Information Systems 37 (2012) 1–12

Contents lists available at SciVerse ScienceDirect

Information Systems

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

View determinacy for preserving selected information in data transformations

Wenfei Fan [a], Floris Geerts [a], Lixiao Zheng [b,*]

[a] University of Edinburgh, United Kingdom
[b] Institute of Software, Chinese Academy of Sciences, P.O. Box 8718, Beijing 100190, China

ARTICLE INFO

Article history:
Received 18 April 2011
Accepted 9 September 2011
Recommended by: J. Van den Bussche
Available online 16 September 2011

Keywords:
Information preservation
Queries
Views
Rewriting
View determinacy

ABSTRACT

When transforming data one often wants certain information in the data source to be preserved, *i.e.*,we identify parts of the source data and require these parts to be transformed without loss of information. We characterize the preservation of selected information in terms of the notions of invertibility and query preservation, in a setting when transformations are specified as a view $\mathbf{V}$ (a set of queries), and source information is selected by a query Q. We investigate the problem for determining whether transformations $\mathbf{V}$ preserve the information selected by Q. (1) We show that the notion of invertibility coincides with view determinacy studied for query rewriting. (2) We establish the undecidability of the problem when either Q or $\mathbf{V}$ is in DATALOG or first-order logic, for invertibility and query preservation. (3) When Q and $\mathbf{V}$ are conjunctive queries (CQ), the problem is as hard as view determinacy for CQ queries and CQ views, an open problem. Nevertheless, we provide complexity bounds of the problem, either in PTIME or NP-complete, when $\mathbf{V}$ ranges over subclasses of CQ (*i.e.*, SP, SC, PC), and when Q is assumed to be a minimal CQ query or not. (4) We show that CQ is complete for $\mathcal{L}$-to-CQ rewriting when $\mathcal{L}$ is SP, SC or PC, *i.e.*,every CQ query can be rewritten in terms of SP, SC or PC views using a query in CQ.

1. Introduction

When transforming data from a data source to a target database in practice, we often want to preserve certain information in the data source. That is, we identify certain parts of the source data and require the parts to be transformed *without loss of information*. For example, to migrate a customer database D from one platform to another, we may want the transformation to ensure that the entire set of customers in D can be retrieved from the target database. When exchanging the data with a database of domestic customers, on the other hand, we may want the transformation to warrant that all conjunctive queries about domestic customers in D can still be answered by using conjunctive queries posed on the target data.

The practical need gives rise to the following questions. How should we model the preservation of selected information in data transformations? Can we effectively determine whether a given transformation preserves the information selected?

To answer these questions, this paper introduces a characterization of selected information preservation, investigates its fundamental problems and establishes their complexity bounds.

Information preservation: We propose two criteria to specify the preservation of selected information. Consider the setting in which data transformations are specified in terms of a view $\mathbf{V}$ (a set of queries) from source to target, and the selected information is identified by a query Q defined on the data source.

We say that $\mathbf{V}$ is *invertible* relative to Q if there exists a query Q^{-1} such that for every source database D,

* Corresponding author. Tel.: +86 10 62661600; fax: +86 10 62661627.
E-mail addresses: wenfei@inf.ed.ac.uk (W. Fan), fgeerts@inf.ed.ac.uk (F. Geerts), zhenglx@ios.ac.cn (L. Zheng).

0306-4379/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2011.09.001

$Q(D) = Q^{-1}(\mathbf{V}(D))$. Intuitively, it says that source data $Q(D)$ selected by Q can be effectively reconstructed from the target data $\mathbf{V}(D)$. In other words, when $Q(D)$ is concerned, the transformation $\mathbf{V}$ does not lose any information.

Consider a query language $\mathcal{L}_q$. We say that $\mathbf{V}$ is *query preserving* relative to Q and $\mathcal{L}_q$ if there exists a computable function $F : \mathcal{L}_q \rightarrow \mathcal{L}_q$ such that for any query $Q' \in \mathcal{L}_q$ and source database D, $Q'(Q(D)) = F(Q')(\mathbf{V}(D))$. Intuitively, for any Q' in $\mathcal{L}_q$ that can be answered in $Q(D)$, the same answer can also be found in the target $\mathbf{V}(D)$ by using a query in the same $\mathcal{L}_q$; *i.e.,*when queries in someone's favorite languages are concerned, no information in $Q(D)$ is lost in the transformation.

Observe that when Q is the identity query, $Q(D)$ selects the entire data source D, and invertibility and query preservation aim to preserve the information of the entire D.

We investigate the connection between invertibility and query preservation. These two notions are not equivalent. The former asks for the ability to restore the selected source data $Q(D)$, while the latter concerns the information in $Q(D)$ that can be retrieved using queries in a particular language $\mathcal{L}_q$. We show that when $\mathcal{L}_q$ contains the identity query as found in most sensible relational query languages, query preservation is a stronger notion. Indeed, if $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$, then $\mathbf{V}$ is invertible relative to Q. In contrast, there exist $\mathbf{V}$,Q and $\mathcal{L}_q$ such that $\mathbf{V}$ is invertible relative to Q but $\mathbf{V}$ is not query preserving relative to Q and $\mathcal{L}_q$. In addition, we identify sufficient conditions for the two notions to coincide.

Connection with view determinacy: There is also an intimate connection between invertibility and the notion of view determinacy introduced in [1]. A view $\mathbf{V}$ is said to *determine* a query Q iff for all databases D_1 and D_2, if $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ then $Q(D_1) = Q(D_2)$. That is, $\mathbf{V}$ provides enough information to uniquely determine the answer to Q. The notion of view determinacy has proved useful in a variety of applications such as query rewriting using views, semantic caching, security and privacy [1–7].

We show that invertibility and view determinacy coincide: for any view $\mathbf{V}$ and query Q, $\mathbf{V}$ is invertible relative to Q iff $\mathbf{V}$ determines Q. Among other things, this tells us that the study of view determinacy also finds applications in preserving selected information in data transformations, and vice versa.

Complexity results: We study two problems for determining whether a transformation preserves selected information.

The *invertability problem* is to decide, given a view $\mathbf{V}$ and a query Q, whether $\mathbf{V}$ is invertible relative to Q.

The *query preservation problem* is to determine, given $\mathbf{V}$, Q and a query language $\mathcal{L}_q$, whether $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$.

We parametrize the problems with various $\mathcal{L}_s$ and $\mathcal{L}_v$, the query languages in which selection queries Q are expressed and in which views $\mathbf{V}$ are defined, respectively. We consider the following $\mathcal{L}_s$ and $\mathcal{L}_v$: DATALOG, first-order queries (FO), and conjunctive queries (CQ). We also consider SP, PC and SC, subclasses of CQ denoted by listing the operators supported (selection, projection and Cartesian product).

We show that both problems are undecidable when one of $\mathcal{L}_s$ and $\mathcal{L}_v$ is CQ while the other is either DATALOG or FO. These results carry over to the problem for deciding whether $\mathbf{V}$ determines Q. While it is known that the view determinacy problem is undecidable when $\mathbf{V}$ or Q is in FO [5], the results on DATALOG are new additions to the study of view determinacy.

When both $\mathcal{L}_s$ and $\mathcal{L}_v$ are CQ, the invertibility problem is as hard as the view determinacy problem when $\mathbf{V}$ and Q are in CQ, which remains open [5]. We focus on special cases when Q is a CQ query and views $\mathbf{V}$ are defined in SP, SC or PC. We show that the invertibility problem is in PTIME for PC views, but it becomes NP-complete for SP and SC views. Moreover, we show that the problem is also in PTIME for SP views when Q is a minimal CQ query (see, *e.g.,* [8] for minimal CQ queries). These complexity bounds remain intact for their view determinacy counterparts. In addition, we show that these results carry over to the query preservation problem when $\mathcal{L}_q$ is CQ.

