$ and $\psi_{H,F}(x, y) \Rightarrow x = y$. Thus R verifies H implies F .

In the above, equality is used explicitly in the formula for the view and query. Equality can be avoided as follows. A relation R is said to be *pseudo-monoidal* if the equivalence relation defined by $z \simeq z'$ iff $\exists x, y R(x, y, z) \wedge R(x, y, z')$ is a congruence with respect to R and R/\simeq is a monoidal function.

The property that \simeq is a congruence with respect to R can be enforced by the following equalities, ensuring that equivalent z and z' cannot be distinguished using R :

$$\begin{aligned} & \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(z, u, v)\} = \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(z', u, v)\}, \\ & \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(u, z, v)\} = \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(u, z', v)\}, \text{ and} \\ & \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(u, v, z)\} = \{(u, v, z, z') \mid \exists x, y R(x, y, z) \wedge R(x, y, z') \wedge R(u, v, z')\}. \end{aligned}$$

It is easily seen that for every R satisfying the above equalities, \simeq is a congruence.

We now modify \mathbf{V} by replacing the query that checks that R is the graph of a function (equation (ii)) by a set of queries corresponding to the new equations above and, replacing in the others all equalities $x = y$ by $\exists u, v R(u, v, x) \wedge R(u, v, y)$. We also modify the query $Q_{H,F}$, by replacing all equalities $x = y$ by $\exists u, v R(u, v, x) \wedge R(u, v, y)$.

As before, we can show that $\mathbf{V} \rightarrow Q_{H,F}$ iff H implies F over all finite pseudo-monoidal relations. The latter is undecidable, since implication over finite pseudo-monoidal relations is the same as implication over finite monoidal functions. This follows from the fact that every finite monoidal function is in particular a finite pseudo-monoidal relation, and every finite pseudo-monoidal relation can be turned into a finite monoidal function by taking its quotient with the equivalence relation \simeq defined above.

Finally, note that the zero-ary relations p_1 and p_2 can be avoided in the database schema if so desired by using instead two Boolean CQs over some non-zero-ary relation, whose truth values are independent of each other. For example, such sentences using a binary relation P might be $p_1 = \exists x, y, z P(x, y) \wedge P(y, z) \wedge P(z, x)$ and $p_2 = \exists x, y P(x, y) \wedge P(y, x)$. \square

We next turn to the question of UCQ-to-UCQ rewritings. Unfortunately, UCQ is not complete for UCQ-to-UCQ rewritings, nor are much more powerful languages such as Datalog[≠]. Indeed, the following shows that no monotonic language can be complete even for UCQ-to-CQ rewritings. In fact, this holds even for unary databases, views, and queries.

Proposition 4.2 *Any language complete for UCQ-to-CQ rewritings must express non-monotonic queries. Moreover, this holds even if the database relations, views, and query are restricted to be unary.*

Proof: Consider the schema $\sigma = \{R, P\}$ where R and P are unary. Let \mathbf{V} be the set consisting of three views V_1, V_2, V_3 defined by the following queries:

$$\begin{aligned}\phi_1(x) &: \exists u R(u) \wedge P(x) \\ \phi_2(x) &: R(x) \vee P(x) \\ \phi_3(x) &: R(x).\end{aligned}$$

Let $Q(x)$ be the query $P(x)$. It is easily seen that $\mathbf{V} \rightarrow Q$. Indeed, if $R \neq \emptyset$ then V_1 provides the answer to Q ; if $R = \emptyset$ then V_2 provides the answer to Q . Finally, V_3 provides R . Therefore, $\mathbf{V} \rightarrow Q$. Now consider $Q_{\mathbf{V}}$, associating $Q(D)$ to $\mathbf{V}(D)$. We show that $Q_{\mathbf{V}}$ is not monotonic. Indeed, let D_1 be the database instance where $P = \{a, b\}$ and R is empty. Let D_2 consist of $P = \{a\}$ and $R = \{b\}$. Then $\mathbf{V}(D_1) = \langle \emptyset, \{a, b\}, \emptyset \rangle$, $\mathbf{V}(D_2) = \langle \{a\}, \{a, b\}, \{b\} \rangle$, so $\mathbf{V}(D_1) \subseteq \mathbf{V}(D_2)$. However, $Q(D_1) = \{a, b\}$, $Q(D_2) = \{a\}$, and $Q(D_1) \not\subseteq Q(D_2)$. \square

We next show a similar result for CQ[≠] views and CQ queries.

Proposition 4.3 *Any language complete for CQ[≠]-to-CQ rewritings must express non-monotonic queries. Moreover, this holds even if the views and query are monadic (i.e. they answers are unary relations).*

Proof: Let $\sigma = \{R\}$, where R is binary. Consider the view \mathbf{V} consisting in three views V_1, V_2, V_3 defined as follows: $\phi_1(x) = \exists y R(x, y) \wedge R(y, x)$, $\phi_2(x) = \exists y R(x, y) \wedge R(y, x) \wedge x \neq y$ and

$\phi_3(x) = \exists y R(x, x) \wedge R(x, y) \wedge R(y, x) \wedge x \neq y$. Consider the query $Q(x)$ defined by $R(x, x)$. It is easy to check that $\mathbf{V} \rightarrow Q$. Indeed Q can be defined by $(V_1 \wedge \neg V_2) \vee V_3$. Consider the databases instances D and D' where R is respectively $\{(a, a)\}$ and $\{(a, b), (b, a)\}$. Then $\mathbf{V}(D) = \langle \{a\}, \emptyset, \emptyset \rangle$, $\mathbf{V}(D') = \langle \{a, b\}, \{a, b\}, \emptyset \rangle$, $Q(D) = \{a\}$ and $Q(D') = \emptyset$. Thus, $\mathbf{V}(D) \subset \mathbf{V}(D')$ but $Q(D) \not\subseteq Q(D')$. Therefore Q cannot be rewritten in terms of \mathbf{V} using a monotonic language. \square

From Proposition 4.2 and Proposition 4.3 we immediately have:

Corollary 4.4 *Datalog $^\neq$ is not complete for CQ $^\neq$ -to-CQ rewritings or for UCQ-to-CQ rewritings, even if the views and query are restricted to be monadic.*

Proposition 4.2 provides a lower bound of sorts for any rewriting language complete for UCQ-to-UCQ rewritings or for CQ $^\neq$ -to-CQ rewritings. Note that it also follows from Theorem 4.1 that UCQ is not complete for UCQ-to-UCQ rewritings. Indeed, by Theorem 3.9 in [29], it is decidable if a UCQ query Q can be rewritten in terms of a set of UCQ views \mathbf{V} using a UCQ. Therefore, completeness of UQC for UCQ-to-UCQ rewritings would contradict the undecidability of determinacy.

Recall that, from Theorem 3.5, we know that \exists SO and \forall SO are both complete for UCQ-to-UCQ rewritings and for CQ $^\neq$ -to-CQ rewritings. It remains open if we can do better. In particular, we do not know if FO is complete for UCQ-to-UCQ rewritings or for CQ $^\neq$ -to-CQ rewritings.

5 Conjunctive views and queries

Conjunctive queries are the most widely used in practice, and the most studied in relation to answering queries using views. Given CQ views \mathbf{V} and a CQ query Q , it is decidable whether Q can be rewritten in terms of \mathbf{V} using another CQ ([29]). But is CQ complete for CQ-to-CQ rewriting? Note that this would immediately imply decidability of determinacy for CQ views and queries, by the above result of [29]. Surprisingly, we show that CQ is *not* complete for CQ-to-CQ rewriting. Moreover, the decidability of determinacy for conjunctive views and queries remains open, and appears to be a hard question. The problem is only settled for some special fragments of CQs, described later in this section.

5.1 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.

Before proving 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} . In this section, we will use the standard technique of identifying each conjunctive query with an *instance* called its *frozen body*. To this end, let \mathbf{var} be an infinite set of *variables* disjoint from \mathbf{dom} , such that all variable used in conjunctive queries belong to \mathbf{var} . Instead of instances over \mathbf{dom} , we will consider in this section instances over the extended domain $\mathbf{dom} \cup \mathbf{var}$. Now we can associate an instance to a conjunctive query Q as follows. The *frozen body of Q* , denoted $[Q]$, is the instance over σ such that $(x_1, \dots, x_k) \in R$ iff $R(x_1, \dots, x_k)$ is an atom in Q . Note that (x_1, \dots, x_k) may contain constants (from \mathbf{dom}) as well as variables (from \mathbf{var}). 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 $[Q]$.

A *homomorphism* from an instance I to an instance J over the extended domain $\mathbf{dom} \cup \mathbf{var}$ is a classical homomorphism that is the identity on \mathbf{dom} . Recall that a tuple \bar{c} is in $Q(D)$ for a CQ Q and database instance D iff there exists a homomorphism h from $[Q]$ to D such that $h(\bar{x}) = \bar{c}$. In this case we say that h *witnesses* $\bar{c} \in Q(D)$, or that $\bar{c} \in Q(D)$ via h .

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 $\phi_V(\bar{x})$. For every tuple \bar{c} belonging to V in S , we include in D the tuples of $\alpha([\phi_V])$ where $\alpha(\bar{x}) = \bar{c}$ and α maps every variable of $[\phi_V]$ not in \bar{x} to some new distinct value not in $\mathit{adom}(S) \cup C$. For the reader familiar to the chase, $\mathbf{V}_C^{-1}(S)$ is obtained as a *chase* of S with respect to the tuple generating dependencies $\forall \bar{x}(V(\bar{x}) \rightarrow \exists \bar{y} \alpha_V(\bar{x}, \bar{y}))$, where $V \in \sigma_{\mathbf{V}}$ and α_V is the body of the query defining V in \mathbf{V} , in which all values introduced as witnesses are outside $\mathit{adom}(S)$ and C . To simplify, we usually assume that C consists of the entire active domain of D 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 were also observed in [17]:

Proposition 5.1 *Let $Q(\bar{x})$ be a CQ (with free variables \bar{x}) and $S = \mathbf{V}([Q])$. Let $\varphi_{\mathbf{V}}(\bar{x})$ be the CQ over $\sigma_{\mathbf{V}}$ for which $[\varphi_{\mathbf{V}}] = S$. We have the following:*

- (i) $S \subseteq \mathbf{V} \circ \mathbf{V}^{-1}(S)$;
- (ii) $Q \subseteq \varphi_{\mathbf{V}} \circ \mathbf{V}$;
- (iii) *If $\bar{x} \in Q(\mathbf{V}^{-1}(S))$ then $Q \equiv \varphi_{\mathbf{V}} \circ \mathbf{V}$. In particular, $\mathbf{V} \rightarrow Q$.*
- (iv) *If Q has a CQ rewriting in terms of \mathbf{V} , then $\varphi_{\mathbf{V}}$ is such a rewriting.*

Proof: Part(i) is obvious from the definition of $\mathbf{V}^{-1}(S)$.

