Views and Queries: Determinacy and Rewriting*

Luc Segoufin
INRIA-Futurs, Université de Paris Sud
http://www-rocq.inria.fr/~segoufin

Victor Vianu†
U.C. San Diego
vianu@cs.ucsd.edu

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 $\mathbf{V} \twoheadrightarrow Q$) iff $\mathbf{V}(D_1) = \mathbf{V}(D_2) \rightarrow Q(D_1) = Q(D_2)$, for all database instances D_1 and D_2. Intuitively, determinacy says that $\mathbf{V}$ provides enough information to uniquely determine the answer

*An extended abstract of this paper appeared in *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems* (PODS), 2005.

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

to Q. However, it does not say that this can be done effectively, or using a particular query language. The next formulation is language specific: a query Q can be *rewritten* in terms of $\mathbf{V}$ using a rewriting language $\mathcal{R}$ iff there exists some query $R \in \mathcal{R}$ such that $Q(D) = R(\mathbf{V}(D))$ for all databases D. 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 $\mathbf{V} \twoheadrightarrow Q$. The converse is generally not true. Given a view language $\mathcal{V}$ and query language $\mathcal{Q}$, if $\mathcal{R}$ can be used to rewrite a query Q in $\mathcal{Q}$ in terms of a set of views $\mathbf{V}$ in $\mathcal{V}$ whenever $\mathbf{V} \twoheadrightarrow Q$, we say that $\mathcal{R}$ is 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 $\mathbf{V} \twoheadrightarrow Q$ for $\mathbf{V}$ in $\mathcal{V}$ and Q in $\mathcal{Q}$?

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

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 [20]. 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 CQs with Boolean or unary answer (for single views). Interestingly, the *unrestricted* variant of determinacy is easier to settle: it is decidable for CQs, and equivalent to the existence of a CQ rewriting of the query in terms of the view.

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 $\mathbf{V} \twoheadrightarrow Q$. Consider the query $Q_{\mathbf{V}}$ associating to each view answer to $\mathbf{V}(D)$ the corresponding query answer $Q(D)$, where D is a database instance. In other words[1] , $Q = Q_{\mathbf{V}} \circ \mathbf{V}$. 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 in terms of $\mathbf{V}$.

[1]For 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 $f \circ g(x) = f(g(x))$.

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 [11]. 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.

For views expressed in weaker languages, less powerful rewriting languages are needed. If views are expressed in ∃FO (existential FO), FO is still not complete for ∃FO-to-FO rewritings. However, both ∃SO and ∀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 ∃FO-to-FO rewritings must be able to express all queries in ∃SO ∩ ∀SO. The lower bound holds even if views are restricted to UCQs. The proof uses results on the expressive power of implicit definability [22, 17]. It turns out that FO does not become complete as a rewriting language even if queries are in $\mathrm{CQ}^{\neg}$ (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]).

Consider UCQ views and queries. Similarly (but for different reasons), UCQ is not complete for UCQ-to-UCQ rewritings, nor are much more powerful languages such as Datalog $^{\neq}$. This is due the fact that, for UCQ views $\mathbf{V}$ and queries Q such that $\mathbf{V} \twoheadrightarrow Q$, the query $Q_{\mathbf{V}}$ is generally not monotonic. Thus, any language complete for UCQ-to-UCQ rewritings must be able to express non-monotonic queries. This also turns out to hold for $\mathrm{CQ}^{\neq}$-to-CQ rewritings.

The rewriting problem remains open for CQs. Thus, it is not known whether CQ is complete for CQ-to-CQ rewritings. In fact, the following questions are all open and turn out to be equivalent:

1. CQ is complete for CQ-to-CQ rewritings;

2. for CQ views $\mathbf{V}$ and queries Q, the query $Q_{\mathbf{V}}$ is monotonic.

Related work Answering queries using views arises in numerous contexts including data integration [27], query optimization and semantic caching [13], data warehousing [4], support of physical data independence by describing storage schemas as views [14, 26, 28], etc. The problem comes in several flavors, depending on assumptions on the views and their use. Mainly, the different settings vary along these dimensions:

(i) assumptions on the views: these may be *exact* (i.e. contain precisely the set of tuples in their definitions), or just *sound* (they provide only a subset of the tuples in the answer)

(ii) how the views are used: *query rewriting* requires reformulating the query in terms of the views, using some query language. One may require an *equivalent* rewriting, or just a *maximally contained* one. Another use of views is called *query answering*. This consists of finding all *certain* answers to a query given an instance of the view [1].

In our investigation, we focus on exact view definitions, and equivalent query rewritings, with the accompanying information-theoretic notion of determinacy. 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 [23] 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 [12]. Answering queries using views in the presence of binding patterns is considered in [25]. Views and queries defined by CQs with arithmetic comparisons over dense orders are considered in [3], where it is shown that the existence of an equivalent rewriting using Datalog with comparisons is decidable. The problem for recursive queries is considered in [15], 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 [7, 8, 9]. Here the views and queries are defined by regular path expressions, possibly including inverse edge navigation. In [7] 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 [18, 19], 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, [19] 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 [9], the authors consider the notion of *lossless* view with respect to a query, in the context of regular path queries on semi-structured data. Losslessness is considered under the exact view assumption and under the sound view assumption. In the first case, losslessness is equivalent to determinacy and it remains open whether losslessness is decidable for regular path views and queries. In the second case, losslessness is shown to be decidable using automata-theoretic techniques. Again, these results have no bearing upon ours because of the differences in the settings and because we consider exact views.

In the context of view updates, Bancilhon and Spyratos introduced the notion of *view complement* [5]. 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 $cert_Q(E) = \cap\{Q(D) \mid \mathbf{V}(D) = E\}$. The data complexity of computing $cert_Q(E)$ 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 $\mathbf{V}$, $R(E) = cert_Q(E)$. 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 $V \twoheadrightarrow Q$ then for every database D, $Q(D) = cert_Q(\mathbf{V}(D))$. 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, $cert_Q(E) = \cap\{Q(D) \mid E \subseteq \mathbf{V}(D)\}$. If $\mathbf{V} \twoheadrightarrow Q$, it is easily seen that $cert_Q(E) = Q_{\mathbf{V}}(E)$ 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

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

Figure 1: Query languages used in the paper

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. We make use of these connections in Section 5.