Complete rewriting: Another notion introduced in [1] concerns the completeness of a rewriting language. In a query language $\mathcal{L}$, a query Q can be *rewritten* using a view $\mathbf{V}$ iff there exists a query Q^{-1} in $\mathcal{L}$ such that $Q(D) = Q^{-1}(\mathbf{V}(D))$ for all databases D [1]. That is, the inverse Q^{-1} of Q is definable in $\mathcal{L}$. Clearly, if Q can be rewritten using a view $\mathbf{V}$ with a query Q^{-1} in a language $\mathcal{L}$, then $\mathbf{V}$ determines Q, while the converse may not be true. The language $\mathcal{L}$ is said to be *complete* for $\mathcal{L}_v$-to-$\mathcal{L}_s$ rewritings if $\mathcal{L}$ can be used to rewrite a query Q in $\mathcal{L}_s$ using $\mathbf{V}$ in $\mathcal{L}_v$ whenever $\mathbf{V}$ determines Q. That is, $\mathcal{L}$ is expressive enough to capture rewritings of $\mathcal{L}_s$ queries using $\mathcal{L}_v$ views as long as the views determine those queries.

It is known that CQ is not complete for CQ-to-CQ rewritings [1]. Nevertheless, we show that CQ is complete for $\mathcal{L}$-to-CQ rewritings when $\mathcal{L}$ ranges over SP, PC and SC.

This work is a first step towards characterizing the preservation of selected information in data transformations. Our results reveal the connection and differences between the two notions for information preservation, namely, invertibility and query preservation. In addition, the complexity results of the paper are of interest to both the study of data transformations and research on query rewriting using views. A variety of techniques are used to prove the results, including characterizations of CQ subclasses, reductions and constructive proofs with algorithms.

Related work: Closest to this work is the study of view determinacy, introduced in [1]. A number of results have been developed for the view determinacy problem and the completeness of rewriting languages, briefly summarized as follows [4,5,1]. (1) The view determinacy problem is undecidable when either queries or views are in FO. Furthermore, FO is not complete for FO-to-FO rewritings. In fact, it has been shown that any language that is complete for FO-to-FO rewritings must be Turing-complete. (2) The problem remains undecidable for UCQ queries and UCQ views, and moreover, UCQ is not complete for UCQ-to-UCQ rewritings. Indeed, no monotonic language is complete for CQ-to-CQ rewriting. (3) It remains unknown whether the view determinacy problem is decidable when the view and queries are in CQ [5].

In light of the practical interests in CQ queries, view determinacy has been studied for a variety of special classes of CQ queries and views in [4,5,1]. It has been shown there that the problem is decidable and that CQ is complete for rewritings in the following cases: (1) arbitrary CQ queries and Boolean CQ views; (2) arbitrary CQ queries and monadic CQ views (*i.e.,* CQ views with only one free variable); and (3) arbitrary CQ queries and a single path CQ view, which is defined over a single binary relation and has the form $Q(x,y) = \exists x_1, \ldots, x_k(R(x,x_1) \wedge R(x_1,x_2) \wedge \cdots \wedge R(x_{k-1},x_k) \wedge R(x_k,y))$.

Special cases of the view determinacy problem for CQ have also been studied in [2,3,6,7]. (1) The packed fragment of FO (PFO) was considered in [3], which is a generalization of the guarded fragment of FO. It was shown that PFO is complete for PFO-to-PFO rewritings, and the determinacy problem for PFO queries and PFO views is decidable in 2EXPTIME. Moreover, for the packed fragment of conjunctive queries (PCQ), PCQ is complete for PCQ-to-PCQ rewritings and thus the determinacy problem is decidable. These results also extend to unions of PCQs. (2) Chain CQ queries, denoted as CQ_{chain}, were studied in [2], which extend path CQ queries by allowing multiple binary relations. It was shown there that determinacy is decidable for chain queries and chain views, and that FO is complete for CQ_{chain}-to-CQ_{chain} rewritings. (3) These results were extended in [6] to connected graph CQ queries, denoted by CQ_{cgraph}, which are binary CQ queries whose body, if viewed as an undirected graph, is connected. It was reported there that FO is complete for CQ_{chain}-to-CQ_{cgraph} rewritings. (4) Ref. [7] studied CQ queries that are defined over unary database schemas, in which each relation has only one attribute. It was shown that for this class of queries and views, determinacy is decidable in PTIME and CQ is complete for rewritings. Nevertheless, none of these results transfers to the cases we consider.

As observed in [5], view determinacy (invertibility) is equivalent to the notion of lossless views under the exact view assumption, which has been studied for regular path queries [9,10]. Also related is the large amount of work on equivalent rewritings of queries using views (*e.g.,* [11,12]). It was shown that it is NP-complete to decide whether a given CQ query has an equivalent rewriting using a given set of CQ views [11], and several of its special PTIME cases were identified in [12].

As we shall show shortly, invertibility and view determinacy are equivalent. Therefore, all of our results on invertibility carry over to view determinacy. In particular, this work shows that the view determinacy problem is undecidable when either queries or views are in DATALOG. In addition, we provide the complexity of the problem for queries in CQ and for views in SP, PC or SC, either NP-complete or in PTIME. We also show that CQ is complete for $\mathcal{L}$-to-CQ rewritings when $\mathcal{L}$ ranges over subclasses SP, PC and SC of CQ. On the other hand, previous results on view determinacy also transfer to invertibility. In addition, when invertibility (view determinacy) and query preservation coincide as we shall elaborate, prior results on view determinacy also remain intact on query preservation, and vice versa.

The notions of invertibility and query preservation are also related to the notions of dominance and calculus dominance, which were proposed in [13] to specify relative information capacity, and were studied for data integration [14–16]. A schema S is said to *dominate* another schema T if there exist schema mappings V and V^{-1} from S to T and from T to S, respectively, such that for any source instance D of S, $D = V^{-1}(V(D))$. Schema S *calculously dominates* T if S dominates T with (V,V^{-1}) and moreover, both V and V^{-1} are expressible in relational calculus. Clearly, dominance is a special case of invertibility when selection query Q is the identity query, and calculus dominance is the special case when Q is the identity and views are in FO. These notions were also considered in the XML settings in [17,18]. No previous results on (calculus) dominance can carry over to the cases studied in this work.

Organization: Section 2 presents the notions of query preservation, invertability and view determinism, and investigates their connections. Section 3 states the decision problems studied in this paper. Section 4 provides the undecidability results for DATALOG and FO, followed by the decidable cases for subclasses of CQ in Section 5. Finally, Section 6 summarizes the main results of the paper and identifies open questions.

2. Selected information preservation

In this section, we first introduce the notions of query preservation, invertability and view determinism. We then investigate the connections between these concepts.

A database schema $\mathcal{R} = (R_1, \ldots, R_k)$ consists of a finite set of relation symbols R_i, each of which is associated with an arity $n_i \geq 0$. Let **dom** be an infinite set of values. A (database) instance $D = (I_1, \ldots, I_k)$ of $\mathcal{R}$ associates with each symbol R_i a relation I_i consisting of n_i-ary tuples over **dom**. In this paper, we only consider finite instances. We denote by $\mathcal{I}(\mathcal{R})$ the set of all instances of $\mathcal{R}$ that take values from **dom**. The active domain of a relation I, denoted by $\mathsf{adom}(I)$, is the set of values in **dom** that occur in I. Similarly, for $D = (I_1, \ldots, I_k)$ we define $\mathsf{adom}(D)$ as the union of $\mathsf{adom}(I_i)$ for $i \in [1,k]$.