For part (ii) notice that by definition of $\varphi_{\mathbf{V}}$, the frozen body of $\varphi_{\mathbf{V}} \circ \mathbf{V}$ is isomorphic to $\mathbf{V}^{-1}(S)$. The inclusion then follows from the fact that there is a homomorphism from $\mathbf{V}^{-1}(S)$ to $[Q]$ preserving \bar{x} . To see this, recall that every tuple $\bar{s} \in V([Q])$ is witnessed by a homomorphism $h_{\bar{s}}$ from $[\phi_V]$ to $[Q]$ and also induces an copy of $[\phi_V]$ in $\mathbf{V}^{-1}(S)$. The union of all the $h_{\bar{s}}$ yields the desired homomorphism from $\mathbf{V}^{-1}(S)$ to $[Q]$.

Consider now (iii). If $\bar{x} \in Q(\mathbf{V}^{-1}(S))$ then there is a homomorphism from $[Q]$ to $\mathbf{V}^{-1}(S)$ fixing \bar{x} . As the body of $\varphi_{\mathbf{V}} \circ \mathbf{V}$ is $\mathbf{V}^{-1}(S)$, it follows that $\varphi_{\mathbf{V}} \circ \mathbf{V} \subseteq Q$ and we conclude using (ii).

It remains to show (iv). Suppose that $q_{\mathbf{V}}$ is a CQ rewriting of Q using \mathbf{V} . In particular we have for all D , $Q(D) = q_{\mathbf{V}} \circ \mathbf{V}(D)$. We show that $\bar{x} \in Q(\mathbf{V}^{-1}(S))$ and we conclude using (iii). We have $\bar{x} \in Q([Q])$. Hence $\bar{x} \in q_{\mathbf{V}}(\mathbf{V}([Q])) = q_{\mathbf{V}}(S)$. By (i), $S \subseteq \mathbf{V} \circ \mathbf{V}^{-1}(S)$. Because $q_{\mathbf{V}}$ is conjunctive, so monotonic, we also have $\bar{x} \in q_{\mathbf{V}}(\mathbf{V} \circ \mathbf{V}^{-1}(S))$. But $q_{\mathbf{V}}(\mathbf{V} \circ \mathbf{V}^{-1}(S))$ is $q_{\mathbf{V}} \circ \mathbf{V}(\mathbf{V}^{-1}(S))$. Hence $\bar{x} \in Q(\mathbf{V}^{-1}(S))$. \square

Note that Proposition 5.1 applies both to the finite and unrestricted cases.

We can now show the following.

Theorem 5.2 *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 $\mathbf{V} \rightarrow Q$ but $\bar{x} \notin Q(\mathbf{V}^{-1}(S))$, where $S = \mathbf{V}([Q])$. By Proposition 5.1, 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):

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

$$\text{Let } Q(x, y) = \exists a \exists b \exists c [R(a, x) \wedge R(a, b) \wedge R(b, c) \wedge R(c, y)] \quad x \leftarrow a \rightarrow b \rightarrow c \rightarrow y.$$

We first show that $\mathbf{V} \rightarrow Q$. To do so, we prove that the formula

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

is a rewriting of Q using \mathbf{V} . In other words, for each database D over σ and $u, v \in \text{adom}(D)$, $\langle u, v \rangle \in Q(D)$ 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 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 $\langle d, v \rangle$ 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 $\mathbf{V} \rightarrow Q$.

In view of Proposition 5.1, 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	$x \quad c$	V_2	$a \quad c$	V_3	$a \quad y$
	$b \quad c$		$b \quad y$		
	$c \quad y$				

and $\mathbf{V}^{-1}(S)$ is depicted in Figure 2.

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} . \square

Remark 5.3 *A different and somewhat simpler proof of Theorem 5.2 has recently been exhibited in [3]. It extends the technique in the above proof by providing an infinite set of examples of CQ views and queries for which the view determines the query but the query has no CQ rewriting in terms of the view. The examples involve path queries $P_n(x, y)$ on a binary relation R stating that there is a path of length n from x to y in R ($n > 1$). For instance, it is shown that $\{P_3, P_4\} \rightarrow P_5$ but P_5 has no CQ rewriting in terms of P_3 and P_4 . However, P_5 has the FO rewriting*

$$P_5(x, y) \equiv \exists z [P_4(x, z) \wedge \forall v (P_3(v, z) \rightarrow P_4(v, y))].$$

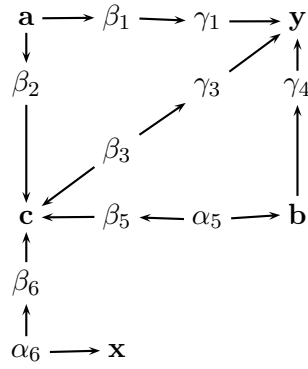


Figure 2: The graph of $\mathbf{V}^{-1}(S)$ (elements in $\text{adom}(S)$ are in boldface).

Remark 5.4 *Note that Theorem 5.2 holds in the finite as well as the unrestricted case. This shows that Theorem 3.3 in [35], claiming that CQ is complete for CQ-to-CQ rewriting in the unrestricted case, is erroneous. Also, Theorem 3.7 in [35], claiming decidability of determinacy in the unrestricted case as a corollary, remains unproven. The source of the problem is Proposition 3.6 in [35], which is unfortunately false (the views and queries used in the above proof are a counterexample). Specifically, part (5) of that proposition would imply that $\langle x, y \rangle \in Q(\mathbf{V}^{-1}(S))$ for \mathbf{V} and Q in the proof of Theorem 5.2, which is false.*

As a consequence of the proof of Theorem 5.2 we have:

Corollary 5.5 *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 5.2. It was shown that $\mathbf{V} \rightarrow Q$. Consider the database instances $D_1 = [Q]$ and $D_2 = R$ where R is the relation depicted in Figure 2. 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

5.2 Completeness for monotonic mappings

To show that CQ is not complete for CQ-to-CQ rewritings, we exhibited in the proof of Theorem 5.2 a set \mathbf{V} of CQ views and a CQ query Q such that $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ is non-monotonic. One might wonder if there are always CQ rewritings in the cases when $Q_{\mathbf{V}}$ is monotonic. An affirmative answer is provided by the following result.

Theorem 5.6 *Let \mathbf{V} be a set of CQ views and Q a CQ query such that $\mathbf{V} \rightarrow Q$. If $Q_{\mathbf{V}}$ is monotonic, then there exists a CQ rewriting of Q using \mathbf{V} .*

Proof: Let \mathbf{V} be a set of CQ views and $Q(\bar{x})$ a CQ query (with free variables \bar{x}) such that $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ is monotonic. We use the notation of Proposition 5.1. Let $D_1 = [Q]$ be the database consisting of the body of Q , and $D_2 = \mathbf{V}^{-1}(\mathbf{V}([Q]))$. By construction, $\mathbf{V}(D_1) \subseteq \mathbf{V}(D_2)$. Since $Q_{\mathbf{V}}$ is monotonic, $Q(D_1) \subseteq Q(D_2)$. Note that $\bar{x} \in Q(D_1)$. It follows that $\bar{x} \in Q(D_2)$, so by (iii) of Proposition 5.1, $\varphi_{\mathbf{V}}$ is a CQ rewriting of Q using \mathbf{V} . \square

Another way to look at Theorem 5.6 is as a kind of “gap theorem”. Indeed, the theorem says that adding “small” features such as inequality, constants, or union to CQs does not increase its power as a language for CQ-to-CQ rewritings. Instead, in order to obtain any increase in rewriting power, one has to augment CQs all the way to a non-monotonic language.

A similar result can be obtained for some languages beyond CQs. Consider UCQs. Recall that no monotonic language can be complete for UCQ-to-UCQ rewriting, because, as in the case of CQs, there are UCQ views \mathbf{V} and query Q such that $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ is non-monotonic (see Proposition 4.2). A similar situation holds for CQ^{\neq} views and CQ queries (see Proposition 4.3). It turns out that the completeness result of Theorem 5.6 can be extended to these cases, as shown next.

Theorem 5.7 (i) Let $\mathbf{V} \in CQ^{\neq}$ and $Q \in (U)CQ$. If $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ is monotonic, then there exists $R \in (U)CQ$ such that $Q \Rightarrow_{\mathbf{V}} R$. (ii) Let $\mathbf{V} \in UCQ$ and $Q \in UCQ$. If $\mathbf{V} \rightarrow Q$ and $Q_{\mathbf{V}}$ is monotonic, then there exists $R \in UCQ$ such that $Q \Rightarrow_{\mathbf{V}} R$.

Proof: For proving part (i) we will make use of results of Abiteboul and Duschka [1] on rewriting of certain answers to queries, in terms of sound views. For the sake of self containment, we briefly recall these results.