Organization After introducing some basic concepts in Section 2, we discuss determinacy and rewriting in the unrestricted case (in which instances can be finite or infinite) in Section 3. Finite determinacy is discussed next in Section 4 and finite rewriting in Section 5.

2 Basic concepts and notation

We begin with some basic definitions and notation. A database schema σ is a finite set of relation symbols with associated non-negative arities. A relation with arity zero is referred to as a *proposition*. A database instance D over σ associates a relation $\mathrm{D}(R)$ of appropriate arity with values from some fixed infinite domain **dom** to each relation symbol R in σ (true/false for propositions). The active domain of an instance D consists of the set of elements in **dom** occurring in D and is denoted $adom(D)$. 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. Queries are defined as usual, as computable mappings from instances of an input schema to instances of an output schema that are generic, i.e. commute with isomorphisms of **dom** (e.g., see [2]). We assume familiarity with the query languages in Figure 1. As usual in query languages, all of these languages (over relational vocabulary) may refer to values from **dom**, always interpreted as themselves. This differs from constants in logic, which are part of the vocabulary and are interpreted as arbitrary values from the universe of the structure.

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

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

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

3 Unrestricted Determinacy and Rewriting

We begin by considering the unrestricted variant of determinacy and rewriting, in which database instances are allowed to be arbitrary (finite or infinite). We look at the two important extremes along the spectrum of languages we study: FO and CQ. For FO, determinacy is clearly undecidable (the proof is the same as for the finite case, see Proposition 4.1). However, FO is complete for FO-to-FO rewritings. For CQ, determinacy is decidable and CQ is complete for CQ-to-CQ rewritings.

We begin with the completeness of FO.

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

Proof: 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 instance (finite or infinite) satisfying φ also satisfies ψ.

Suppose $\mathbf{V} \twoheadrightarrow Q$ over arbitrary instances, where $\mathbf{V}$ and Q are FO views and queries over schema σ. Let σ_1, σ_2 be disjoint copies of σ, and $\mathbf{V}_i, Q_i$, $i = 1, 2$, be the versions of $\mathbf{V}$ and Q using σ_1 and σ_2. Given two queries Q and Q' with the same output schema we will write $Q = Q'$ for the formula $\forall \bar{x}\ Q(\bar{x}) \leftrightarrow Q'(\bar{x})$. We will also write $\mathbf{V}_i = \sigma_{\mathbf{V}}$ for $\bigwedge_{V \in \sigma_{\mathbf{V}}} Q_V^i = V$ and $\mathbf{V}_1 = \mathbf{V}_2$ for $\bigwedge_{V \in \sigma_{\mathbf{V}}} Q_V^1 = Q_V^2$. Because $\mathbf{V} \twoheadrightarrow Q$, we have that $\mathbf{V}_1 = \mathbf{V}_2 \models Q_1 = Q_2$. In other words, for instances over the schema $\sigma_1 \cup \sigma_2 \cup \sigma_{\mathbf{V}}$ we have:

$$\mathbf{V}_1 = \sigma_{\mathbf{V}} \wedge \mathbf{V}_2 = \sigma_{\mathbf{V}} \models Q_1 = Q_2.$$

Let k be the arity of Q and let $\bar{c}$ be a vector of k constant symbols. For instances over the schema $\sigma_1 \cup \sigma_2 \cup \sigma_{\mathbf{V}} \cup \bar{c}$ we have:

$$(\mathbf{V}_1 = \sigma_{\mathbf{V}} \wedge \mathbf{V}_2 = \sigma_{\mathbf{V}}) \models Q_1(\bar{c}) \leftrightarrow Q_2(\bar{c})$$

This implies:

$$(\mathbf{V}_1 = \sigma_{\mathbf{V}} \wedge Q_1(\bar{c})) \models (\mathbf{V}_2 = \sigma_{\mathbf{V}} \rightarrow Q_2(\bar{c}))$$

Note that the schema common to both sentences is $\sigma_{\mathbf{V}} \cup \bar{c}$. We can now apply Craig's Interpolation Theorem (see [11]) to the sentences above and obtain an FO formula $\theta(\sigma_{\mathbf{V}})(\bar{c})$ over schema $\sigma_{\mathbf{V}} \cup \bar{c}$ such that:

$$(\dagger)\quad (\mathbf{V}_1 = \sigma_{\mathbf{V}} \wedge Q_1(\bar{c})) \models \theta(\sigma_{\mathbf{V}})(\bar{c})$$

and

$$(\ddagger)\quad \theta(\sigma_{\mathbf{V}})(\bar{c}) \models (\mathbf{V}_2 = \sigma_{\mathbf{V}} \rightarrow Q_2(\bar{c}))$$

We now show that $\theta(\sigma_{\mathbf{V}})$ is a rewriting of Q using V. From ($\dagger$) we obtain (renaming symbols in σ_1 to symbols in σ): $\mathbf{V} = \sigma_{\mathbf{V}} \models Q(\bar{c}) \rightarrow \theta(\sigma_{\mathbf{V}})(\bar{c})$ and from ($\ddagger$) $\mathbf{V} = \sigma_{\mathbf{V}} \models \theta(\sigma_{\mathbf{V}})(\bar{c}) \rightarrow Q(\bar{c})$. We thus have: $\mathbf{V} = \sigma_{\mathbf{V}} \models q(\sigma)(\bar{c}) \leftrightarrow \theta(\sigma_{\mathbf{V}})(\bar{c})$ and by universality of constants $\mathbf{V} = \sigma_{\mathbf{V}} \models Q = \theta(\sigma_{\mathbf{V}})$. $\square$

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 [16]. Indeed, Theorem 3.1 does not hold in the finite case, as shown next.

Example 3.2 Let σ be a database schema and $\sigma_{\leq}$ the extension of σ with a binary relation "$\leq$". Let $\varphi(\leq)$ be an FO sentence over $\sigma_{\leq}$. 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 it 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 $\theta(true/R_\psi)$, 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.

4 Finite Determinacy

In this section, we consider determinacy when database instances are restricted to be finite. Instances are henceforth assumed to be finite in all definitions, including determinacy. We begin with several easy but useful observations.