A query Q over $\mathcal{R}$ is defined as a computable (generic) mapping from $\mathcal{I}(\mathcal{R})$ to $\mathcal{I}(R)$, for some output relation R. Let $\mathcal{R} = (R_1, \ldots, R_k)$ and $\mathcal{V} = (V_1, \ldots, V_\ell)$ be two database schemas. A view $\mathbf{V}$ from $\mathcal{R}$ to $\mathcal{V}$ is a set of queries Q_i from $\mathcal{I}(\mathcal{R})$ to $\mathcal{I}(V_i)$, one for each $i \in [1,\ell]$. For a query language $\mathcal{L}_v$, we say that $\mathbf{V}$ is *a view in* $\mathcal{L}_v$ if Q_i is in $\mathcal{L}_v$ for each $i \in [1,\ell]$. We refer to $\mathcal{R}$ and $\mathcal{V}$ as the input and output schema of $\mathbf{V}$, respectively.

2.1. Invertibility and query preservation

Let Q be a query over source schema $\mathcal{R}$ and let $\mathbf{V}$ be a view from $\mathcal{R}$ to $\mathcal{V}$. We say that $\mathbf{V}$ is *invertible* relative to Q if there exists a query Q^{-1} over $\mathcal{V}$ such that for every instance D of $\mathcal{R}$, $Q(D) = Q^{-1}(\mathbf{V}(D))$. Intuitively, invertibility says that the selected part of source data, identified by Q, can be recovered from the view. It does not say, however, whether the inverse Q^{-1} belongs to a certain query language or whether the inverse can be computed efficiently.

Let $\mathcal{L}_q$ be a query language. We say that a view $\mathbf{V}$ is *query preserving* relative to Q and $\mathcal{L}_q$ if there exists a computable function $F : \mathcal{L}_q \rightarrow \mathcal{L}_q$ such that for any query

$Q' \in \mathcal{L}_q$ and any instance D of $\mathcal{R}$, $Q'(Q(D)) = F(Q')(\mathbf{V}(D))$. Intuitively, any query (in a specific query language $\mathcal{L}_q$) imposed on the selected part of source data can be effectively answered using the view.

We say that a view $\mathbf{V}$ is *information preserving* relative to a query Q and a query language $\mathcal{L}_q$ if $\mathbf{V}$ is both invertible and query preserving relative to Q and $\mathcal{L}_q$.

We next reveal the connection and differences between invertibility and query preservation. We start with sufficient conditions for the two notions to be equivalent, which extend an observation of [18] for special cases of these two notions in the context of semi-structured data and query languages.

Proposition 1. *Let Q be a query, $\mathbf{V}$ a view and $\mathcal{L}_q$ a query language.*

- *If $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$, and the identity query* **id** *is expressible in $\mathcal{L}_q$, then $\mathbf{V}$ is invertible relative to Q, and moreover, Q^{-1} is a query in $\mathcal{L}_q$ as well.*
- *If $\mathbf{V}$ is invertible relative to Q, the inverse Q^{-1} is expressible in $\mathcal{L}_q$, and $\mathcal{L}_q$ is closed under composition, then $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$.*

Here a query language $\mathcal{L}_q$ is closed under composition if for any Q_1, Q_2 in $\mathcal{L}_q$, $Q_1 \circ Q_2$ (if defined) is also in $\mathcal{L}_q$.

Proof. Suppose that $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$. Then there exists a computable function $F : \mathcal{L}_q \to \mathcal{L}_q$ such that for any query $Q' \in \mathcal{L}_q$ and any instance D, $Q'(Q(D)) = F(Q')(\mathbf{V}(D))$. By assumption, **id** is expressible in $\mathcal{L}_q$ and thus $Q(D) = \mathbf{id}(Q(D)) = F(\mathbf{id})(\mathbf{V}(D))$ for any instance D. That is, $Q^{-1} = F(\mathbf{id})$. Hence $\mathbf{V}$ is invertible relative to Q and moreover, the inverse Q^{-1} is a query in $\mathcal{L}_q$.

Suppose that $\mathbf{V}$ is invertible relative to Q and the inverse Q^{-1} is in $\mathcal{L}_q$. By assumption, $\mathcal{L}_q$ is closed under composition and therefore, we can define a function $F : \mathcal{L}_q \to \mathcal{L}_q$ as $F(Q') = Q' \circ Q^{-1}$ for any $Q' \in \mathcal{L}_q$. Clearly, for any $Q' \in \mathcal{L}_q$ and any instance D, $Q'(Q(D)) = Q' \circ Q^{-1}(\mathbf{V}(D)) = F(Q')(\mathbf{V}(D))$. That is, $\mathbf{V}$ is query preserving relative to Q and $\mathcal{L}_q$. $\square$

Observe that **id** is definable in all commonly used relational query languages. In the sequel we consider *w.l.o.g.* only query languages in which **id** is definable. Hence, the notion of query preservation is generally stronger than invertibility. This is verified by the separation result below, which we shall prove shortly.

Proposition 2. *There exist a* CQ *query Q and a view $\mathbf{V}$ in* CQ *such that* (1) $\mathbf{V}$ *is invertible relative to Q, but* (2) $\mathbf{V}$ *is not query preserving relative to Q and* CQ.

2.2. View determinacy

It turns out that the notion of invertibility coincides with the notion of *view determinacy* [5], which we recall next. Let Q be a query over source schema $\mathcal{R}$ and let $\mathbf{V}$ be a view from $\mathcal{R}$ to $\mathcal{V}$. A view $\mathbf{V}$ *determines* Q, denoted by $\mathbf{V} \twoheadrightarrow Q$, iff for all instances D_1, D_2 of $\mathcal{R}$, if $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ then $Q(D_1) = Q(D_2)$.

Lemma 1. *Let Q be a query and $\mathbf{V}$ a view. Then $\mathbf{V}$ is invertible relative to Q iff $\mathbf{V}$ determines Q.*

Proof. Suppose that $\mathbf{V}$ is invertible relative to Q. Then for any pair of instances D_1 and D_2 we have that $Q(D_1) = Q^{-1}(\mathbf{V}(D_1))$ and $Q(D_2) = Q^{-1}(\mathbf{V}(D_2))$. Thus if $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ then clearly $Q(D_1) = Q(D_2)$, and hence $\mathbf{V} \twoheadrightarrow Q$.

Conversely, suppose that $\mathbf{V} \twoheadrightarrow Q$. Let σ be the mapping that associates $\mathbf{V}(D)$ with the corresponding value of $Q(D)$, for every instance D. It is easily verified (see e.g., [1]) that σ is generic, computable and furthermore can be taken as the inverse Q^{-1}. Hence, $\mathbf{V}$ is indeed invertible relative to Q. $\square$

The completeness of rewriting languages has also been studied in [5]. We say that a query Q can be *rewritten* using $\mathbf{V}$ in a language $\mathcal{L}$ iff there exists some query $Q^{-1} \in \mathcal{L}$ over the schema $\mathcal{V}$ such that $Q(D) = Q^{-1}(\mathbf{V}(D))$ for all instances D of $\mathcal{R}$. We denote this by $Q \Rightarrow_{\mathbf{V}} Q^{-1}$. Observe that if $Q \Rightarrow_{\mathbf{V}} Q^{-1}$ for a query Q^{-1} in some query language $\mathcal{L}$, then obviously $\mathbf{V} \twoheadrightarrow Q$. The converse is, however, generally not true.