Let \mathbf{V} be a set of views and Q a query over the same database schema. In the context of data integration, it is often of interest to consider *sound* views of a database D with respect to \mathbf{V} . An instance E of $\sigma_{\mathbf{V}}$ is a sound view of D with respect to \mathbf{V} iff $E \subseteq \mathbf{V}(D)$. Given an instance E over $\sigma_{\mathbf{V}}$ that is a sound view of some database with respect to \mathbf{V} , and a query Q , one is typically interested to compute the set of *certain* answers to Q with respect to \mathbf{V} , defined as $cert_Q(E) = \cap \{Q(D) \mid E \subseteq \mathbf{V}(D)\}$.

Now suppose $\mathbf{V} \in CQ^{\neq}$ and $Q \in \text{Datalog}$. We call an instance E over $\sigma_{\mathbf{V}}$ a *sound view* with respect to \mathbf{V} iff $E \subseteq \mathbf{V}(D)$ for *some* database instance D . Note that, due to the presence of inequalities in \mathbf{V} , not every instance over $\sigma_{\mathbf{V}}$ is a sound view.

Theorem 4.3 in [1] states that the inequalities in \mathbf{V} do not affect the certain answers to Q , so can be eliminated. More precisely:

Lemma 5.8 (Theorem 4.3 in [1]) Let $\mathbf{V} \in CQ^{\neq}$ and $Q \in \text{Datalog}$. Let \mathbf{V}^- be obtained from \mathbf{V} by removing all inequalities. Then for every sound view E with respect to \mathbf{V} , $cert_Q(E)$ with respect to \mathbf{V} equals $cert_Q(E)$ with respect to \mathbf{V}^- .

We next consider the rewriting question for certain answers and sound views. In this context, the question is to find a query P over $\sigma_{\mathbf{V}}$ such that $P(E) = cert_Q(E)$ for every sound view E with respect to \mathbf{V} . We use again a result from [1], that provides a rewriting of $cert_Q$ in terms of \mathbf{V} when $\mathbf{V} \in CQ$ and $Q \in \text{Datalog}$.

Lemma 5.9 (consequence of Theorem 4.2 in [1]) Let $\mathbf{V} \in CQ$ and $Q \in \text{Datalog}$. There exists a query $P \in \text{Datalog}$, computable from \mathbf{V} and Q , such that for every instance E over $\sigma_{\mathbf{V}}$, $cert_Q(E) = P(E)$ under the sound view semantics.

Now consider \mathbf{V} and Q as in (i). Recall that $cert_Q(E) = \cap_D \{Q(D) \mid E \subseteq \mathbf{V}(D)\}$. Since $\mathbf{V} \rightarrow Q$, $cert_Q(E) = \cap_S \{Q_{\mathbf{V}}(S) \mid E \subseteq S \wedge S = \mathbf{V}(D)\}$. Since $Q_{\mathbf{V}}$ is monotonic, $Q_{\mathbf{V}}(E) = cert_Q(E)$ for each E in the image of \mathbf{V} . Hence, for every database D , $Q(D) = cert_Q(\mathbf{V}(D))$. Thus, it is enough to find a rewriting for $cert_Q$. By Lemma 5.8, we can assume that \mathbf{V} contains no inequalities. By Lemma 5.9, there exists a Datalog program P over $\sigma_{\mathbf{V}}$ that computes $cert_Q$, so is also a rewriting of Q with

respect to \mathbf{V} . Since P expanded with the view definition is an infinite union $\cup_{i \geq 0} \varphi_i$ of conjunctive queries over the input database and Q is a finite union of conjunctive queries over the same, it easily follows that Q is equivalent to a finite union of conjunctive queries among $\{\varphi_i \mid i \geq 0\}$, so can be rewritten as a finite union of conjunctive queries over \mathbf{V} . Furthermore, if Q is in CQ then Q is equivalent to a single conjunctive query among $\{\varphi_i \mid i \geq 0\}$. Note that this provides an alternative proof to Theorem 5.6.

Now consider (ii). The proof is an extension of that of Theorem 5.6. For readability, we sketch the proof for the case when \mathbf{V} consists of a single UCQ view V . Let $V(\bar{v}) = \bigvee_{i=1}^n V_i(\bar{v})$ where each

$V_i(\bar{v})$ is a CQ and \bar{v} are the free variables. Similarly, let $Q(\bar{q}) = \bigvee_{j=1}^m Q_j(\bar{q})$ where each $Q_j(\bar{q})$ is a CQ

and \bar{q} are the free variables. We use the notation developed in the proof of Proposition 5.1. Thus, for each Q_j we denote by $[Q_j]$ the frozen body of Q_j and let $S_j = V([Q_j])$. Let $\varphi_V(\bar{q})$ be the UCQ $\exists \bar{y} \bigvee_{j=1}^m \bigwedge_{\bar{t} \in S_j} V(\bar{t})$ where \bar{y} contains all the bound variables of Q .

We claim that φ_V defines Q_V . In other words, Q is equivalent to $\varphi_V \circ V$.

Consider $Q \subseteq \varphi_V \circ V$. Let $\psi_j = \exists \bar{y} (\bigwedge_{\bar{t} \in S_j} V(\bar{t})) \circ V$. Clearly, it is enough to show that $Q_j \subseteq \psi_j$ for every $j, 1 \leq j \leq m$. Let us fix such j . Let Σ_j be the set of mappings σ from S_j to $\{1, \dots, n\}$ such that $\bar{t} \in V_{\sigma(\bar{t})}([Q_j])$ for every $\bar{t} \in S_j$. For each $\sigma \in \Sigma_j$, let $\psi_j^\sigma(\bar{q})$ be the CQ whose free variables are \bar{q} and whose frozen body is $[\psi_j^\sigma] = \bigcup_{\bar{t} \in S_j} V_{\sigma(\bar{t})}^{-1}(\bar{t})$. Using distributivity of conjunction over disjunction,

$\psi_j = \bigvee_{\sigma \in \Sigma_j} \psi_j^\sigma$. Thus, $\psi_j^\sigma \subseteq \psi_j$ for every $\sigma \in \Sigma_j$. On the other hand, by construction there exists a homomorphism from each $[\psi_j^\sigma]$ to $[Q_j]$ fixing \bar{q} . It follows that $Q_j \subseteq \psi_j^\sigma$ for every $\sigma \in \Sigma_j$. This together with $\psi_j^\sigma \subseteq \psi_j$ implies that $Q_j \subseteq \psi_j$.

Now consider $\varphi_V \circ V \subseteq Q$. Clearly, it is enough to show that $\psi_j \subseteq Q$ for every $j, 1 \leq j \leq m$. Let Σ_j and ψ_j^σ be defined as above ($\sigma \in \Sigma_j$). As noted above, ψ_j is equivalent to $\bigvee_{\sigma \in \Sigma_j} \psi_j^\sigma$. Thus, it is enough to show that $\psi_j^\sigma \subseteq Q$ for each σ . Consider the database instances $D_1 = [Q_j]$ and $D_2 = [\psi_j^\sigma]$. By construction, $V(D_1) = V([Q_j]) = S_j \subseteq V(D_2)$. Since Q_V is monotonic, it follows that $Q(D_1) \subseteq Q(D_2)$. However, $\bar{q} \in Q(D_1)$, so $\bar{q} \in Q(D_2)$ and there is $i, 1 \leq i \leq m$, such that $\bar{q} \in Q_i(D_2)$. Thus, there is a homomorphism from $[Q_i]$ to $[\psi_j^\sigma]$ fixing \bar{q} , and so $\psi_j^\sigma \subseteq Q_i \subseteq Q$. Since $\psi_j = \bigvee_{\sigma \in \Sigma_j} \psi_j^\sigma$, it follows that $\psi_j \subseteq Q$. \square

5.3 Effective FO rewriting in the unrestricted case

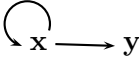
As shown above, CQ is not complete for CQ-to-CQ rewriting. What is the rewriting power needed of a complete language? In the finite case, it is open whether FO is sufficient, and the best known upper bounds remain \exists SO and \forall SO. In the unrestricted case, we know from Theorem 3.1 that FO is complete for CQ-to-CQ rewriting. However, this does not necessarily guarantee *effective* rewritings. We next show that such effective FO rewritings do indeed exist for CQ-to-CQ rewritings. Recall that it is open whether finite and unrestricted determinacy coincide for CQ views and queries.

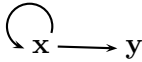
First, we review a characterization of determinacy in the unrestricted case, based on the chase. Let \mathbf{V} be a set of CQ views and $Q(\bar{x})$ a CQ over the same database schema σ . We construct two possibly infinite instances D_∞ and D'_∞ such that $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$ 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}).

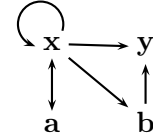
Referring to the proof of Theorem 5.2, note that the instance depicted in Figure 2 coincides with D'_0 .

Example 5.10 We illustrate with a simple example the first steps in the chase construction. Consider a database schema consisting of a single binary relation R . Consider the binary view $V(u, v) = \exists z R(u, v) \wedge R(u, z) \wedge R(z, v)$. Consider the Boolean query Q defined by $\exists x, y R(x, x) \wedge R(x, y)$. We construct the first steps of the chase, specifically $D_0, V_0, D'_0, V'_1, D_1, V_1$. Since all instances are binary, we represent them as directed graphs.

By definition, $D_0 = [Q]$ is the graph 

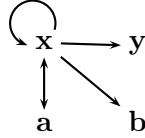
Clearly, $V_0 = \mathbf{V}(D_0)$ is the same graph 

We can now compute $D'_0 = \mathbf{V}^{-1}(V_0)$ and obtain the graph

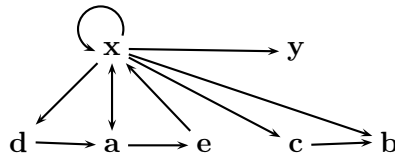


where **a** and **b** are

distinct new variables. Next, $V'_1 = \mathbf{V}(D'_0)$ is the graph

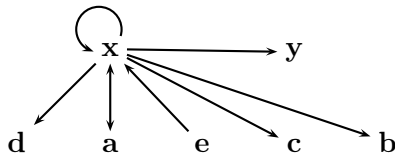


Recall that $S'_1 = V'_1 - V_0$ and $D_1 = D_0 \cup \mathbf{V}^{-1}(S'_1)$. Thus, D_1 is the graph



where **c**, **d**, and **e** are distinct new variables.