Proposition 4.1 *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 $\mathbf{V} \twoheadrightarrow Q$, 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, $\mathbf{V} \twoheadrightarrow Q$ 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, $\mathbf{V} \twoheadrightarrow Q$ iff φ is valid. □

Corollary 4.2 *If $\mathcal{V}$ is FO or $\mathcal{Q}$ is FO, it is undecidable whether $\mathbf{V} \twoheadrightarrow Q$ 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, $\mathbf{V} \twoheadrightarrow Q$ 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 [6]. 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 $\mathbf{V} \twoheadrightarrow Q$. The following provides useful information on the behavior of the mapping $Q_{\mathbf{V}}$ associating to every instance $\mathbf{V}(D)$ in the image of $\mathbf{V}$ the corresponding $Q(D)$.

Proposition 4.3 *Suppose* $\mathbf{V}$ *and* Q *are computable views and queries over database schema* σ *and* $\mathbf{V} \twoheadrightarrow Q$*. 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)* $adom(Q(D)) \subseteq adom(\mathbf{V}(D))$*, and (ii) every permutation of* **dom** *that is an automorphism of* $\mathbf{V}(D)$ *is also an automorphism of* $Q(D)$*.*

Proof: Let f be a permutation of **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 4.4 *Suppose* $\mathbf{V}$ *and* Q *are as in Proposition 4.3 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* $\mathbf{V} \twoheadrightarrow Q$ *but* $Q_{\mathbf{V}}$ *cannot be extended to a halting computable query on all of* $\mathcal{I}(\sigma_{\mathbf{V}})$*.*

Unions of conjunctive 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 using no constant values.

Theorem 4.5 *There exists a database schema* σ *and a fixed view* $\mathbf{V}$ *over* σ *defined by UCQs without constants, for which it is undecidable, given a UCQ query* Q *without constants over* σ*, whether* $\mathbf{V} \twoheadrightarrow Q$*.*

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 [20].

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 $f : X \times X \to X$ 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 $\mathbf{V} \twoheadrightarrow Q_{H,F}$ 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 $Q_{V_1}(x,y,z)$ be $R(x,y,z)$, Q_{V_2} be $p_1 \vee p_2$, Q_{V_3} be $p_1 \wedge p_2$, and, for each equation α of the form $S = T$ in the list above let Q_{V_α} be $(p_1 \wedge S) \vee (p_2 \wedge T)$. Let $\mathbf{V}$ consist of the queries Q_{V_i} and Q_{V_α} so defined.

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.

We now define $Q_{H,F}$. Given H and $F = \{x = y\}$, 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 $\mathbf{V} \twoheadrightarrow Q_{H,F}$ 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 $Q_{H,F}(D_1) = Q_{H,F}(D_2) = \mathit{adom(R)} \times \mathit{adom(R)}$. 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 $\mathbf{V} \twoheadrightarrow Q_{H,F}$. 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 view and query. Equality can be avoided as follows. A relation R is said to be *pseudo-monoidal* if there exists an equivalence relation $\simeq$ over the domain of R such that $\simeq$ is a congruence for R and $R/\simeq$ is monoidal. In other words, $R(x,y,z)$ and $R(x,y,z')$ may hold for distinct z and z', but z and z' are *equivalent* with respect to R, i.e. they cannot be distinguished using R. This can be enforced using the following equalities:

$\{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(z,u,v)\} = \{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(z',u,v)\}$,

$\{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(u,z,v)\} = \{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(u,z',v)\}$, and

$\{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(u,v,z)\} = \{(u,v,z,z') \mid \exists x,y\ R(x,y,z) \wedge R(x,y,z') \wedge R(u,v,z')\}$.

We 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 $\mathbf{V} \twoheadrightarrow Q_{H,F}$ iff H implies F over all finite pseudo-monoidal functions. The latter is undecidable, since implication over finite pseudo-monoidal functions is the same as implication over finite monoidal functions. This follows from the fact that a pseudo-monoidal function can be turned into a monoidal function by taking the quotient with $\simeq$ defined by $x \simeq y$ iff $\exists u, v\ R(u,v,x) \wedge R(u,v,y)$.

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 $p_2 = \exists x, y\ P(x,y) \wedge P(y,x)$. □

Conjunctive queries 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. We mention some partial results and examples.

We first show that determinacy is decidable in the case when the views are Boolean CQs. Furthermore, CQ is complete for rewritings of such queries and views.

Proposition 4.6 *It is decidable, given a set* $\mathbf{V}$ *of Boolean CQ views and a CQ query* Q*, whether* $\mathbf{V} \twoheadrightarrow Q$.

Proof: Let $\mathbf{V} = \{V_1, \ldots V_k\}$ and Q be a CQ. We may assume that different views have disjoint sets of variables. If Q is not Boolean then $\mathbf{V}$ cannot determine Q (otherwise genericity of $Q_{\mathbf{V}}$ would be violated for D's such that $Q(D) \neq \emptyset$). Thus, suppose Q is Boolean. For two CQs Q_1, Q_2, let us denote by $Q_1 \hookrightarrow Q_2$ the fact that there exists a homomorphism mapping $[Q_1]$ to $[Q_2]$. Let $I = \{i \mid V_i \hookrightarrow Q\}$. We claim that $\mathbf{V}$ determines Q iff $I \neq \emptyset$ and Q is equivalent to $\wedge_{i \in I} V_i$. The "if" part is obvious. Consider the "only if" part. Suppose $\mathbf{V}$ determines Q. First, note that $I \neq \emptyset$. Indeed, suppose $I = \emptyset$ and consider the instances $[Q]$ and $\emptyset$. Clearly, $\mathbf{V}([Q]) = \mathbf{V}(\emptyset) = 0$ but $Q([Q]) = 1$ and $Q(\emptyset) = 0$. Thus, $\mathbf{V}$ does not determine Q, which is a contradiction. It follows that $I \neq \emptyset$.

By definition of I, Q implies $\wedge_{i \in I} V_i$. For the converse it is enough to show that $[Q] \hookrightarrow \cup_{i \in I}[V_i]$. Consider the instance $D_1 = \cup_{i \in I}[V_i]$ and let D_2 be the complete structure formed with elements in $adom(D_1)$. Clearly, $\wedge_{i \in I} V_i(D_1) = \wedge_{i \in I} V_i(D_2) = 1$ and $Q(D_2) = 1$. Since $\mathbf{V}$ determines Q it follows that $Q(D_1) = Q(D_2) = 1$, so $[Q] \hookrightarrow \cup_{i \in I}[V_i]$. This completes the proof. □

A consequence of the above proof is that CQ is complete for rewritings of CQ queries in terms of Boolean CQ views.

We next consider another special case, when the set of views consists of a single unary view.