Given a view language $\mathcal{L}_v$ and a query language $\mathcal{L}_s$, we say that a query language $\mathcal{L}$ is a *complete rewriting language for $\mathcal{L}_v$-to-$\mathcal{L}_s$ rewritings* if for all query $Q \in \mathcal{L}_s$ and view $\mathbf{V}$ in $\mathcal{L}_v$, $\mathcal{L}$ can be used to rewrite Q using $\mathbf{V}$ whenever $\mathbf{V} \twoheadrightarrow Q$.

It is known that CQ is not complete for CQ-to-CQ rewritings [5]. Capitalizing on this, we give a proof of Proposition 2.

Proof of Proposition 2. It is known that there exist a CQ query Q and a CQ view $\mathbf{V}$ such that $\mathbf{V} \twoheadrightarrow Q$, but the inverse Q^{-1} is not definable in CQ. Such concrete examples can be found in [2,5]. Let Q and $\mathbf{V}$ be such a pair. By Lemma 1, $\mathbf{V}$ is invertible relative to Q. Hence to prove Proposition 2, it suffices to show that $\mathbf{V}$ is *not* query preserving relative to Q and CQ.

Assume by contradiction that $\mathbf{V}$ is query preserving relative to Q and CQ. Then there exists a computable function $F : \mathrm{CQ} \to \mathrm{CQ}$ such that for any query $Q' \in \mathrm{CQ}$ and any instance D, $Q'(Q(D)) = F(Q')(\mathbf{V}(D))$. Let Q' be **id**, then $Q(D) = F(\mathbf{id})(\mathbf{V}(D))$, *i.e.*, $Q \Rightarrow_{\mathbf{V}} F(\mathbf{id})$, and $F(\mathbf{id})$ is a CQ query. This contradicts the fact that Q^{-1} is not definable in CQ. $\square$

3. Problem statements

We investigate the following decision problems. Let $\mathcal{L}_s$, $\mathcal{L}_v$ and $\mathcal{L}_q$ be query languages. The first problem is referred to as *the invertibility problem*, stated as follows.

PROBLEM:	VDet($\mathcal{L}_s$,$\mathcal{L}_v$)
INPUT:	A query $Q \in \mathcal{L}_s$, a view $\mathbf{V} = \{Q_1, \ldots, Q_\ell\}$ defined in terms of queries in $\mathcal{L}_v$.
QUESTION:	Is $\mathbf{V}$ invertible relative to Q?

By Lemma 1, VDet($\mathcal{L}_s$,$\mathcal{L}_v$) can be equivalently stated as the *view determinacy problem* for ($\mathcal{L}_s$,$\mathcal{L}_v$). It is the problem to determine, given $Q \in \mathcal{L}_s$ and $\mathbf{V} = \{Q_1, \ldots, Q_\ell\}$ such that $Q_i \in \mathcal{L}_v$ for $i \in [1,\ell]$, whether $\mathbf{V}$ determines Q. We shall use these two statements interchangeably in the sequel.

We shall also consider *the query preservation problem*:

PROBLEM:	QPre($\mathcal{L}_s$,$\mathcal{L}_v$,$\mathcal{L}_q$)
INPUT:	A query $Q \in \mathcal{L}_s$, a view $\mathbf{V} = \{Q_1,\ldots,Q_\ell\}$ defined in terms of queries in $\mathcal{L}_v$, and a query language $\mathcal{L}_q$.
QUESTION:	Is $\mathbf{V}$ query preserving relative to Q and $\mathcal{L}_q$?

Query languages used in this paper range over: (1) CQ, the class of conjunctive queries built up from relation atoms, by closing under conjunction $\wedge$ and existential quantification $\exists$; (2) FO, first-order logic queries built from atomic formulas using $\wedge$, disjunction $\vee$, negation $\neg$, $\exists$ and universal quantification $\forall$; and (3) DATALOG, datalog queries defined as a collection of rules $p(\bar{x})$: $-\ p_1(\bar{x}_1),\ldots,p_n(\bar{x}_n)$, where each p_i is either an atomic formula (a relation atom in $\mathcal{R}$, or equality $=$), or an IDB predicate. That is, DATALOG is an extension of union of conjunctive queries with an inflational fixpoint operator. We refer to [8] for more details concerning these languages.

Recall that the class of conjunctive queries, CQ, is the class of SPC queries built up from the relational algebra operators: selection (S), projection (P) and Cartesian product (C). We also consider fragments of CQ, denoted by listing the operators allowed in the fragment. In particular, we consider the following three classes of CQ queries:

- SP: the fragment defined with S and P operators only;
- PC: the fragment defined with P and C operators only;
- SC: the fragment defined with S and C operators only.

4. Undecidability results

In this section we study VDet($\mathcal{L}_s$,$\mathcal{L}_v$) and QPre($\mathcal{L}_s$,$\mathcal{L}_v$,$\mathcal{L}_q$) when $\mathcal{L}_s$ or $\mathcal{L}_v$ is either FO or DATALOG. The main results are negative: both problems are undecidable in these settings.

We first consider the invertibility problem for DATALOG. It is known that the view determinacy problem is undecidable when either $\mathcal{L}_s$ or $\mathcal{L}_v$ is FO, and when both $\mathcal{L}_s$ and $\mathcal{L}_v$ are UCQ [5]. By Lemma 1, the undecidability results carry over to VDet($\mathcal{L}_s$,$\mathcal{L}_v$). We next show that VDet($\mathcal{L}_s$,$\mathcal{L}_v$) (and hence view determinacy) is also undecidable when $\mathcal{L}_s$ is DATALOG and $\mathcal{L}_v$ is CQ, and when $\mathcal{L}_s$ is CQ and $\mathcal{L}_v$ is DATALOG.

Theorem 1. VDet($\mathcal{L}_s$,$\mathcal{L}_v$) *is undecidable when*

(1) $\mathcal{L}_s$ *is* DATALOG *and* $\mathcal{L}_v$ *is* CQ, *or*
(2) $\mathcal{L}_s$ *is* CQ *and* $\mathcal{L}_v$ *is* DATALOG. □

Proof. Both proofs are by reduction from the containment problem for DATALOG, which is to determine, given two DATALOG queries Q_1 and Q_2, whether $Q_1(D) \subseteq Q_2(D)$ for every instance D. This problem is known to be undecidable [19].

(1) VDet(DATALOG,CQ). Let Q_1 and Q_2 be two DATALOG queries defined over schema $\mathcal{R}$, with answer predicates $\text{ans}_1(\bar{x})$ and $\text{ans}_2(\bar{x})$, respectively. Let N be a nullary relation symbol not appearing in $\mathcal{R}$. We define a DATALOG query Q over $(\mathcal{R},N)$ consisting of the rules of Q_1 and Q_2 together with

$Q(\bar{x})$: $-\ \text{ans}_1(\bar{x}),N()$

$Q(\bar{x})$: $-\text{ans}_2(\bar{x})$

The CQ view $\mathbf{V}$ over $(\mathcal{R},N)$ is defined such that for any instance (D,I_N) of $(\mathcal{R},N)$, $\mathbf{V}(D,I_N) = D$. We next show that $Q_1 \subseteq Q_2$ iff $\mathbf{V} \twoheadrightarrow Q$. Suppose that $Q_1 \subseteq Q_2$. Then Q is equivalent to Q_2. Since $\mathbf{V}$ simply copies the instance D of $\mathcal{R}$ we have that $Q \Rightarrow_{\mathbf{V}} Q_2$, from which $\mathbf{V} \twoheadrightarrow Q$ follows. Conversely, if $Q_1 \not\subseteq Q_2$, then there exists an instance D of $\mathcal{R}$ such that $Q_1(D) \cup Q_2(D) \neq Q_2(D)$. Given such D, we define two database instances $D_1 = (D,\{()\})$ and $D_2 = (D,\emptyset)$ of $(\mathcal{R},N)$. Because $\mathbf{V}(D_1) = \mathbf{V}(D_2) = D$ but $Q(D_1) \neq Q(D_2)$, we can conclude that $\mathbf{V}$ does not determine Q.