Finally, $V_1 = \mathbf{V}(D_1)$ is the graph



It can be easily seen that the chase never terminates for this example, so D_∞ and D'_∞ are infinite instances. Although immaterial to the chase construction, note that $\{V\} \rightarrow Q$. Indeed, Q is equivalent to $\exists x \exists y (V(x, x) \wedge V(x, y))$. \square

We will use the following properties.

Proposition 5.11 *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 $\text{dom}([Q])$.*
3. *For every pair of database instances I, I' such that $\mathbf{V}(I) = \mathbf{V}(I')$ and $\bar{a} \in Q(I)$ via homomorphism μ (from $[Q]$ to I), there exists a homomorphism from D'_∞ into I' mapping each free variable x of Q occurring in D'_∞ to the corresponding $\mu(x)$ in \bar{a} .*

Proof: Consider (1). Recall that $D_\infty = \bigcup_k D_k$ and $D'_\infty = \bigcup_k D'_k$. Since \mathbf{V} are CQs and the sequences of sets $\{D_k\}_{k \geq 0}$ and $\{D'_k\}_{k \geq 0}$ are non-decreasing, $\mathbf{V}(D_\infty) = \bigcup_{k \geq 0} \mathbf{V}(D_k)$ and $\mathbf{V}(D'_\infty) = \bigcup_{k \geq 0} \mathbf{V}(D'_k)$. We show that $\mathbf{V}(D_k) \subseteq \mathbf{V}(D'_k)$. Recall that by definition $\mathbf{V}(D_k)$ is $V_k = V'_k \cup S_k$ and that D'_k is $D'_{k-1} \cup \mathbf{V}^{-1}(S_k)$. Because \mathbf{V} is CQ, we have $V'_k \cup \mathbf{V} \circ \mathbf{V}^{-1}(S_k) \subseteq \mathbf{V}(D'_k)$. By (i) of Proposition 5.1, $S_k \subseteq \mathbf{V} \circ \mathbf{V}^{-1}(S_k)$ and hence $\mathbf{V}(D_k) \subseteq \mathbf{V}(D'_k)$. Similarly we can show that $\mathbf{V}(D'_k) \subseteq \mathbf{V}(D_{k+1})$. It follows that $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$.

Next, consider (2). First, note that, by construction, $\text{dom}(D_k) \cap \text{dom}(D'_k) = \text{dom}(V_k)$ (recall that $V_k = \mathbf{V}(D_k)$). We show by induction the following.

- (\dagger) for each $k \geq 0$ there exist homomorphisms h_k from D_k to $[Q]$ and h'_k from D'_k to $[Q]$ that agree on their common domain $\text{dom}(V_k)$ and are the identity on $\text{dom}(V_k) \cap \text{dom}([Q])$. Furthermore, h_{k+1} extends h_k and h'_{k+1} extends h'_k .

For $k = 0$, $D_0 = [Q]$ and $D'_0 = \mathbf{V}^{-1}(\mathbf{V}([Q]))$ so h_0 is the identity and h'_0 extends h_0 to the new witnesses in $\mathbf{V}^{-1}(\mathbf{V}([Q]))$. In particular, h_0 and h'_0 agree on $\text{dom}(D_0) \cap \text{dom}(D'_0) = \text{dom}(V_0)$. Now let $k \geq 0$ and suppose h_i and h'_i have the desired properties for $0 \leq i \leq k$. In particular, h_k extends h_0 , and since h_0 is the identity on $\text{dom}([Q])$, h_k is also the identity on $[Q]$. We define h_{k+1} and h'_{k+1} as follows. Recall that $S'_{k+1} = \mathbf{V}(D'_k) - \mathbf{V}(D_k)$ and $D_{k+1} = D_k \cup \mathbf{V}^{-1}(S'_{k+1})$. By construction, there exists a homomorphism f from $\mathbf{V}^{-1}(S'_{k+1})$ to D'_k that is the identity on $\text{dom}(D_k) \cup \text{dom}(D'_k)$ (since new witnesses are used in every application of \mathbf{V}^{-1}). It follows that $f \circ h'_k$ is a homomorphism from $\mathbf{V}^{-1}(S'_{k+1})$ to $[Q]$ that agrees with h_k on $\text{dom}(D_k) \cap \text{dom}(D'_k)$ so on $\text{dom}(D_k) \cap \text{dom}(\mathbf{V}^{-1}(S'_{k+1}))$. Let h_{k+1} be the union of h_k and $f \circ h'_k$ (so in particular h_{k+1} extends h_k). Thus, h_{k+1} is a homomorphism from D_{k+1} to $[Q]$ that is the identity on $\text{dom}([Q])$. Next, recall that $D'_{k+1} = D'_k \cup \mathbf{V}^{-1}(S_{k+1})$, where $S_{k+1} = \mathbf{V}(D_{k+1}) - \mathbf{V}(D'_k)$. By definition, there is a homomorphism g from $\mathbf{V}^{-1}(S_{k+1})$ to D_{k+1} that is the identity on $\text{dom}(D_{k+1})$. It follows that

$g \circ h_{k+1}$ is a homomorphism from $\mathbf{V}^{-1}(S_{k+1})$ to $[Q]$ that is the identity on $[Q]$ and agrees with h'_k on their common domain. Let h'_{k+1} be the union of h'_k and $g \circ h_{k+1}$. In view of the above, h'_{k+1} is a homomorphism from D'_{k+1} to $[Q]$ that agrees with h_{k+1} on $\text{dom}(V_{k+1})$ and is the identity on $\text{dom}(V_{k+1}) \cap \text{dom}([Q])$. This completes the induction. By construction, h_{k+1} extends h_k and h'_{k+1} extends h'_k for every $k \geq 0$. This proves (\dagger) .

To prove (2), let $h = \cup_{k \geq 0} h_k$ and $h' = \cup_{k \geq 0} h'_k$. Since h_{k+1} extends h_k and h'_{k+1} extends h'_k for each $k \geq 0$, h and h' are well-defined homomorphisms from D_∞ , resp. D'_∞ , to $[Q]$, that are the identity on $\text{dom}([Q])$.

Now consider (3). Let I, I' be database instances as in the statement. We prove by induction the following.

(\ddagger) for each $k \geq 0$ there exist homomorphisms f_k from D_k to I and f'_k from D'_k to I' such that:

- (i) $f_0 = \mu$,
- (ii) f_{k+1} extends f_k and f'_{k+1} extends f'_k , and
- (iii) f_k and f'_k agree on $\text{dom}(V_k)$.

Clearly, (\ddagger) suffices to prove (3). Indeed, let $f = \cup_{k \geq 0} f_k$ and $f' = \cup_{k \geq 0} f'_k$. Due to (ii), f and f' are well defined homomorphisms from D_∞ to I , resp. D'_∞ to I' . From (i) – (iii) it then follows that f' agrees with μ on every free variable x of Q occurring in D'_∞ (note that not all such free variables must occur in D'_∞ , since we do not assume that $\mathbf{V} \xrightarrow{\infty} Q$).

We now prove (\ddagger) . For the basis, let $f_0 = \mu$ (thus, (i) is satisfied). Clearly, $\mu(V_0) \subseteq \mathbf{V}(I)$, so $\mu(V_0) \subseteq \mathbf{V}(I')$. Since $D'_0 = \mathbf{V}^{-1}(V_0)$, it follows that there exists a homomorphism f'_0 from D'_0 to I' that agrees with μ on $\text{dom}(V_0)$. But $\mu = f_0$, so f'_0 agrees with f_0 on $\text{dom}(V_0)$.

For the induction, suppose $k \geq 0$ and f_k and f'_k are constructed so that (i), (iii) hold. Since $S'_{k+1} \subseteq \mathbf{V}(D'_k)$ it follows that $f'_k(S'_{k+1}) \subseteq \mathbf{V}(f'_k(D'_k)) \subseteq \mathbf{V}(I') = \mathbf{V}(I)$. Since $D_{k+1} = D_k \cup \mathbf{V}^{-1}(S'_{k+1})$, there exists an extension f_{k+1} of f_k such that $f_{k+1}(D_{k+1}) \subseteq I$ and f_{k+1} agrees with f'_k on $\text{dom}(S'_{k+1})$. Since by the induction hypothesis f_k and f'_k agree on $\text{dom}(V_k)$, and f_{k+1} extends f_k , it follows that f_{k+1} and f'_k agree on $\text{dom}(V_{k+1})$. Since $S_{k+1} \subseteq \mathbf{V}(D_{k+1})$, it further follows that

$$f_{k+1}(S_{k+1}) \subseteq \mathbf{V}(f_{k+1}(D_{k+1})) \subseteq \mathbf{V}(I) = \mathbf{V}(I').$$

Since $D'_{k+1} = D'_k \cup \mathbf{V}^{-1}(S_{k+1})$, there exists an extension f'_{k+1} of f'_k mapping D'_{k+1} to I' and agreeing with f_{k+1} on $\text{dom}(S_{k+1})$. This in conjunction with the fact that f_{k+1} agrees with f'_k on $\text{dom}(V_{k+1})$ implies that f_{k+1} and f'_{k+1} agree on V_{k+1} . This completes the induction. \square

As a consequence of the above we have the following.