Proposition 4.7 *Given a unary CQ view* V *and a CQ query* Q*,* $\{V\} \twoheadrightarrow Q$ *iff* Q *is equivalent to* V^k*, where* k *is the number of free variables of* Q*. In particular, it is decidable if* $\{V\} \twoheadrightarrow Q$.

Proof: For simplicity we outline the proof for the case where the database is a single binary relation. The extension to arbitrary schemas is straightforward. Let $V(x)$ and $Q(x_1, \ldots, x_k)$ be CQs and suppose $\{V\} \twoheadrightarrow Q$. Let $Q_i = \pi_i(Q)$ (so Q_i is unary with free variable x_i). We first note that $Q(x_1, \ldots, x_k) \equiv Q_1(x_1) \wedge \ldots \wedge Q_k(x_k)$. Indeed, let I be the complete graph on nodes $\{x_1, \ldots, x_k\}$ and J be the graph consisting of the self loops $\{(x_i, x_i) \mid i \in [1,k]\}$. Clearly, $V(I) = V(J) = \{x_1, \ldots, x_k\}$. Therefore, $Q(I) = Q(J)$. But $(x_1, \ldots, x_k) \in Q(I)$, so it must be that $(x_1, \ldots, x_k) \in Q(J)$. Therefore there exists a homomorphism h from $[Q]$ to J such that $h(x_i) = x_i$, $i \in [1,n]$. Since x_i, x_j are not connected in J, they cannot be connected in $[Q]$. It easily follows that $Q \equiv Q_1(x_1) \wedge \ldots \wedge Q_k(x_k)$.

Finally, we show that each Q_i is equivalent to V. Fix $i \in [1, n]$. Since $\{V\} \twoheadrightarrow Q$, $\{V\} \twoheadrightarrow Q_i$. By Proposition 4.3 (i), $Q_i \subseteq V$. It remains to show that $V \subseteq Q_i$. Let $I = [V]$ and $J = \{(v, v) \mid v \in V([V])\}$. Clearly, $V(I) = V(J)$. It follows that $Q_i(I) = Q_i(J)$. Clearly, $x \in Q_i(J)$, since $x \in V([V])$ and $(x, x) \in J$. But then $x \in Q_i(I) = Q_i([V])$, so there exists a homomorphism from $[Q_i]$ to $[V]$ mapping x_i to x. It follows that $V \subseteq Q$, which completes the proof. □

Note that the above proof does not easily extend to the case of multiple unary views. Indeed, suppose we have two views V_1, V_2. The stumbling block in extending the proof is the construction of two instances I and J such that $V_1(I) = V_1(J)$ *and* $V_2(I) = V_2(J)$ allowing to conclude that there is a homomorphism from $[Q_i]$ to $[V]$ mapping x_i to x. The construction used above no longer works and cannot be easily extended.

Remark 4.8 *As an aside, we note that there is a more straightorward proof of Proposition 4.7. Suppose V ad Q are CQs, V is unary, and $\{V\} \twoheadrightarrow Q$. Let I be a database instance (a binary relation) and J be the complete graph with nodes $V(I)$. Clearly, $V(I) = V(J)$. Therefore, $Q(I) = Q(J)$. Since J is the complete graph on $V(I)$, $Q(J) = V(I)^k$. It follows that $Q(I) = V(I)^k$, which completes the proof. While this argument is more direct, it provides less insight than the above into the difficulty involved in extending the result to multiple unary views.*

Finally, we note that Proposition 4.7 can be easily extended to the case when Q is a union of conjunctive queries.

5 Finite Rewriting

In this section we consider the problem of rewriting a query Q in terms of a view $\mathbf{V}$, assuming that $\mathbf{V} \twoheadrightarrow Q$.

In the unrestricted case, we have seen that FO is complete for FO-to-FO rewritings. The proof relied on Craig's Interpolation theorem. As we have seen (Example 3.2), neither the proof technique nor the result carry over to the finite case. Indeed, we show the following.

Theorem 5.1 *If $\mathcal{R}$ is complete for FO-to-FO rewritings then $\mathcal{R}$ expresses all computable queries.*

Proof: Let M be an arbitrary Turing machine expressing a total, computable 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 $adom(G)$, and $\leq$ a total order on the nodes. We consider a standard encoding $enc_{\leq}(G)$ of G as a string in $\{0, 1\}^*$ of length $|adom(G)|^2$ whose $< i, j >$-th position in lexicographic order is 1 iff $< a_i, a_j > \in E$ where a_i and a_j have rank i resp. j with respect to the order $\leq$. M computes q iff M on input $enc_{\leq}(G)$ halts with output $enc_{\leq}(q(G))$, for every total order $\leq$ on $adom(G)$. 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 $D \supseteq adom(R_1)$ with $adom(R_1)$ as initial elements, and T represents a halting computation of M on input $enc_{\leq}(R_1)$, with output $enc_{\leq}(R_2)$. 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 $adom(G)$ as initial elements, and that T is a correct representation of a halting computation of M on input $enc_{\leq}(R_1)$, with output $enc_{\leq}(R_2)$. Let $\mathbf{V}$ be the view with $\sigma_{\mathbf{V}} = \{R_1\}$ defined by $Q_{R_1} = \varphi_M \wedge R_1(x, y)$, and Q be the query $\varphi_M \wedge R_2(x, y)$.

We claim that $\mathbf{V} \twoheadrightarrow Q$ 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. Therefore Q equals $R_2 = q(R_1)$. Thus, $Q = q \circ \mathbf{V}$. □

Existential FO views We have seen in Theorem 5.1 that for FO views and queries, the query answering problem is Turing complete. We now show that, when the views are defined in ∃FO, the complexity of query answering goes down to **NP** ∩ co-**NP**. By Fagin's Theorem (see [21]) this implies that ∃SO and ∀SO are complete for ∃FO-to-FO rewritings. We also show that this bound is tight using a logical characterization of **NP** ∩ co-**NP** by means of implicit definability [22, 17]. First, we show:

Theorem 5.2 *∃SO and ∀SO are both complete for ∃FO-to-FO rewritings.*

Proof: Let $\mathbf{V}$ be a view from $\mathcal{I}(\sigma)$ to $\mathcal{I}(\sigma_{\mathbf{V}})$ defined in ∃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 5.3 *Let $\mathbf{V}$ be defined in ∃FO and k be the maximum number of variables 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 $|adom(D)| \leq k|adom(S)|^k$.*