(2) VDet(CQ,DATALOG). Let Q_1 and Q_2 be two DATALOG queries defined over relational schema $\mathcal{R}$, with answer predicates $\text{ans}_1(\bar{x})$ and $\text{ans}_2(\bar{x})$, respectively. Let R_1, R_2 be two relation symbols not appearing in $\mathcal{R}$, which have the same arity as ans_1 and ans_2. We define the DATALOG view $\mathbf{V}$ over $(\mathcal{R},R_1,R_2)$ as $\mathbf{V} = \{V_1,V_2,V_3\}$, where

$V_1(\bar{x})$: $-R_1(\bar{x}),\text{ans}_2(\bar{y}),R_2(\bar{y})$

$V_2(\bar{x})$: $-R_1(\bar{x})$

$V_2(\bar{x})$: $-\text{ans}_1(\bar{x}),R_2(\bar{x})$

$V_3(\bar{x})$: $-\text{ans}_2(\bar{x}),R_2(\bar{x})$

We define the CQ query Q such that for any instance $D' = (D,I_1,I_2)$ over $\{\mathcal{R},R_1,R_2\}$, $Q(D') = I_1$. We next show that $Q_1 \subseteq Q_2$ iff $\mathbf{V} \twoheadrightarrow Q$. Suppose that $Q_1 \subseteq Q_2$. We can define the inverse Q^{-1} in FO as follows:

$$Q^{-1}(\bar{x}) = \exists \bar{y}((V_1(\bar{x}) \wedge V_3(\bar{y})) \vee (V_2(\bar{x}) \wedge \neg V_3(\bar{y})))$$

Indeed, for any database instance $D' = (D,I_1,I_2)$ of schema $(\mathcal{R},R_1,R_2)$, if $V_3(D')$ is nonempty, then $V_1(D')$ returns I_1. If $V_3(D') = \emptyset$, which means that $Q_2(D) \cap I_2 = \emptyset$, then from $Q_1 \subseteq Q_2$ we can conclude that $Q_1(D) \cap I_2 = \emptyset$ and hence $V_2(D')$ returns I_1. That is, $Q \Rightarrow_{\mathbf{V}} Q^{-1}$ and hence $\mathbf{V} \twoheadrightarrow Q$. Conversely, suppose that $Q_1 \not\subseteq Q_2$. Then there exists an instance D of relational schema $\mathcal{R}$ and a tuple $\bar{t}$ such that $\bar{t} \in Q_1(D)$ and $\bar{t} \notin Q_2(D)$. Given such D and $\bar{t}$, we define two instances $D' = (D,\{\bar{t}\},\{\bar{t}\})$ and $D'' = (D,\emptyset,\{\bar{t}\})$. It is easy to see that $\mathbf{V}(D') = \mathbf{V}(D'') = (\emptyset,\{\bar{t}\},\emptyset)$ but $Q(D') = \{\bar{t}\} \neq Q(D'') = \emptyset$. Thus $\mathbf{V}$ does not determine Q. □

When it comes to QPre($\mathcal{L}_s$,$\mathcal{L}_v$,$\mathcal{L}_q$), the query preservation problem, we show that it is also beyond reach in practice when any of $\mathcal{L}_s$ and $\mathcal{L}_v$ is either FO or DATALOG.

Theorem 2. QPre($\mathcal{L}_s$,$\mathcal{L}_v$,$\mathcal{L}_q$) *is undecidable when*

(1) $\mathcal{L}_s$ *is* FO, *and* $\mathcal{L}_v$ *and* $\mathcal{L}_q$ *are* CQ,
(2) $\mathcal{L}_s$ *is* CQ, $\mathcal{L}_v$ *is* FO *and* $\mathcal{L}_q$ *is* CQ,
(3) $\mathcal{L}_s$ *is* DATALOG, $\mathcal{L}_v$ *is* CQ *and* $\mathcal{L}_q$ *is* DATALOG, *or*
(4) $\mathcal{L}_s$ *is* CQ, $\mathcal{L}_v$ *is* DATALOG *and* $\mathcal{L}_q$ *is* FO. □

Proof. We show the undecidability of (1) and (2) by reduction from the satisfiability problem of FO, which is known to be undecidable [8]. The undecidability of (3) and (4) follows from the proofs of Theorem 1.

(1) Let $Q_0(\bar{y})$ be an FO query over a relation schema R and let N_1 and N_2 be two nullary relations. Let $\mathcal{R} = (R,N_1,N_2)$. We define the view V and query Q over schema $\mathcal{R}$. Let $V(\bar{x}) = R(\bar{x}) \wedge N_1$ and $Q(\bar{y}) = Q_0(\bar{y}) \wedge N_2$. We show that V is

query preserving relative to Q and CQ iff Q_0 is not satisfiable.

First assume that Q_0 is not satisfiable. Then for any instance $D=(I_R,I_1,I_2)$ of $\mathcal{R}$, $Q(D)=\emptyset$. Moreover, for any $Q' \in$ CQ we have that $Q'(\emptyset)=\emptyset$. Hence $F(Q') := \emptyset$ satisfies the desired properties. Hence V is query preserving relative to Q and V.

Conversely, assume that Q_0 is satisfiable. That is, there exists an instance I_0 of R such that $Q_0(I_0)\neq\emptyset$. By Proposition 1, it suffices to show that V does not determine Q. For if it holds, then by **id** $\in$ CQ, no function F can exist that makes V query preserving. We distinguish between the following two cases: (a) $Q_0(\emptyset)=\emptyset$ and (b) $Q_0(\emptyset)\neq\emptyset$. For case (a), consider instances $D_1=(\emptyset,\emptyset,())$ and $D_2=(I_0,\emptyset,())$, where $Q_0(I_0)\neq\emptyset$ as assumed above. We have that $Q(D_1)=\emptyset$ and $Q(D_2)=Q_0(I_0)\neq\emptyset$ whereas $V(D_1)=V(D_2)=\emptyset$. For case (b), consider instances $D_1=(\emptyset,\emptyset,())$ and $D_2=(I,\emptyset,\emptyset)$, where I is arbitrary. Then $Q(D_1)\neq\emptyset$ and $Q(D_2)=\emptyset$, whereas $V(D_1)=V(D_2)=\emptyset$. Hence, V does not determine Q.

(2) Let $Q_0(\overline{y})$ be an FO query over a single relation schema R. Consider the schema $\mathcal{R}=(R,R')$, where R' is a copy of R. Let $Q(\overline{x})=R(\overline{x})$ and let $V(\overline{x})=R(\overline{x})\wedge(\neg\exists\overline{y}Q_0(\overline{y}))\wedge\neg(\neg\exists\overline{x}R'(\overline{x})\wedge\exists\overline{y}{Q_0}'(\overline{y}))$, where Q_0' is equal to Q_0 but with R replaced by R'. We show that V is query preserving relative to Q and CQ iff Q_0 is not satisfiable.