Proposition 5.12 *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 $\mathbf{V} \xrightarrow{\infty} Q$ iff $\bar{x} \in Q(D'_\infty)$.*

Proof: Suppose $\mathbf{V} \xrightarrow{\infty} Q$. By (1) of Proposition 5.11, $\mathbf{V}(D_\infty) = \mathbf{V}(D'_\infty)$. It follows that $Q(D_\infty) = Q(D'_\infty)$. But $[Q] = D_0 \subseteq D_\infty$, so $\bar{x} \in Q(D_\infty)$. It follows that $\bar{x} \in Q(D'_\infty)$. Conversely, suppose that $\bar{x} \in Q(D'_\infty)$ is witnessed by h . 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 $Q(I) \subseteq Q(I')$. Let $\bar{a} \in Q(I)$. By (3) of Proposition 5.11, there exists a homomorphism h' from D'_∞ into I' mapping \bar{x} to \bar{a} . The composition of h and h' yields a homomorphism from $[Q]$ to I' mapping \bar{x} to \bar{a} , and hence $\bar{a} \in Q(I')$. \square

We are now ready to show that FO is complete for CQ-to-CQ rewriting in the unrestricted case, and that the FO rewriting can be effectively computed. 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 $t \in S(R)$. Note that ϕ_S is simply *true* if $S = \emptyset$. Let $k \geq 0$ be fixed. We define a sequence of formulas $\{\varphi_i^k\}_{i=0}^k$ by induction on i , backward from k . Let $\bar{\alpha}_i$ be the elements in $\text{adom}(S_i) - \text{adom}(V'_i)$ if $i > 0$ and in $\text{adom}(S_0) - \{\bar{x}\}$ if $i = 0$. For the basis, let

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

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 as above and $\bar{\alpha}'_{i+1}$ are the elements in $\text{adom}(S'_{i+1}) - \text{adom}(V_i)$.

We now show the following.

Theorem 5.13 *Let \mathbf{V} be a set of CQ views and Q be a CQ, both over database schema σ . If $\mathbf{V} \xrightarrow{\infty} Q$, then there exists $k \geq 0$ such that $Q \cong_{\mathbf{V}} \varphi_0^k$.*

Proof: Suppose $\mathbf{V} \xrightarrow{\infty} Q$. Let \bar{x} be the free variables of Q . By Proposition 5.12, there exists a homomorphism from $[Q]$ into D'_∞ fixing the free variables \bar{x} of Q . Since $D'_\infty = \bigcup_{k \geq 0} D'_k$, it follows that there exists $i \geq 0$ for which there is a homomorphism from $[Q]$ to D'_i fixing \bar{x} . Let k be the minimum such i . We claim that φ_0^k is a rewriting of Q using \mathbf{V} . We need to show that for every database instance D , $Q(D) = \varphi_0^k(\mathbf{V}(D))$.

Consider $Q(D) \subseteq \varphi_0^k(\mathbf{V}(D))$. Let $\bar{u} \in Q(D)$ via some homomorphism h . We show that $\bar{u} \in \varphi_0^k(\mathbf{V}(D))$. To this end we construct below a tree of partial valuations of the variables in D_k into D , extending h , such that for each $i \geq 0$:

1. each node at depth $2i$ is a valuation μ of the variables of D_i which is a homomorphism from D_i to D such that $\mathbf{V}(D), \mu \models \phi_{V_i}$; in particular, μ provides a valuation for $\bar{\alpha}_i$
2. each node at depth $2i + 1$ is an extension μ' of its parent valuation to the variables of V'_{i+1} such that $\mathbf{V}(D), \mu' \models \phi_{V'_{i+1}}$; in particular, μ' provides a valuation for $\bar{\alpha}'_{i+1}$
3. the valuation at each node is consistent with the valuation at its parent node.

Note that the set of variables of φ_0^k is included in the set of variables in D_k . In particular, every valuation at level $2k$ provides values for all variables of φ_0^k .

The tree is constructed inductively as follows.

- the root consists of h ; in particular, h provides a valuation for the variables in $\bar{\alpha}_0 \cup \bar{x}$ sending \bar{x} to \bar{u} . By definition, this valuation satisfies (1).
- consider a node at depth $2i$ consisting of a valuation μ for the variables in D_i . Then, by construction of the tree of partial valuations ((2) and (3) above), μ has one child for each of its extensions μ' to the variables of V'_{i+1} such that $\mathbf{V}(D), \mu' \models \phi_{V'_{i+1}}$. By construction, (2) and (3) hold.

- consider a node at depth $2i+1$ consisting of a valuation μ' . By construction of the tree of partial valuations, $\mathbf{V}(D), \mu' \models \phi_{V'_{i+1}}$, so there exists an extension μ of μ' mapping $\{\mathbf{V}^{-1}(\bar{v}) \mid \bar{v} \in S'_{i+1}\}$ into D . By construction of D'_{i+1} , μ is a homomorphism from D'_{i+1} to D and $\mathbf{V}(D), \mu \models \phi_{V_{i+1}}$.

Using the above tree of valuations, we can show that $\mathbf{V}(D), h|_{\bar{x}} \models \varphi_0^k$. Intuitively, each valuation at depth $2i$ has as children all its extensions to $\bar{\alpha}'_{i+1}$ satisfying $\phi_{S'_{2i+1}}$, while each valuation at depth $2i+1$ has a single child inducing an extension to $\bar{\alpha}_{i+1}$. We show by reverse induction (from k to 0) that the nodes at even levels $2i$ provide valuation for the existentially quantified variables $\bar{\alpha}_i$ of φ_0^k , witnessing the fact that $\bar{u} \in \varphi_0^k(\mathbf{V}(D))$. Let us denote by $\bar{\varphi}_i^k$ the formula $\phi_{S_i} \wedge \forall \bar{\alpha}'_{i+1} (\phi_{S'_{i+1}} \rightarrow \varphi_{i+1}^k)$, that is φ_i^k without its existential quantification $\exists \bar{\alpha}_i$. Let $i = k$ and μ be a valuation at level $2k$. By construction of the tree of partial valuations, $\mathbf{V}(D), \mu \models \phi_{S_k} = \bar{\varphi}_k^k$. For the induction step, suppose that for every valuation μ at level $2(i+1)$, $\mathbf{V}(D), \mu \models \bar{\varphi}_{i+1}^k$. Consider a valuation ν at level $2i$. We have that $\mathbf{V}(D), \nu \models \phi_{S_i}$ and for every extension μ' of ν to α'_{i+1} satisfying $\phi_{S'_{i+1}}$, there exists an extension μ of μ' to $\bar{\alpha}_{i+1}$ such that $\mathbf{V}(D), \mu \models \bar{\varphi}_{i+1}^k$. In particular, $\mathbf{V}(D), \nu \models \exists \bar{\alpha}_{i+1} \bar{\varphi}_{i+1}^k = \varphi_{i+1}^k$. Thus,

$$\mathbf{V}(D), \nu \models \phi_{S_i} \wedge \forall \bar{\alpha}'_{i+1} (\phi_{S'_{i+1}} \rightarrow \varphi_{i+1}^k),$$

which completes the induction. Thus, for $i = 0$, we have that $\mathbf{V}(D), h|_{(\bar{\alpha}_0 \cup \bar{x})} \models \bar{\varphi}_0^k$, so $\mathbf{V}(D), h|_{\bar{x}} \models \exists \bar{\alpha}_0 \bar{\varphi}_0^k = \varphi_0^k$. In other words, $\bar{u} \in \varphi_0^k(\mathbf{V}(D))$. Thus, $Q(D) \subseteq \varphi_0^k(\mathbf{V}(D))$.

Now consider $\varphi_0^k(\mathbf{V}(D)) \subseteq Q(D)$. Suppose $\bar{u} \in \varphi_0^k(\mathbf{V}(D))$. We prove that there exists a homomorphism from D'_k into D mapping \bar{x} to \bar{u} . Since there is a homomorphism from $[Q]$ to D'_k fixing \bar{x} , this suffices to conclude.

We construct valuations h_i of the variables in \bar{x} and D'_{i-1} (for simplicity we assume that $D'_{-1} = \emptyset$) into D such that h_i induces a homomorphism of D'_{i-1} into D and $\mathbf{V}(D), h_i \models \varphi_i^k$. Note that for $i = 0$, h_0 simply maps \bar{x} to \bar{u} .

We shall use the following simple claim:

Claim 5.14 *Let h be a valuation of the variables in \bar{x} and D'_{i-1} that induces a homomorphism from D'_{i-1} into D . Suppose there exists a valuation g of the variables in S_i , compatible with h , such that $\mathbf{V}(D), g \models \phi_{S_i}$. Then there exists a valuation of the variables in \bar{x} and D'_i , extending h and inducing a homomorphism from D'_i to D .*

Proof: By construction, D'_i is obtained from D'_{i-1} by adding to it $\mathbf{V}^{-1}(S_i)$. Because $\mathbf{V}(D), g \models \phi_{S_i}$, $g(S_i) \subseteq \mathbf{V}(D)$. Thus, there is a homomorphism f consistent with h mapping $\mathbf{V}^{-1}(g(S_i))$ to D , and $h \cup f$ is a homomorphism from D'_i to D extending h . \square

We now construct the valuations h_i by induction, for $0 \leq i \leq k+1$ (h_{k+1} is the desired homomorphism from D'_k into D). Specifically, for each i we exhibit a valuation h_i on the variables in \bar{x} and D'_{i-1} that induces a homomorphism from D'_{i-1} to D and for which $\mathbf{V}(D), h_i \models \varphi_i^k$. For the basis, $\mathbf{V}(D), h_0 \models \varphi_0^k$ since $\bar{u} \in \varphi_0^k(\mathbf{V}(D))$. For the inductive step, consider i , $0 \leq i \leq k$ and suppose we have constructed h_i such that $\mathbf{V}(D), h_i \models \varphi_i^k$ and h_i induces a homomorphism from D'_{i-1} to D . Suppose first that $i = k$. Then, as $\varphi_k^k = \exists \bar{\alpha}_k \phi_{S_k}$, there exists a valuation g of $\bar{\alpha}_k$, compatible with h_i , such that $\mathbf{V}(D), g \models \phi_{S_k}$. By Claim 5.14 there exists an extension of h_k to a homomorphism h_{k+1} from D'_k to D , as desired. Suppose now that $i < k$. Then, as $\varphi_i^k = \exists \bar{\alpha}_i (\phi_{S_i} \wedge \forall \bar{\alpha}'_{i+1} (\phi_{S'_{i+1}} \rightarrow \varphi_{i+1}^k))$, it is possible to extend h_i with a valuation g of $\bar{\alpha}_i$ such that $\mathbf{V}(D), (h_i \cup g) \models [\phi_{S_i} \wedge \forall \bar{\alpha}'_{i+1} (\phi_{S'_{i+1}} \rightarrow \varphi_{i+1}^k)]$. By Claim 5.14, h_i can then be extended

to a homomorphism h_{i+1} from D'_i to D . It remains to show that with this valuation φ_{i+1}^k holds in $\mathbf{V}(D)$. By construction, h_{i+1} provides a valuation for all variables in S'_{i+1} , including $\bar{\alpha}'_{i+1}$. Furthermore, because by construction $S'_{i+1} \subseteq \mathbf{V}(D'_i)$, and h_{i+1} induces a homomorphism from D'_i to D , $h_{i+1}(S'_{i+1}) \subseteq \mathbf{V}(D)$, so $\mathbf{V}(D), h_{i+1} \models \phi_{S'_{i+1}}$. We conclude that $\mathbf{V}(D), h_{i+1} \models \varphi_{i+1}^k$. This completes the induction. \square