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 ∃FO, $\mathbf{V}$ is closed under extension, so $\mathbf{V}(D) \subseteq \mathbf{V}(D') = S$. Altogether $\mathbf{V}(D) = S$ and $adom(D) = A$ has size bounded by $k|adom(S)|^k$. □

Assume now that Q is in FO and that $\mathbf{V} \twoheadrightarrow Q$. Let $Q_{\mathbf{V}}$ be the mapping associating to every instance S in the image of $\mathbf{V}$ the corresponding value of Q. Recall that, by Proposition 4.3, $adom(Q_{\mathbf{V}}(S)) \subseteq adom(S)$. By Lemma 5.3 a non-deterministic polynomial algorithm for checking whether a tuple $\bar{c}$ over $adom(S)$ 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 5.3 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 **NP** while the second is in co-**NP**. By Fagin's theorem this implies that $\bar{c} \in Q_{\mathbf{V}}(S)$ can be expressed by both ∃SO and ∀SO formulas $\varphi(\bar{c})$. By universality of constants, it follows that there are ∃SO and ∀SO formulas $\varphi(\bar{x})$ defining $Q_{\mathbf{V}}$. □

We next show that Theorem 5.2 is tight, i.e. every rewriting language complete for ∃FO-to-FO rewritings must be able to express all properties in ∃SO ∩ ∀SO. In fact, the lower bound holds even for UCQ-to-FO rewritings.

Theorem 5.4 *Let τ be a schema. For every query q of instances over τ definable in $\exists SO \cap \forall SO$ there exists a set $\mathbf{V}$ of UCQ views and and an FO sentence Q such that $\mathbf{V} \twoheadrightarrow Q$ and $Q_{\mathbf{V}}$ defines q.*

Proof: We use a result of [22] (see also [17]) on the expressive power of implicit definability over finite structures. 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 the exists a schema $\tau' = \tau \cup \{T, \bar{S}\}$ where T is k-ary 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 $adom(D)$ such that $D \models \varphi(q(D), \bar{S})$ and (ii) for all relations T and $\bar{S}$ over $adom(R)$ we have $D \models \varphi(T, \bar{S})$ implies $T = Q(D)$.

The set of queries implicitly definable over τ is denoted by GIMP(τ). We use the following known result [22, 17]: GIMP(τ) consists of all queries whose data complexity is **NP** $\cap$ co-**NP**. In view of Fagin's Theorem, this yields:

Theorem 5.5 *[22, 17]* GIMP(τ) *consists of all queries over τ expressible in $\exists SO \cap \forall SO$.*

Thus, to establish Theorem 5.4 it is enough to show that every language complete for UCQ-to-FO rewritings must express every query in GIMP(τ).

Let τ be a schema and q a query in GIMP(τ). 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 $\bar{R}_\theta$ 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 $\bar{R}_\theta$ to be the complement of R_θ (for technical reasons, $\bar{R}_\theta$ is needed even when $\neg\theta$ 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. $\bar{R}_\theta = adom(D)^k - R_\theta$, $R_{\neg\theta} = \bar{R}_\theta$,
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 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 $\mathbf{V}_\tau$ of $\mathbf{V}$ simply returns $D(\tau)$. In particular, $\mathbf{V}_\tau$ provides the active domain A of D. A second set of views, $\mathbf{V}_\sigma$, allows verifying whether the relations in σ satisfy (1)-(3). Specifically, $\mathbf{V}_\sigma$ contains the following:

(i) Views allowing to check (1). To verify that $\bar{R}_\theta$ 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 $\mathbf{V} \twoheadrightarrow Q$. Consider $D \in \mathcal{I}(\tau'')$. As described above, the views $\mathbf{V}_\sigma$ 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 $\mathbf{V}_\tau$ provides $D(\tau)$, it follows that $\mathbf{V} \twoheadrightarrow Q$.

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 $adom(D(\tau))^k$ for the view relations in $\mathbf{V}_\sigma$ corresponding to the case when φ is satisfied by the pre-image of the view, as described in (i)-(iii), and with and with *true* for V_φ, corresponding to the case when the pre-image satisfies φ. This extension is trivial, and easily expressible from $D(\tau)$. □

Note that Theorem 5.4 requires only UCQ views and therefore Theorem 5.2 is also tight when views are restricted to UCQs. As an immediate consequence of Theorem 5.4 we have:

Corollary 5.6 *Datalog* $^\neg$ *and fixpoint logic*[2] *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 $\mathrm{CQ}^\neg$, where $\mathrm{CQ}^\neg$ is CQ extended with safe negation.

Proposition 5.7 *FO is not complete for* $\mathrm{CQ}^\neg$*-to-FO rewritings.*

Proof: We revisit Example 3.2. 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 $\mathrm{FO}(\sigma_<)$ 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 $R \in \sigma$ of arity k and distinct $i, j \in [1, k]$, one view $R(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_k) \wedge \neg(x_i < x_j) \wedge \neg(x_j < x_i)$
4. for each pair of relations $R_1, R_2 \in \sigma$ and appropriate i, j, one view $R_1(\ldots, x_i, \ldots) \wedge R_2(\ldots, y_j, \ldots) \wedge \neg(x_i < x_j) \wedge \neg(x_j < x_i)$
5. for each $R \in \sigma$, one view returning R.

We claim that $\mathbf{V} \twoheadrightarrow Q_\varphi$. First, note that ψ holds (i.e. $<$ is a strict total order on the domain) iff views (1)-(4) are empty. Indeed, this ensures antisymmetry (1), transitivity (2), and totality (3,4). If any of these views is not empty, then ψ is false so Q is false. Otherwise, Q 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, $\mathbf{V} \twoheadrightarrow Q$ and $Q_{\mathbf{V}}$ allows defining $\varphi(<)$ using just the relations in σ, which cannot be done in FO. □

[2] FO+LFP is FO extended with a least fixpoint operator, see [16].