When Q_0 is not satisfiable, then F can be taken as the identity mapping. One can readily verify that V is query preserving relative to Q and CQ with such a query rewriting function.

On the other hand, if Q_0 is satisfiable, it suffices to show that V does not determine Q. For if it holds, then by **id** $\in$ CQ, no function F can exist that makes V query preserving relative to Q and CQ by Proposition 1. Again we distinguish between the following two cases: (a) $Q_0(\emptyset)=\emptyset$ and (b) $Q_0(\emptyset)\neq\emptyset$. For case (a), consider the instances $D_1=(\emptyset,\emptyset)$ and $D_2=(I,\emptyset)$, where $I\neq\emptyset$ and $Q_0(I)\neq\emptyset$. Since Q_0 is satisfiable, there must exist such an instance I of R. Then $V(D_1)=V(D_2)=\emptyset$, whereas $Q(D_1)=\emptyset\neq I=Q(D_2)$. For case (b), consider $D_1=(\emptyset,\emptyset)$ and $D_2=(I,\emptyset)$, where I is an arbitrary nonempty instance of R. Then again $V(D_1)=V(D_2)=\emptyset$, whereas $Q(D_1)=\emptyset\neq I=Q(D_2)$. In both cases, V does not determine Q.

(3) Consider the view V and query Q as described in the proof of Theorem 1(1). It is easily verified that when $Q_1\subseteq Q_2$ then $F(Q')$ can be defined as the composition of Q with Q'. When $Q_1\not\subseteq Q_2$, it follows from the proof of Theorem 1(1) that V does not determine Q and hence, by Proposition 1, V is not query preserving relative to Q and DATALOG.

(4) Consider V and Q defined in the proof of Theorem 1(2). It is easily verified that when $Q_1\subseteq Q_2$ then $F(Q')$ can be defined as the composition of Q^{-1} with Q'. Since Q^{-1} is in FO, $F(Q')$ is in FO as well. When $Q_1\not\subseteq Q_2$, it follows from the proof of Theorem 1(2) that V does not determine Q. By Proposition 1, V is not query preserving relative to Q and FO. □

5. Decidable Cases for CQ queries

We next study VDet($\mathcal{L}_s,\mathcal{L}_v$) and QPre($\mathcal{L}_s,\mathcal{L}_v,\mathcal{L}_q$) when $\mathcal{L}_s$, $\mathcal{L}_v$ and $\mathcal{L}_q$ are conjunctive queries (CQ). In general, it is unknown whether the view determinacy problem is decidable for conjunctive queries [5]. We focus on special cases VDet(CQ,$\mathcal{L}$) and QPre(CQ,$\mathcal{L}$,CQ), for selection queries Q in CQ and views **V** in a fragment $\mathcal{L}$ of CQ, where $\mathcal{L}$ is SP, PC or SC. We show that these problems are either NP-complete or in PTIME (Theorems 3, 4, Corollaries 2 and 3), and that CQ is complete $\mathcal{L}$-to-CQ rewritings (Corollary 1).

The proofs of Theorems 3 and 4 are nontrivial. To simplify the discussion, we first present some notations and lemmas that will be used throughout the proofs (Section 5.1). We then study VDet and QPre for the special case when selection queries Q are minimal CQ queries (Section 5.2). Finally we extend the results to general CQ queries (Section 5.3).

5.1. Preliminaries

We use $\mathcal{R}=(R_1,\ldots,R_k)$ and $\mathcal{V}=(V_1,\ldots,V_\ell)$ to denote the source and target schema, respectively. Let **var** be an infinite set of variables that are disjoint from **dom**. Let $Q(\overline{x})$ be a CQ query over $\mathcal{R}$ with free variables $\overline{x}$.

We consider instances over the extended domain **dom** $\cup$ **var**. More specifically, we associate with each CQ query Q an instance over this extended domain in the usual way. That is, the *frozen body* of Q, denoted by $[Q]$, is the instance over $\mathcal{R}$ such that $(x_1,\ldots,x_n)$ belongs to the relation in $[Q]$ corresponding to R_i iff $R_i(x_1,\ldots,x_n)$ is an atom in Q. Note that $(x_1,\ldots,x_n)$ may contain both constants (from **dom**) and variables (from **var**). Similarly, for a set **V** of CQ queries, we use [**V**] to denote the union of the frozen bodies $[Q]$ for all Q in **V**.

Consider a mapping h from variables to variables and constants. Let $\overline{t}$ be a tuple over **dom** $\cup$ **var**. Then $h(\overline{t})$ is defined in the usual way by applying h to each component of $\overline{t}$. Similarly, we denote by $h([Q])$ the instance obtained by taking the union of $h(\overline{t})$ for $\overline{t}\in[Q]$. A homomorphism h from an instance I to an instance J over the extended domain, denoted as $h: I\to J$, is a standard homomorphism that is identity on **dom**. More specifically, $h(I)\subseteq J$. Recall that for a CQ query $Q(\overline{x})$ and an instance D, a tuple $\overline{t}$ is in $Q(D)$ iff there exists a homomorphism h from $[Q]$ to D such that $h(\overline{x})=\overline{t}$.

A query Q_1 is contained in a query Q_2, denoted as $Q_1\subseteq Q_2$, if for any instance D, $Q_1(D)\subseteq Q_2(D)$. Two queries are equivalent, denoted as $Q_1\equiv Q_2$, if $Q_1\subseteq Q_2$ and $Q_2\subseteq Q_1$.

A classical result in the theory of conjunctive queries is the following Homomorphism Theorem [20]: Let $Q_1(\overline{x}_1)$ and $Q_2(\overline{x}_2)$ be two CQ queries over the same schema $\mathcal{R}$ with free variables $\overline{x}_1$ and $\overline{x}_2$, respectively. Then $Q_1\subseteq Q_2$ iff there exists a homomorphism h from $[Q_2]$ to $[Q_1]$ such that $h(\overline{x}_2)=\overline{x}_1$, or in other words, $Q_1\subseteq Q_2$ iff $\overline{x}_1\in Q_2([Q_1])$.

The following proposition (slightly modified) from [2] states some observations.

Proposition 3. *Let $Q(\overline{x})$ be a* CQ *query with free variables $\overline{x}$ and let* **V** *be a set of* CQ views. *If* $\mathbf{V}\twoheadrightarrow Q$ *then* (i) $\mathbf{V}([Q])\neq\emptyset$; *and* (ii) *all the relation symbols appearing in Q also appear in some query in* **V**.

Our results make use of the following results on view determinacy for conjunctive queries [1,5]. Let $Q(\overline{x})$ be a CQ query and $\mathbf{V}=\{V_1(\overline{x}_1),\ldots,V_\ell(\overline{x}_\ell)\}$ be a set of views in CQ. Let $\mathcal{S}=(S_1,\ldots,S_\ell)=\mathbf{V}([Q])$. We construct an instance D over $\mathcal{R}$ from $\mathcal{S}$ as follows: For each $i\in[1,\ell]$ and for every tuple $\bar{t}$ belonging to S_i, we include in D the tuples of $h([V_i])$ with $h(\bar{x}_i)=\bar{t}$, where h maps every variable in $[V_i]$ not in $\bar{x}_i$ to some new distinct value. We call this instance the $\mathbf{V}$-inverse of $\mathcal{S}$, denoted as $\mathbf{V}^{-1}(\mathcal{S})$. Let $Q_{\mathbf{V}}(\bar{x})$ be the CQ query over $\mathcal{V}$ with free variables $\bar{x}$ and frozen body $[Q_{\mathbf{V}}]=\mathcal{S}$. The instance $\mathbf{V}^{-1}(\mathcal{S})$ is actually obtained from $[Q_{\mathbf{V}}]$ by unfolding view definitions, with bound variables in view definitions renamed to new distinct variables. That is, $\mathbf{V}^{-1}(\mathcal{S})=[Q_{\mathbf{V}}\circ\mathbf{V}]$. The following proposition (slightly modified) is from [1,5].