Remark 5.15 Recall that we earlier showed in Theorem 3.1, using Craig's Interpolation Theorem, that for any set \mathbf{V} of FO views and FO query Q such that $\mathbf{V} \xrightarrow{\infty} Q$ 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 5.13 provides a constructive algorithm to obtain the rewriting formula. Indeed, assume that $\mathbf{V} \xrightarrow{\infty} Q$ 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 successively 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 5.13 works only for CQ views and queries, while the interpolation argument applies to all FO views and queries.

5.4 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.

Monadic Views We first consider the special case when the views are monadic. A *monadic conjunctive query* (MCQ) is a CQ with one free variable. We show the following.

Theorem 5.16 (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 5.1.

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 5.17 Let \mathbf{V} be a set of MCQ views and $Q(x_1, \dots, x_k)$ a CQ query with free variables x_1, \dots, 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 $\mathbf{V} \rightarrow Q$ then Q is equivalent to $\bigwedge_{i=1}^k Q_i(x_i)$;
- (ii) if $Q = \bigwedge_{i=1}^k Q_i(x_i)$ then $\mathbf{V} \rightarrow Q$ iff $\mathbf{V} \rightarrow Q_i(x_i)$ for $1 \leq i \leq k$.

Proof: Consider (i). Suppose $\mathbf{V} \rightarrow Q$ 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 (see [27]) of $[Q]$. This clearly proves (i).

Consider instances A and B over σ defined as follows. Let D consist of k elements $\{a_1, \dots, a_k\}$. Let $A = \{R(a, \dots, a) \mid R \in \sigma, a \in D\}$, 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, $\langle a_1, \dots, a_k \rangle \in Q(A)$. 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 5.17, to establish Theorem 5.16 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 $Q(A) = Q(B)$. Recall that x is the free variable of V , hence $x \in V([V])$ and $(x, x) \in B$. Therefore $x \in Q(B)$. 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, which is used in Lemma 5.18 below.

We will need the following notions. For any instance D , we define the *type* of $a \in \text{adom}(D)$ to be $\tau(D, a) = \{i \mid a \in V_i(D)\}$ which is a subset of $[k] = \{1, \dots, k\}$. A type S is *realized* in D if $\tau(D, c) = S$ for some $c \in \text{adom}(D)$. A type S is *realizable* if it is realized in some D . A type is *maximal* if it is maximal under set inclusion among some set of types. We also denote by $\#(S, D)$ the number of elements $c \in \text{adom}(D)$ for which $\tau(D, c) = S$.

Given two instances D and D' over the same schema, we write $D' \rightarrow D$ if there is a homomorphism from D' to D . Suppose in addition that $D \subseteq D'$. We say that D' *retracts* to D , which we write $D' \hookrightarrow D$ if there is a homomorphism from D' to D fixing the domain of D . We call such a homomorphism a *retraction*.

The proof of Theorem 5.16 (i) is as follows. Let \mathbf{V} be a set of MCQ views and Q a CQ query with free variable x , both over database schema σ , such that $\mathbf{V} \twoheadrightarrow Q$. We want to show that Q has a CQ rewriting in terms of \mathbf{V} . By Proposition 5.17, we can assume wlog that Q is MCQ.

Set $A_0 = [Q]$ and $B_0 = \mathbf{V}^{-1}(\mathbf{V}([Q]))$. Let $C = \text{adom}(\mathbf{V}(A_0))$. Note that, by construction, $C \subseteq \text{adom}(\mathbf{V}(B_0))$. Also, note that $x \in C$. Indeed, suppose this is not the case. Let z be a new element and let $[Q]^z$ be the instance obtained from $[Q]$ by replacing x with z . Then $V([Q]) = V([Q]^z)$, hence $Q([Q]) = Q([Q]^z)$ as $V \twoheadrightarrow Q$, but $x \in Q([Q])$ and $x \notin Q([Q]^z)$, a contradiction.

We next prove Lemma 5.18 below, which allows us to conclude. Indeed, the lemma guarantees the existence of A and B such that $A \hookrightarrow A_0$, $B \hookrightarrow B_0$, and $V(A) = V(B)$. Then

1. $x \in Q(A_0)$ because $A_0 = [Q]$,
2. $x \in Q(A)$ because $A_0 \subseteq A$,
3. $x \in Q(B)$ because $\mathbf{V} \twoheadrightarrow Q$ and $V(A) = V(B)$, and therefore
4. $x \in Q(B_0)$ because $B \hookrightarrow B_0$ (and recall that $x \in \text{adom}(B_0)$).

This implies that there is a homomorphism from Q to B_0 preserving x , as desired. In view of Proposition 5.1, this implies that Q has a CQ rewriting in terms of \mathbf{V} , which concludes the proof of Theorem 5.16.

We next proceed with Lemma 5.18. Note that the statement is slightly more general than we need, since we do not even assume that $\mathbf{V} \rightarrow Q$.

Lemma 5.18 *Let \mathbf{V} be a set of MCQs over a database schema σ , Q an MCQ over the same schema σ , and let A_0 and B_0 be constructed as above. There exist finite instances A and B such that:*

1. $A \hookrightarrow A_0$,
2. $B \hookrightarrow B_0$, and
3. $\mathbf{V}(A) = \mathbf{V}(B)$.

Proof: As above, let $C = \text{adom}(\mathbf{V}(A_0))$. Recall that, by construction of B_0 , there exists a homomorphism h from B_0 to A_0 preserving C . We begin with the following observations:

- (i) every $a \in C$ has the same type in $\mathbf{V}(A_0)$ and $\mathbf{V}(B_0)$, and
- (ii) the maximal types realized in A_0 and B_0 are the same.

By construction, for every $a \in C$ and $V \in \mathbf{V}$, $a \in V(A_0)$ iff $a \in V(B_0)$. Thus, (i) holds. For part (ii), we show that every type S realized in A_0 is included in a type S' realized in B_0 , and every type S' realized in B_0 is included in a type S realized in A_0 . Suppose S is a type realized in A_0 . Let a be such that $\tau(A_0, a) = S$. By (i), since $a \in C$, $\tau(A_0, a) = \tau(B_0, a)$. Thus, $S \subseteq \tau(B_0, a)$. Now let S' be a type realized in B_0 . Let $b \in \text{adom}(\mathbf{V}(B_0))$ be such that $\tau(B_0, b) = S'$. Consider $h(b) \in \text{adom}(\mathbf{V}(A_0))$. Since h is a homomorphism from B_0 to A_0 , if $b \in V(B_0)$ then $h(b) \in V(A_0)$. Thus, $S' = \tau(B_0, b) \subseteq \tau(A_0, h(b))$. This establishes (ii).

Next, for each type $S \neq \emptyset$, let V_S be the query $\bigwedge_{i \in S} V_i(x_S)$, where x_S is a fresh variable (here $V_i(x_S)$ stands for the query over σ defining the i -th view). 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 frozen body, as needed. Clearly, if $S \subseteq S'$ then $V_S \rightarrow V_{S'}$.

We construct A and B in three steps.

- (a) Set A_1 to consist of the disjoint union of A_0 with the set of instances

$$\{V_S \mid S \text{ is realized in } A_0 \text{ or } B_0\}.$$

Similarly, let B_1 be the disjoint union of B_0 with the same set of instances. Note that $\tau(A_1, x_S) = \tau(B_1, x_S) = S$. This ensures that A_1 and B_1 realize the same set of types. Moreover, $A_1 \hookrightarrow A_0$ and $B_1 \hookrightarrow B_0$, because A_0 and B_0 have the same maximal realized types (by (ii) above) so $V_S \hookrightarrow A_0$ and $V_S \hookrightarrow B_0$ for every S realized in A_0 or in B_0 .

- (b) Now consider an arbitrary instance D and $y \in \text{adom}(D)$. We can construct $D' \supseteq D$ such that $\text{adom}(D') - \text{adom}(D) = \{y'\}$ as follows. Make every relation $D'(R)$ contain all the tuples of $D(R)$ together with the set of tuples t' where $t \in D(R)$ and where every occurrence of y in t has been replaced with y' . It is easy to verify that every $z \in \text{adom}(D)$ has the same type in D and D' and that y' has the same type in D' as y . Furthermore $D' \hookrightarrow D$ by the homomorphism g which is the identity on D and where $g(y') = y$. By repeatedly adding new elements to an instance D in this way, we can increase $\#(S, D)$ for any type S realized in D to whatever count we desire without changing the type of any other element. By an easy induction, it follows that we can construct A_2 and B_2 such that $A_2 \hookrightarrow A_1 \hookrightarrow A_0$ and $B_2 \hookrightarrow B_1 \hookrightarrow B_0$ and

such that for all $S \subseteq [k]$ we have $\#(S, A_2) = \#(S, B_2)$. In particular, this means that $\mathbf{V}(A_2)$ and $\mathbf{V}(B_2)$ are isomorphic by an isomorphism α that preserves C . We assume that all new elements added in constructing A_1, A_2, B_1 , and B_2 are distinct.