UCQ views and queries We consider now the case when the views and queries are in UCQ. Unfortunately, UCQ is not complete for UCQ-to-UCQ rewritings, nor are much more powerful languages such as Datalog$^{\neq}$. 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 5.8 *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 the following three views:

$$\begin{aligned} Q_{V_1}(x): &\quad \exists u R(u) \wedge P(x) \\ Q_{V_2}(x): &\quad R(x) \vee P(x) \\ Q_{V_3}(x): &\quad R(x). \end{aligned}$$

Let $Q(x)$ be the query $P(x)$. It is easily seen that $\mathbf{V} \twoheadrightarrow Q$. 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, $\mathbf{V} \twoheadrightarrow Q$. 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 $Q(D_1) \not\subseteq Q(D_2)$. $\square$

As an immediate consequence of Proposition 5.8, we have:

Corollary 5.9 *Datalog$^{\neq}$ is not complete for UCQ-to-CQ rewritings, even if the database, views, and query are restricted to be unary.*

Proposition 5.8 provides a lower bound of sorts for any rewriting language complete for UCQ-to-UCQ rewritings. Another kind of lower bound on the power of rewriting languages complete for UCQ-to-UCQ rewritings follows from the undecidability of determinacy for UCQ views and queries (Theorem 5.2).

Proposition 5.10 *If $\mathcal{R}$ is complete for UCQ-to-UCQ rewritings, then it is undecidable whether a UCQ query Q can be rewritten in terms of a set of UCQ views $\mathbf{V}$ using a query in $\mathcal{R}$.*

Proposition 5.10 provides an immediate alternate proof that UCQ is not complete for UCQ-to-UCQ rewritings. Indeed, by Theorem 3.9 in [23], it is decidable if a UCQ query Q can be rewritten in terms of a set of UCQ views $\mathbf{V}$ using a UCQ.

Recall that, from Theorem 5.2, we know that $\exists$SO and $\forall$SO are both complete for UCQ-to-UCQ rewritings. It remains open if we can do better. In particular, we do not know if FO is complete for UCQ-to-UCQ rewritings.

CQ 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 ([23]). However, it remains open whether CQ is complete for CQ-to-CQ rewritings. Note that a positive answer immediately implies decidability of whether $\mathbf{V} \twoheadrightarrow Q$, by the above result of [23]. Recall from Section 4 that this problem is also open. When $\mathbf{V} \twoheadrightarrow Q$, one may also wonder what properties are required of $Q_{\mathbf{V}}$. For instance, can $Q_{\mathbf{V}}$ be non-monotonic, as in the case of UCQs? Surprisingly, this question turns out to be again as hard as the original. Indeed, we can show:

Theorem 5.11 *The following are equivalent:*

1. *CQ is complete for CQ-to-CQ rewritings;*

2. *for CQ views $\mathbf{V}$ and queries Q for which $\mathbf{V} \twoheadrightarrow Q$, the query $Q_{\mathbf{V}}$ is monotonic.*

Proof: $1 \to 2$. Obvious.

$2 \to 1$. Suppose 3 holds. Let $\mathbf{V}$ and $Q(\bar{x})$ be CQs such that $\mathbf{V} \twoheadrightarrow Q$ (finitely). Let $Q_{\mathbf{V}}$ be such that $Q = Q_{\mathbf{V}} \circ \mathbf{V}$. By 2, $Q_{\mathbf{V}}$ is monotonic. We wish to show that $Q_{\mathbf{V}}$ is in CQ.

We start by developing the notation and terminology needed for the proof. The *frozen body of* Q, denoted $[Q]$, is the instance over σ such that $(y_1, \ldots, y_k) \in R$ iff $R(y_1, \ldots, y_k)$ is an atom in Q. For a set $\mathbf{V}$ of CQs, $[\mathbf{V}]$ is the union of the $[P]$'s for all $P \in \mathbf{V}$. For a mapping α from variables to variables and constants, we denote by $\alpha([Q])$ the instance obtained by applying α to all variables in $[Q]$.

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

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

We will use the following lemma:

Lemma 5.12 *Let D be a database instance, $S = \mathbf{V}(D)$, and $D' = \mathbf{V}_{\emptyset}^{-1}(S)$. Then there exists a homomorphism h from D' to D which is the identity on $adom(D)$.*

Proof: Let $\bar{c}$ be a tuple occurring in $S(V)$ for $V \in \sigma_{\mathbf{V}}$ with associated query $Q_V(\bar{x})$. Let $h_{\bar{c}}$ be a homomorphism from $[Q_V]$ to D witnessing that $\bar{c} \in Q_V(D)$. By construction, $adom(D')$ contains $\alpha_{\bar{c}}([Q_V])$ for some $\alpha_{\bar{c}}$ mapping $\bar{x}$ to $\bar{c}$ and all other variables to new values. We now define h as follows. For $z \in adom(S) = adom(D) \cap adom(D')$, we set $h(z) = z$. For $z \notin adom(S)$ occurring in $\alpha_{\bar{c}}([Q_V])$, we set $h(z)$ to $h_{\bar{c}}(\alpha_{\bar{c}}^{-1}(z))$. It is now immediate, by construction of D', that h is a homomorphism from D' to D that is the identity on $adom(D)$. $\square$

We can now prove the following key observation:

Proposition 5.13 *Let $Q(\bar{x})$ be a CQ and $S = \mathbf{V}([Q])$. Let $Q_{\mathbf{V}}(\bar{x})$ be the CQ over $\sigma_{\mathbf{V}}$ with free variables $\bar{x}$ for which $[Q_{\mathbf{V}}] = S$. We have the following:*

(i) *$Q_{\mathbf{V}} \circ \mathbf{V}$ is equivalent to the CQ whose frozen body is $\mathbf{V}_{\emptyset}^{-1}(S)$;*

(ii) *$Q \subseteq Q_{\mathbf{V}} \circ \mathbf{V}$;*

(iii) *If $\bar{x} \in Q(\mathbf{V}_{\emptyset}^{-1}(S))$ then $Q = Q_{\mathbf{V}} \circ \mathbf{V}$. In particular, $\mathbf{V} \twoheadrightarrow Q$.*

Proof: Part (i) is obvious from the definition of $\mathbf{V}_\emptyset^{-1}(S)$, which is essentially an unfolding of $Q_{\mathbf{V}} \circ \mathbf{V}$. Part (ii) follows from (i) and Lemma 5.12 with $D = [Q]$. Consider (iii). If $\bar{x} \in Q(\mathbf{V}_\emptyset^{-1}(S))$ this means that there exists a homomorphism from $[Q]$ to $\mathbf{V}_\emptyset^{-1}(S)$ that fixes $\bar{x}$. By the Homomorphism Theorem for CQs [10], the query whose frozen body is $\mathbf{V}_\emptyset^{-1}(S)$ is included in Q. By (i), $Q_{\mathbf{V}} \circ \mathbf{V} \subseteq Q$. In conjunction with (ii), this implies $Q = Q_{\mathbf{V}} \circ \mathbf{V}$. □