Proposition 4. *Let $Q(\bar{x})$ be a* CQ *query and* $\mathbf{V}$ *be a set of* CQ *views. Let $\mathcal{S}=\mathbf{V}([Q])$ and $Q_{\mathbf{V}}(\bar{x})$ be the* CQ *query with $[Q_{\mathbf{V}}]=\mathcal{S}$. We have the following*: (i) *if $\bar{x}\in Q(\mathbf{V}^{-1}(\mathcal{S}))$, then $Q_{\mathbf{V}}$ is a rewriting of Q in terms of* $\mathbf{V}$, *and thus $\mathbf{V}\twoheadrightarrow Q$.* (ii) *if Q has a* CQ *rewriting in terms of* $\mathbf{V}$, *then $Q_{\mathbf{V}}$ is such a rewriting.*

5.2. Minimal CQ queries

We first consider the invertibility problem in which selection queries Q are minimal conjunctive queries. Recall that a conjunctive query Q is minimal if removing any of the rows from $[Q]$ leads to an nonequivalent conjunctive query [8]. For minimal CQ queries, we show the following:

Theorem 3. *When the queries in $\mathcal{L}_s$ are minimal* CQ *queries,* VDet$(\mathcal{L}_s,\mathcal{L}_v)$ *is*

(1) *in* PTIME *when $\mathcal{L}_v$ is* PC,
(2) *in* PTIME *when $\mathcal{L}_v$ is* SP,
(3) NP-*complete when $\mathcal{L}_v$ is* SC. □

The proof is a little involved, and consists of several parts. The PTIME results are shown by leveraging Proposition 4(i). More specifically, for both cases a number of conditions on the query Q and view $\mathbf{V}$ are identified such that (1) when satisfied, the conditions imply that $\bar{x}\in Q(\mathbf{V}^{-1}(\mathcal{S}))$ and thus $\mathbf{V}$ determines Q; and (2) when the conditions are not satisfied, the view does not determine the query. Furthermore, these conditions can be verified in PTIME. The intractability of VDet(CQ,SC) is shown in two steps: First, NP-hardness is established by reduction from the graph 3-colorability problem; and second, the NP upper bound is shown to hold even when queries in $\mathcal{L}_s$ are not minimal. The upper bound proof is deferred to Section 5.3.

Before giving the details of the proof, we elaborate the impact of the minimality assumption for CQ queries in $\mathcal{L}_s$. As previously described, the PTIME results rely on the identification of necessary and sufficient conditions for view determinacy. In order to show that these conditions are necessary, we show that if the conditions fail to hold, then there exist two instances D_1 and D_2 such that $\mathbf{V}(D_1)=\mathbf{V}(D_2)$ but $Q(D_1)\neq Q(D_2)$. We show next that if Q is minimal, then there is a principled way to find (in PTIME) two instances D_1 and D_2 such that $Q(D_1)\neq Q(D_2)$. We shall show in the proof of Theorem 3 that these instances can further be taken such that $\mathbf{V}(D_1)=\mathbf{V}(D_2)$.

More formally, given a CQ query $Q(\bar{x})$ and a tuple $\bar{t}\in[Q]$, we call a set Δ of tuples *critical* for $\bar{t}$ and Q if the CQ queries $Q_1(\bar{x})$ and $Q_2(\bar{x})$ with frozen bodies $[Q_1]=[Q]\cup\Delta$ and $[Q_2]=([Q]\backslash\{\bar{t}\})\cup\Delta$, respectively, satisfy the following two properties: (1) $Q_1\equiv Q$, or in other words, adding Δ does not change the query Q; and (2) $Q\subsetneq Q_2$, that is, replacing $\bar{t}$ with Δ results in a query strictly more general than Q. Critical sets of tuples allow us to construct instances on which Q differs:

Lemma 2. *Let $Q(\bar{x})$ be a* CQ *query, $\bar{t}\in[Q]$ and Δ be a set of critical tuples for $\bar{t}$ and Q. Then for $D_1=[Q]\cup\Delta$ and $D_2=([Q]\backslash\{\bar{t}\})\cup\Delta$ we have that $Q(D_1)\neq Q(D_2)$.*

Proof. Let $Q_1(\bar{x})$ and $Q_2(\bar{x})$ be the CQ queries with frozen bodies $[Q_1]=D_1$ and $[Q_2]=D_2$, respectively. Assume by contradiction that $Q(D_1)=Q(D_2)$. By assumption, $Q_1\equiv Q$, and hence, we also have that $Q_1([Q_1])=Q_1(D_1)=Q_1(D_2)=Q_1([Q_2])$. Furthermore, $\bar{x}\in Q_1([Q_1])$ and thus also $\bar{x}\in Q_1([Q_2])$. By the Homomorphism Theorem, $Q_2\subseteq Q_1$. From the assumption that $Q\subseteq Q_2$, and hence $Q_1\subseteq Q_2$, we can then conclude that $Q\equiv Q_2$. This contradicts the fact that $Q\subsetneq Q_2$ and therefore, $Q(D_1)\neq Q(D_2)$. □

The crucial observation is that when $Q(\bar{x})$ is a minimal query, one can construct critical sets of tuples easily. More precisely, let $\bar{t}\in[Q]$ and let $\bar{s}$ be a tuple obtained from $\bar{t}$ by replacing some occurrences of (i) a constant; or (ii) a variable that appears in multiple rows in $[Q]$; or (iii) a variable that appears in $\bar{x}$, with a distinct new variable; or (iv) replacing some occurrences of a variable that appears multiple times but only in $\bar{t}$, with a distinct new variable while keeping the other occurrences of this variable unchanged. We then have the following:

Lemma 3. *Let $Q(\bar{x})$ be a minimal* CQ *query, $\bar{t}\in[Q]$ and Δ be a set of tuples obtained from $\bar{t}$ as described in* (i)–(iv). *Then Δ is critical for $\bar{t}$ and Q.*

Proof. We need to show that for $Q_1(\bar{x})$ with $[Q_1]=[Q]\cup\Delta$ and $Q_2(\bar{x})$ with $[Q_2]=([Q]\backslash\{\bar{t}\})\cup\Delta$, we have that (a) $Q\equiv Q_1$ and (b) $Q\subsetneq Q_2$. For (a) it suffices to observe that $[Q]\subseteq[Q_1]$ and therefore, $Q_1\subseteq Q$. Furthermore, the trivial homomorphism $h:[Q]\to[Q]$ can be extended to a homomorphism $h':[Q_1]\to[Q]$ since, by construction, every tuple $\bar{s}\in\Delta$ is equal to $\bar{t}$ except that some occurrences of a constant or variable are replaced by a new variable that does not introduce additional equality constraints. Hence, $h'([Q_1])\subseteq[Q]$, $h'(\bar{x})=\bar{x}$ and therefore, $Q\subseteq Q_1$. We can thus conclude that $Q\equiv Q_1$, as desired.