- (c) Let $\bar{\alpha}$ be the extension of the isomorphism α obtained in (b) with the identity on $\text{adom}(A_2) - \text{adom}(\mathbf{V}(A_2))$. Let $A = \bar{\alpha}(A_2)$ and $B = B_2$. Clearly, $A \leftrightarrow A_0$, $B \leftrightarrow B_0$ and $\mathbf{V}(A) = \mathbf{V}(B)$ as desired. □

We illustrate the constructions of Lemma 5.18 with an example

Example 5.19 Let σ consist of a single binary relation E so that instances over E are directed graphs. Let $V_1(x) = \exists y E(x, y)$ and $V_2(x) = \exists y E(y, x)$. That is, V_1 returns the set of all nodes with an outgoing edge and V_2 returns the set of all nodes with an incoming edge. Consider the query $Q(x) = E(x, x)$, which returns the set of all nodes with a self-loop. Then $A_0 = \{E(x, x)\}$, $V_1(A_0) = V_2(A_0) = \{x\}$, and $B_0 = \{E(y, x), E(x, z)\}$. The types $\{1\}$, $\{2\}$ and $\{1, 2\}$ are all realized in both A_0 and B_0 , whereas \emptyset is not. Furthermore,

$$V_{\{1\}} = \{E(a, b)\}, \quad V_{\{2\}} = \{E(c, d)\} \quad V_{\{1,2\}} = \{E(e, f), E(f, g)\},$$

for some newly chosen a, b, c, d, e, f, g . In step (a) of the construction, we obtain A_1 and B_1 by adding disjoint versions of $V_{\{1\}}, V_{\{2\}}, V_{\{1,2\}}$ to A_0 and B_0 . This yields

$$A_1 = \{E(x, x), E(a, b), E(c, d), E(e, f), E(f, g)\}$$

and

$$B_1 = \{E(y, x), E(x, z), E(a', b'), E(c', d'), E(e', f'), E(f', g')\}.$$

The elements of type $\{1\}$ in A_1 are a, c, e and in B_1 they are y, a', c', e' . The elements of type $\{2\}$ in A_1 are b, d, g and in B_1 they are z, b', d', g' . The elements of type $\{1, 2\}$ in A_1 are $\{x, f\}$ and in B_1 they are $\{x, f'\}$. Notice that both A_1 and B_1 have two elements of type $\{1, 2\}$, but disagree on the counts of the types $\{1\}$ and $\{2\}$. In step (b) of the construction, we therefore add to A_1 an element u of type $\{1\}$ and an element v of type $\{2\}$ to obtain A_2 . We set $B_2 = B_1$ and

$$A_2 = \{E(x, x), E(a, b), E(c, d), E(e, f), E(f, g), E(u, b), E(a, v)\}.$$

Here we obtained $E(u, b)$ by replacing a with u in $E(a, b)$. Similarly, we obtained $E(a, v)$ by replacing b with v in $E(a, b)$. Now we define the isomorphism α from $\mathbf{V}(A_2)$ to $\mathbf{V}(B_2)$ as follows:

α	
	$x \quad x$
	$u \quad y$
	$v \quad z$
	$a \quad a'$
	$b \quad b'$
	$c \quad c'$
	$d \quad d'$
	$e \quad e'$
	$f \quad f'$
	$g \quad g'$

Note that $V_1(A_2) = \{u, a, c, e, x, f\}$ and $V_2(A_2) = \{v, b, d, g, x, f\}$ so $\text{adom}(A_2) - \text{adom}(\mathbf{V}(A_2)) = \emptyset$. Thus, $\bar{\alpha} = \alpha$ and step (c) yields

$$A = \bar{\alpha}(A_2) = \{E(x, x), E(a', b'), E(c', d'), E(e', f'), E(f', g'), E(y, b), E(a, z)\}$$

while B remains equal to B_2 so

$$B = \{E(y, x), E(x, z), E(a', b'), E(c', d'), E(e', f'), E(f', g')\}.$$

This concludes the construction. Note that $A \hookrightarrow A_0$, $B \hookrightarrow B_0$, and $\mathbf{V}(A) = \mathbf{V}(B)$ as desired. As an aside, note that \mathbf{V} does not determine Q , since there is no homomorphism from Q to B_0 . Indeed, determinacy is not required by the lemma and the construction does not assume it.

One might wonder if the nice behavior exhibited in Theorem 5.16 for monadic CQs extends to more powerful languages. Proposition 4.2 and Proposition 4.3 already settled this in the negative for UCQs and CQ^\neq , by showing that $Q_{\mathbf{V}}$ may be non-monotonic even for monadic views and query Q . Thus, the good behavior of CQs in the monadic case is lost with even small extensions to the language.

Single-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 5.2. 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

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

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 5.20 *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 5.1, 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 5.1. 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 5.1. 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 5.21 *Under the conditions of Theorem 5.20, S_0 and S'_1 have disjoint domains.*

Proof: Suppose that $\langle a, b \rangle \in P_k(D'_0)$ for some $a \in \text{adom}(S_0)$ (the case when $b \in \text{adom}(S_0)$ is similar). We show that b cannot be in $\text{adom}(D'_0) - \text{adom}(S_0)$. Since $\langle a, b \rangle \in P_k(D'_0)$, there is a path of length k from a to b in D'_0 . 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 $d(a, b) = k$ and $a \in \text{adom}(S_0)$, by construction of D'_0 , b can only be in $\text{adom}(S_0)$. \square

Next, let us construct a special binary relation M as follows. The domain of M consists of $\text{adom}(S'_1)$ and, for each $\alpha \in \text{adom}(S'_1)$, new distinct elements $x_1^\alpha, \dots, x_{k-1}^\alpha$. M has the following edges:

$$\begin{aligned} \text{for each } \alpha \in \text{adom}(S'_1), \text{ the edges } & \alpha \rightarrow x_1^\alpha \rightarrow \dots \rightarrow x_{k-1}^\alpha; \\ \text{and for each } \langle \alpha, \beta \rangle \in S'_1, \text{ an edge } & x_{k-1}^\alpha \rightarrow \beta. \end{aligned}$$

The following is easily shown by construction of M .

Lemma 5.22 *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 $\langle x_i^\alpha, x_i^\beta \rangle \in P_k(M)$ for $1 \leq i \leq k-1$.*

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 5.21, respectively Lemma 5.22, $P_k(J)$ 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 $\{P_k\} \rightarrow Q$, $Q(I) = Q(J)$. 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 5.1, Q has a CQ rewriting in terms of P_k . This completes the proof of Theorem 5.20. \square

As discussed in Remark 5.3, Foto Afrati recently considered in [3] the case of multiple path views and a single path query, and showed that determinacy remains decidable in this case. Furthermore, if the views determine the query, an FO rewriting of the query in terms of the views exists and can be effectively computed.

Remark 5.23 *As mentioned in Remark 3.11, Maarten Marx provides in [30] another well-behaved class of CQs with respect to determinacy and rewriting. The class is obtained by applying the packed condition used for FO to CQs, yielding Packed CQ (PCQ) (see Remark 3.11). The resulting fragment is a slight extension of the well-known acyclic CQs [20]. It is shown in [30] that unrestricted and finite determinacy coincide for PCQs, both are decidable, and PCQ is complete for PCQ-to-PCQ rewriting. These results also extend to unions of PCQs.*

5.5 One View versus Multiple Views

So far we have discussed the case of multiple views and a single query. But is it essential to have multiple views? That is, can the problem of whether a set of CQ views determines a CQ query be reduced to the problem of whether a single CQ view determines a CQ query? It turns out that, in the unrestricted case (finite and infinite instances), the answer is yes. In the finite case the problem is still open.

Definition 5.24 Given a set $\mathbf{V} = \{V_1, \dots, V_k\}$ of CQ views with disjoint sets of variables, of arities a_1, \dots, a_k respectively, we define the product query $\otimes \mathbf{V} = V_1 \times \dots \times V_k$ to be the conjunctive view of arity $\sum_i a_i$ whose set of free variables is the union of those of V_1, \dots, V_k and whose body is the conjunction of the bodies of V_1, \dots, V_k . If the disjointness condition does not hold, we define $\otimes \mathbf{V}$ by first renaming the variables so as to obtain disjointness.

The following is obvious from the definition.

Lemma 5.25 Let $\mathbf{V} = \{V_1, \dots, V_k\}$ be a set of CQs with disjoint sets of variables. Let A and B be database instances. If $\otimes \mathbf{V}(A) = \otimes \mathbf{V}(B) \neq \emptyset$ then $\mathbf{V}(A) = \mathbf{V}(B)$.

It is also useful to note the following.

Lemma 5.26 Let $\mathbf{V} = \{V_1, \dots, V_k\}$ be a set of CQs with disjoint sets of variables and Q a CQ such that $\mathbf{V} \xrightarrow{\infty} Q$. If $i \in [1, k]$ is such that $V_i([Q]) = \emptyset$ then $(\mathbf{V} - \{V_i\}) \xrightarrow{\infty} Q$.

Proof: Recall the construction of D'_∞ from \mathbf{V} and Q . By Proposition 5.12, $\mathbf{V} \xrightarrow{\infty} Q$ iff $\bar{x} \in Q(D'_\infty)$ where \bar{x} is the set of free variables of Q . Since $V_i([Q]) = \emptyset$, there is no homomorphism mapping V_i into $[Q]$. However, by Proposition 5.11, there is a homomorphism from D'_∞ into $[Q]$. It follows that there is no homomorphism from V_i to D'_∞ . Consequently, V_i is never used in the construction of D'_∞ . Thus, D'_∞ is constructed using only Q and $\mathbf{V} - \{V_i\}$. Since $\bar{x} \in Q(D'_\infty)$, it follows that $(\mathbf{V} - \{V_i\}) \xrightarrow{\infty} Q$. \square

We now have the following.