Let $D = [Q]$, $S = \mathbf{V}(D)$, $D' = \mathbf{V}_\emptyset^{-1}(S)$ and, $S' = \mathbf{V}(D')$. By construction we have $\bar{x} \in Q(D)$. Therefore $\bar{x} \in Q_{\mathbf{V}}(S)$. It is also immediate to see that $S \subseteq S'$. Therefore, by monotonicity of $Q_{\mathbf{V}}$, we have $\bar{x} \in Q_{\mathbf{V}}(S')$. This implies $\bar{x} \in Q(D')$. By part (iii) of Proposition 5.13, $Q_{\mathbf{V}}$ is equivalent to the CQ defined there. □

One may wonder if there is a straighforward way to show monotonicity of $Q_{\mathbf{V}}$ for conjunctive queries and views. For example, one might attempt to show that for a conjunctive view V, if E_1 and E_2 are view instances such that $E_1 \subseteq E_2$, then there exist databases D_1 and D_2 such that $V(D_1) = E_1$, $V(D_2) = E_2$, and $D_1 \subseteq D_2$. This would solve the problem, since we would have $Q(D_1) \subseteq Q(D_2)$ so $Q_V(E_1) \subseteq Q_V(E_2)$ by monotonicity of Q. Unfortunately, this tempting approach fails, as shown next.

Example 5.14 Consider a database consisting of a binary relation R and a CQ view $V(x, y) = \exists z(R(x, z) \wedge R(z, y))$. Let $E_1 = \{(1, 2), (1, 1)\}$ and $E_2 = \{(1, 2), (1, 1), (3, 4), (4, 5), (4, 4)\}$. Note that $E_1 \subseteq E_2$ and E_1, E_2 are view instances. Indeed, $E_1 = V(I_1)$ and $E_2 = V(I_2)$, where $I_1 = E_1$ and $I_2 = \{(1, 4), (4, 1), (4, 2), (2, 5), (3, 1)\}$. Furthermore, a straightforward case analysis shows that the restriction of every instance in $V^{-1}(E_1)$ to $\{1, 2\}$ must equal I_1 and the restricion of every instance in $V^{-1}(E_2)$ to $\{1, 2, 3, 4, 5\}$ must equal I_2. Since $I_1 \not\subseteq I_2$, it follows that there are no database instances D_1, D_2 such that $D_1 \subseteq D_2$, $V(D_1) = E_1$, and $V(D_2) = E_2$

Thus, in terms of CQ-to-CQ rewriting, we are left with the upper bound provided by Theorem 5.2: both ∃SO and ∀SO are complete for CQ-to-CQ rewritings. It remains open whether a less powerful language is complete for such rewritings. In particular, we do not know if FO is a complete for CQ-to-CQ rewritings. Interestingly, if inequality is allowed in view definitions, we can show that CQ $^{\neq}$ is *not* a complete rewriting language. In fact, the following provides a lower bound similar to Proposition 5.8.

Proposition 5.15 *Any language complete for CQ $^{\neq}$-to-CQ rewritings must express non-monotonic queries. Moreover, this holds even if the views and query are restricted to be unary.*

Proof: Let $\sigma = \{R\}$, where R is binary. Consider the view $\mathbf{V}$ defined as follows: $Q_{V_1}(x) = \exists y\, R(x, y) \wedge R(y, x)$, $Q_{V_2}(x) = \exists y\, R(x, y) \wedge R(y, x) \wedge x \neq y$ and $Q_{V_3}(x) = \exists y\, R(x, x) \wedge R(x, y) \wedge R(y, x) \wedge x \neq y$. Consider the query $Q(x)$ defined by $R(x, x)$. It is easy to check that $\mathbf{V} \twoheadrightarrow Q$. 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. □

From Proposition 5.15 we immediately have:

Corollary 5.16 *Datalog $^{\neq}$ is not complete for CQ $^{\neq}$-to-CQ rewritings, even if the views and query are restricted to be unary.*

Proposition 5.15 follows from the fact that for $\mathbf{V} \in \mathrm{CQ}^{\neq}$ and $Q \in CQ$ such that $\mathbf{V} \twoheadrightarrow Q$, $Q_{\mathbf{V}}$ may be non-monotonic. Similarly, Corollary 5.9 follows from the fact that $\mathbf{V} \in$ UCQ and $Q \in (U)CQ$ such that $\mathbf{V} \twoheadrightarrow Q$, $Q_{\mathbf{V}}$ may be non-monotonic. Suppose we are given $\mathbf{V}$ and Q in the above languages for which $Q_{\mathbf{V}}$ happens to be monotonic. Can we do better in this case? Indeed, we can show the following.

Proposition 5.17 *(i) Let $\mathbf{V} \in CQ^{\neq}$ and $Q \in (U)CQ$. If $\mathbf{V} \twoheadrightarrow Q$ 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 $\mathbf{V} \twoheadrightarrow Q$ and $Q_{\mathbf{V}}$ is monotonic, then there exists $R \in UCQ$ such that $Q \Rightarrow_{\mathbf{V}} R$.*

Proof: (i) Let $\mathbf{V}$ and Q be as in the statement. As noted in the introduction, since $Q_{\mathbf{V}}$ is monotonic, $Q_{\mathbf{V}}(E) = cert_Q(E)$ for every view instance E, where $cert_Q(E)$ is defined as in [1] for sound views (open world assumption), see "related work" in the Introduction. By Theorem 4.3 in [1], we can assume that $\mathbf{V}$ contains no inequalities. By Theorem 4.2 in [1], there exists a Datalog program P over $\mathbf{V}$ that computes $cert_Q$. Since $\mathbf{V} \twoheadrightarrow Q$ and $Q_{\mathbf{V}}$ is monotonic, $Q(D) = cert_Q(\mathbf{V}(D))$ for every input database D. 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 (3) $\rightarrow$ (1) in Theorem 5.11.
(ii) The proof is an extension of that of (3) $\rightarrow$ (1) in Theorem 5.11. 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 Theorem 5.11. Thus, for each Q_j we denote by $[Q_j]$ the frozen body of Q_j.

Consider j, $1 \leq j \leq m$. Let $S_j = V([Q_j])$. Extending the ideas of Proposition 5.13 we set $Q'(\bar{q})$ to 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 $Q_V = Q'$.