Theorem 5.27 Let \mathbf{V} be a set of CQ views and Q a CQ query. There exists a single CQ view W such that $\mathbf{V} \xrightarrow{\infty} Q$ iff $\{W\} \xrightarrow{\infty} Q$.

Proof: Let \mathbf{V}' be obtained by removing from \mathbf{V} all views V for which $V([Q]) = \emptyset$. Note first that $\mathbf{V}' \neq \emptyset$. Indeed, otherwise $\mathbf{V}([Q]) = \emptyset$. But we also have $\mathbf{V}(\emptyset) = \emptyset$, while $Q([Q]) \neq \emptyset$ and $Q(\emptyset) = \emptyset$. This is a contradiction with $\mathbf{V} \xrightarrow{\infty} Q$. Thus, $\mathbf{V}' \neq \emptyset$.

Next, since $\mathbf{V}' \subseteq \mathbf{V}$, if $\mathbf{V}' \xrightarrow{\infty} Q$ then $\mathbf{V} \xrightarrow{\infty} Q$. Conversely, by Lemma 5.26, if $\mathbf{V} \xrightarrow{\infty} Q$ then $\mathbf{V}' \xrightarrow{\infty} Q$. Thus, $\mathbf{V} \xrightarrow{\infty} Q$ iff $\mathbf{V}' \xrightarrow{\infty} Q$. Now let $W = \otimes \mathbf{V}'$. It is left to show that $\mathbf{V}' \xrightarrow{\infty} Q$ iff $\otimes \mathbf{V}' \xrightarrow{\infty} Q$. Suppose first that $\otimes \mathbf{V}' \xrightarrow{\infty} Q$. We show that $\mathbf{V}' \xrightarrow{\infty} Q$. Let A, B be database instances such that $\mathbf{V}'(A) = \mathbf{V}'(B)$. Clearly, $\otimes \mathbf{V}'(A) = \otimes \mathbf{V}'(B)$. Since $\otimes \mathbf{V}' \xrightarrow{\infty} Q$, it follows that $Q(A) = Q(B)$. Thus, $\mathbf{V}' \xrightarrow{\infty} Q$. Conversely, suppose $\mathbf{V}' \xrightarrow{\infty} Q$ and let A, B be database instances such that $\otimes \mathbf{V}'(A) = \otimes \mathbf{V}'(B)$. If $\otimes \mathbf{V}'(A) = \otimes \mathbf{V}'(B) \neq \emptyset$ then by Lemma 5.25, $\mathbf{V}'(A) = \mathbf{V}'(B)$. Since $\mathbf{V}' \xrightarrow{\infty} Q$, it follows that $Q(A) = Q(B)$. Finally, suppose $\otimes \mathbf{V}'(A) = \otimes \mathbf{V}'(B) = \emptyset$. Then there exist $T_1, T_2 \in \mathbf{V}'$ such that $T_1(A) = \emptyset$ and $T_2(B) = \emptyset$. Since, by construction of \mathbf{V}' , there is a homomorphism from T_1 and T_2 to $[Q]$, this implies that $Q \subseteq T_1$ and $Q \subseteq T_2$ and therefore $Q(A) = Q(B) = \emptyset$. Thus, $\otimes \mathbf{V}' \xrightarrow{\infty} Q$. \square

Remark 5.28 It remains open whether multiple CQ views can be replaced by single views in the case of finite determinacy. The difficulty is that Lemma 5.26 may no longer hold, since in the finite case it is not known whether views without a homomorphism into $[Q]$ can be eliminated. However, it is easy to show, similarly to the above, that if \mathbf{V} is a set of CQ views such that every view in \mathbf{V} has a homomorphism into $[Q]$, then $\mathbf{V} \rightarrow Q$ iff $\otimes \mathbf{V} \rightarrow Q$.

6 Conclusion

The contribution of this paper is a systematic study of determinacy and its connection to rewriting for a variety of view and query languages ranging from FO to CQ. While the questions were settled for many languages, several interesting problems remain open.

From a practical point of view, the case of CQs is the most interesting, because this is the most widely used class of queries, and because, as shown by our negative results, even slight extensions lead to undecidability. For CQs, 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 effective FO rewritings whenever \mathbf{V} determines Q in the unrestricted case, while the best complete language in the finite case remains $\exists\text{SO} \cap \forall\text{SO}$. 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 exhibited 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. Recall that even a slight increase in the power of the rewrite language, such as using UCQ or CQ^\neq instead of CQ, does not help. Indeed, a consequence of Theorem 5.6 is the following: if CQ is not sufficient to rewrite Q in terms of \mathbf{V} , then a non-monotonic language is needed for the rewriting.

For $\mathcal{L} \in \{\text{CQ}, \text{CQ}^\neq, \text{UCQ}\}$, we know that $\exists\text{SO} \cap \forall\text{SO}$ is complete for \mathcal{L} -to- \mathcal{L} rewritings. It is of interest to know if there are less powerful languages that remain complete for such rewritings, such as FO or FO+LFP. This however remains open.

Determinacy and rewriting naturally lead to the following general problem (solved so far only for simple languages). Suppose \mathcal{R} is complete for \mathcal{V} -to- \mathcal{Q} rewritings. Is there an algorithm that, given $\mathbf{V} \in \mathcal{V}$ and $Q \in \mathcal{Q}$ such that $\mathbf{V} \rightarrow Q$, produces $R \in \mathcal{R}$ such that $Q \Rightarrow_{\mathbf{V}} R$? For example, we know that FO is complete for FO-to-FO rewritings in the unrestricted case, but the rewritings cannot be effectively computed. In contrast, effective FO rewritings can be obtained for CQ-to-CQ rewritings in the unrestricted case (Theorem 5.13).

Finally, an interesting direction to be explored is *instance-based* determinacy and rewriting, where determinacy and rewriting are considered relative to a *given* view instance. Such results have been obtained for regular path queries in [11].

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. Rewriting Conjunctive Queries Determined by Views. *MFCSS*, 2007, 78-89.
- [4] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. *PODS* 2002, 209-220.
- [5] S. Agrawal, S. Chaudhuri, and V. Narasayya. Automated selection of materialized views and indexes in Microsoft SQL Server. *VLDB* 2000, 496-505.
- [6] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.* 6(4): 557-575, 1981.
- [7] E. Beth. On Padoa's method in the theory of definition. *Indagationes Mathematicae*, 15:330-339, 1953.
- [8] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer, 1997.
- [9] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Rewriting of regular expressions and regular path queries. *PODS* 1999, 194-204.
- [10] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. View-based query processing for regular path queries with inverse. *PODS* 2000, 58-66.
- [11] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Lossless regular views. *PODS* 2002, 247-258.
- [12] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. *STOC* 1977, 77-90.
- [13] C. C. Chang and H. J. Keisler. *Model Theory*. North-Holland, 1977.
- [14] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *ICDT* 1997, 56-70.
- [15] W. Craig. Three uses of the Herbrand-Gentzen theorem in relation to model theory and proof theory. *Journal of Symbolic Logic*, 22:269-285, 1957.
- [16] S. Dar, M.J. Franklin, B. Jonsson, D. Srivastava and M. Tan. Semantic data caching and replacement. *VLDB* 1996, 330-341.
- [17] A. Deutsch, L. Popa, and V. Tannen. Physical data independence, constraints and optimization with universal plans. *VLDB* 1999, 459-470.
- [18] O. Duschka and M.R. Genesereth. Answering recursive queries using views. *PODS* 1997, 109-116.

- [19] H-D Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
- [20] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comput. Syst. Sci.*, 66(4):775-808, 2003.
- [21] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719-1742, 1999.
- [22] S. Grumbach, Z. Lacroix, and S. Lindell. Generalized implicit definitions on finite structures. In *Computer Science Logic*, 1995, 252-265.
- [23] S. Grumbach, M. Rafanelli, and L. Tininini. Querying aggregate data. PODS 1999, 174-184.
- [24] S. Grumbach and L. Tininini. On the content of materialized aggregate views. PODS 2000, 47-57.
- [25] Yuri Gurevich. The word problem for some classes of semigroups. *Algebra and Logic*, 5(4):25-35, 1966. (Russian).
- [26] D. Leinders, M. Marx, J. Tysiewicz, and J. van den Bussche. The semijoin algebra and the guarded fragment. *Journal of Logic, Language and Information*, 14:331-343, 2005.
- [27] Leonid Libkin. *Elements of finite model theory*. Springer, 2004.
- [28] S. Lindell. *The Logical Complexity of Queries on Unordered Graphs*. PhD thesis, University of California at Los Angeles, 1987.
- [29] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. PODS 1995, 95-104.
- [30] Maarten Marx. *Queries determined by views: pack your views*. PODS 2007, 23-30.
- [31] Alan Nash, Luc Segoufin and Victor Vianu. Determinacy and Rewriting of Conjunctive Queries Using Views: A Progress Report. ICDT 2007.
- [32] Martin Otto. An Interpolation Theorem. In *Bulletin of Symbolic Logic*, volume 6, 2000, pp.447-462.
- [33] Emil L. Post. Recursive unsolvability of a problem of Thue. *Journal on Symbolic Logic*, 12(1):1-11, 1947.
- [34] A. Rajaraman, Y. Sagiv, and J.D. Ullman. Answering queries using templates with binding patterns. PODS 1995, 105-112.
- [35] L. Segoufin and V. Vianu. Views and queries: determinacy and rewriting. PODS 2005, 49-60.
- [36] A. Tarski. Einige methodologische Untersuchungen über die Definierbarkeit der Begriffe. *Erkenntnis*, 5:80-100, 1935.
- [37] O.G. Tsatalos, M.H. Solomon and Y.E. Ioannidis. Describing and using query capabilities of heterogeneous sources. VLDB 1997, 256-265.

[38] J.D. Ullman. Information integration using logical views. ICDT 1997, 19-40.

[39] H.Z. Yang and P.A. Larson. Query transformation for PSJ-queries. VLDB 1987, 245-254.