Consider $Q_V \subseteq Q'$. It is sufficient to show that $Q_j \subseteq \exists \bar{y}(\bigwedge_{\bar{t} \in S_j} V(\bar{t})) \circ V$ for each j. Let $\mathcal{F}_j$ be the set of mappings σ from S_j to $\{1, \ldots, n\}$ such that $\bar{t} \in V_{\sigma(\bar{t})}([Q_j])$ for each $\bar{t} \in S_j$. Intuitively $\mathcal{F}_j$ says all the possible ways to chase S_j. For each $\sigma \in \mathcal{F}_j$ let $V_\sigma(\bar{q})$ be the CQ whose free variables are $\bar{q}$ and whose frozen body is $[V_\sigma] = \bigcup_{\bar{t} \in S_j} (V_{\sigma(\bar{t})})_{\emptyset}^{-1}(\bar{t})$. Similarly to part (i) of Proposition 5.13, it is easy to verify that, V_σ is equivalent to $\exists \bar{y}(\bigwedge_{\bar{t} \in S_j} V_{\sigma(\bar{t})}(\bar{t})) \circ V$ and therefore $\exists \bar{y}(\bigwedge_{\bar{t} \in S_j} V(\bar{t})) \circ V$ is equivalent to $\bigvee_{\sigma \in \mathcal{F}_j} V_\sigma$. By an argument similar to (ii) of Proposition 5.13, for each $\sigma \in \mathcal{F}_j$ we construct a homomorphism from V_σ to Q_j and therefore $Q_j \subseteq V_\sigma$. [[[*luc: this is strange it implies* <==

that Q_j is actually included in $\bigwedge_{\sigma\in\mathcal{F}_j} V_\sigma$]]] Therefore $Q_j \subseteq \bigvee_{\sigma\in\mathcal{F}_j} V_\sigma$, and $Q_V \subseteq Q'$.

Consider $Q' \subseteq Q_V$. Fix j. As noted above, $(\bigwedge_{\bar{t}\in S_j} V(\bar{t}))\circ V$ is equivalent to $\bigvee_{\sigma\in\mathcal{F}_j} V_\sigma$. Now consider $\sigma \in \mathcal{F}_j$ and $[V_\sigma]$. By construction, $S_j \subseteq V([V_\sigma])$. By monotonicity of Q_V, and since $S_j = V([Q_j])$, it follows that $Q([Q_j]) \subseteq Q([V_\sigma])$. As obviously $\bar{q} \in Q([Q_j])$, then $\bar{q} \in Q(V_\sigma)$, so $\bar{q} \in Q_i([V_\sigma])$ for some i. Thus, $V_\sigma \subseteq Q_i \subseteq Q$. This implies that $\bigvee_{\sigma\in\mathcal{F}_j} V_\sigma \subseteq Q$, and completes the proof of (ii). $\square$

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, in both the unrestricted and finite cases. While the questions were settled for many languages, several interesting problems remain open. First and foremost, little is known about CQ in the finite case: it remains open whether determinacy is decidable, or whether CQ is complete for CQ-to-CQ rewritings. We have made some inroads by settling some special cases, and showing the equivalence of several open questions about CQs.

For UCQ, we know that no monotonic language is complete for UCQ-to-CQ rewritings (and similarly for CQ $^\neg$-to-CQ rewritings) and we know that $\exists$SO and $\forall$SO are complete for such rewritings. It is of interest to know if there are languages less powerful than $\exists$SO and $\forall$SO that remain complete for UCQ-to-UCQ rewritings, such as FO or FO+LFP. Similarly, short of settling whether CQ is complete for CQ-to-CQ rewritings, it would be of interest to find languages less powerful than $\exists$SO and $\forall$SO that are complete for CQ-to-CQ rewritings.

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 $\mathbf{V} \twoheadrightarrow Q$, produces $R \in \mathcal{R}$ such that $Q \Rightarrow_{\mathbf{V}} R$?

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 [9].

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. PODS 1998, 254-263.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley, 1995.

[3] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. PODS 2002, 209-220.

[4] S. Agrawal, S. Chaudhuri, and V. Narasayya. Automated selection of materialized views and indexes in Microsoft SQL Server. VLDB 2000, 496-505.

[5] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.* 6(4): 557-575, 1981.

[6] Egon Börger, Erich Gräel, and Yuri Gurevich. *The Classical Decision Problem.* Springer, 1997.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Rewriting of regular expressions and regular path queries. PODS 1999, 194-204.

[8] 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.

[9] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Lossless regular views. PODS 2002, 247-258.

[10] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. STOC 1977, 77-90.

[11] C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, 1977.

[12] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. ICDT 1997, 56-70.

[13] S. Dar, M.J. Franklin, B. Jonsson, D. Srivastava and M. Tan. Semantic data caching and replacement. VLDB 1996, 330-341.

[14] A. Deutsch, L. Popa, and V. Tannen. Physical data independence, constraints and optimization with universal plans. VLDB 1999, 459-470.

[15] O. Duschka and M.R. Genesereth. Answering recursive queries using views. PODS 1997, 109-116.

[16] H-D Ebbinghaus and J. Flum. *Finite Model Theory.* Springer, 1995.

[17] S. Grumbach, Z. Lacroix, and S. Lindell. Generalized implicit definitions on finite structures. In *Computer Science Logic*, 1995, 252-265.

[18] S. Grumbach, M. Rafanelli, and L. Tininini. Querying aggregate data. PODS 1999, 174-184.

[19] S. Grumbach and L. Tininini. On the content of materialized aggregate views. PODS 2000, 47-57.

[20] Yuri Gurevich. The word problem for some classes of semigroups. *Algebra and Logic*, 5(4):25–35, 1966. (Russian).

[21] Leonid Libkin. *Elements of finite model theory.* Springer, 2004.

[22] S. Lindell. *The Logical Complexity of Queries on Unordered Graphs.* PhD thesis, University of California at Los Angeles, 1987.

[23] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. PODS 1995, 95–104.

[24] Emil L. Post. Recursive unsolvability of a problem of Thue. *Journal on Symbolic Logic*, 12(1):1–11, 1947.

[25] A. Rajaraman, Y. Sagiv, and J.D. Ullman. Answering queries using templates with binding patterns. PODS 1995, 105-112.

[26] O.G. Tsatalos, M.H. Solomon and Y.E. Ioannidis. Describing and using query capabilities of heterogeneous sources. VLDB 1997, 256-265.

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

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