For (b) we first observe that $[Q_2]\subseteq[Q_1]$ and thus $Q_1\subseteq Q_2$. From (a) we can also infer that $Q\subseteq Q_2$. Assume by contradiction that $Q\equiv Q_2$. Since $|[Q_2]|=|[Q_1]|-1\geq|[Q]|$ and Q is minimal, there exists a subset $\mathbf{T}$ of $[Q_2]$ such that $Q_2'(\bar{x})$ with $[Q_2']=\mathbf{T}$ is equivalent to Q, and $|[Q_2']|=|[Q]|$. We consider the following cases: (i) $[Q]\backslash\{\bar{t}\}\subseteq[Q_2']$ and (ii) $[Q]\backslash\{\bar{t}\}\not\subseteq[Q_2']$.

Case (i): Note that $[Q_2']$ consists of all the tuples of $[Q]\backslash\{\bar{t}\}$ plus a newly constructed tuple $\bar{s}\in\Delta$. Since Q_2' and Q are minimal and equivalent, the tableaux $([Q_2'],\bar{x})$ and $([Q],\bar{x})$ are the same up to renaming of variables (cf. [8, Proposition 6.2.9]). This is impossible, however, by the construction of tuples in Δ.

Case (ii): Observe that there exists a tuple $\bar{u}\in[Q]\backslash\{\bar{t}\}$ such that $\bar{u}\notin[Q_2']$. Let $Q'(\bar{x})$ be the CQ query with $[Q']=[Q]\backslash\{\bar{u}\}$. It is easily verified that there exists a homomorphism $h_1:[Q_2']\to[Q']$ with $h_1(\bar{x})=\bar{x}$. On the other hand,

$Q \equiv Q'_2$ and there exists a homomorphism $h_2 : [Q] \to [Q'_2]$ with $h_2(\bar{x}) = \bar{x}$. Thus there exists a homomorphism $h = h_1 \circ h_2 : [Q] \to [Q']$ with $h(\bar{x}) = \bar{x}$ and hence, $Q' \subseteq Q$. Furthermore, from $[Q'] \subseteq [Q]$ we infer that $Q \subseteq Q'$. Hence, $Q' \equiv Q$. This, however, contradicts the assumption that Q is minimal and therefore, $Q \not\equiv Q_2$. □

Proof of Theorem 3. We are now ready to prove Theorem 3.

(1) VDet(CQ,PC). We first consider the case when $\mathbf{V}$ consists of a single view V. We then show the result for general views.

Single PC *view*: Let $Q(\bar{x}_Q)$ be a minimal CQ query and $V(\bar{x}_V)$ be a PC view defined over relational schema $\mathcal{R} = (R_1, \ldots, R_k)$. Since V is a PC query, $[V]$ contains no constants and each variable in $[V]$ appears only once. This implies that for any pair of tuples $\bar{t}_V \in [V]$ and $\bar{t}_Q \in [Q]$ over the same relation in $\mathcal{R}$, there is a unique homomorphism h from $\bar{t}_V$ to $\bar{t}_Q$.

We show that V determines Q iff the following conditions are satisfied: (1) the relation symbols appearing in Q are exactly the same as those appearing in V; (2) for each tuple $\bar{t}_Q \in [Q]$ there exists a tuple $\bar{t}_V \in [V]$ over the same relation in $\mathcal{R}$ such that for each variable x in $\bar{t}_V$ and for the homomorphism h from $\bar{t}_V$ to $\bar{t}_Q$, if (a) $h(x)$ is a constant; or (b) $h(x)$ appears more than once in $[Q]$; or (c) $h(x)$ appears in $\bar{x}_Q$, then x must appear in $\bar{x}_V$. These conditions can be easily checked in PTIME.

We first show that if these conditions hold then $V \twoheadrightarrow Q$. More specifically, we show that the conditions imply that $\bar{x}_Q \in Q(V^{-1}(S))$. Let $S = V([Q])$. By condition (1), $S \neq \emptyset$, and one can construct the instance $V^{-1}(S)$ from S. Consider an arbitrary tuple $\bar{t}_Q \in [Q]$ and let $\bar{t}_V$ be a tuple $[V]$ that satisfies condition (2). Let h be the homomorphism from $\bar{t}_V$ to $\bar{t}_Q$. Since V is a PC query, we can extend h to be a homomorphism $\bar{h}$ from $[V]$ to $[Q]$. Moreover, by the construction of $V^{-1}(S)$, there is a tuple $\bar{t} \in V^{-1}(S)$ such that $\bar{t} = h'(\bar{t}_V)$, where $h'(x) = \bar{h}(x) = h(x)$ if x appears in $\bar{x}_V$, and $h'(x)$ is a new distinct variable otherwise. Now consider tuples $\bar{t}_Q$ and $\bar{t}$. By conditions (2a) and (2b), $\bar{t}_Q$ and $\bar{t}$ are isomorphic, and the homomorphism h'' from $\bar{t}_Q$ to $\bar{t}$ is identity on variables that appear more than once in $[Q]$. By gathering all homomorphism h'' between tuples $\bar{t}_Q$ in $[Q]$ and tuples $\bar{t} \in V^{-1}(S)$, constructed as above from tuples $\bar{t}_V$ that satisfy condition (2), we thus obtain a homomorphism $\bar{h}''$ from $[Q]$ to $V^{-1}(S)$. By condition (2c), $\bar{h}''$ is identity on variables that appear in $\bar{x}_Q$, and thus $\bar{h}''(x_Q) = x_Q$ and $\bar{x}_Q \in Q(V^{-1}(S))$. From Proposition 4(i) it follows that $V \twoheadrightarrow Q$.

We next show that the conditions are also necessary. We first consider condition (1). Suppose that Q has less relation symbols than V. In this case, $V([Q]) = \emptyset$ and by Proposition 3(i), V does not determine Q. If Q has more relation symbols than Q, then by Proposition 3(ii), V cannot determine Q either. In other words, condition (1) needs to be satisfied.

Next, consider condition (2). Suppose that there exists a tuple $\bar{t}_Q \in [Q]$ such that for any tuple $\bar{t}_V \in [V]$ over the same relation as $\bar{t}_Q$, one of the conditions (2a)–(2c) is not satisfied. Let $\bar{t} = \bar{t}_Q$. For each tuple $\bar{t}_V \in [V]$ over the same relation as $\bar{t}$, we construct a tuple $\bar{s}$ from $\bar{t}$ as follows. Let h be the unique homomorphism from $\bar{t}_V$ to $\bar{t}$. Let x be the variable in $\bar{t}_V$ that does not occur in $\bar{x}_V$ but either (a) $h(x)$ is a constant; (b) $h(x)$ occurs multiple times in $[Q]$; or (c) $h(x)$ appears in $\bar{x}_Q$. We then construct $\bar{s}$ from $\bar{t}$ by replacing each $h(x)$ in $\bar{t}$ with a new distinct variable. For each $\bar{t}_V$ we put the resulting tuple $\bar{s}$ in the set Δ. Lemma 3 implies that Δ is critical for $\bar{t}$ and Q, and Lemma 2 tells us that $Q(D_1) \neq Q(D_2)$ for $D_1 = [Q] \cup \Delta$ and $D_2 = ([Q] \setminus \{\bar{t}\}) \cup \Delta$. Hence if $V(D_1) = V(D_2)$, then V does not determine Q.

We now verify that $V(D_1) = V(D_2)$. Since $D_2 \subseteq D_1$, we have that $V(D_2) \subseteq V(D_1)$. Hence, we need to show that $V(D_1) \subseteq V(D_2)$. Let $\bar{u} \in V(D_1)$ and let $h' : [V] \to D_1$ such that $h'(\bar{x}_V) = \bar{u}$. Then for the tuple $\bar{t}_V \in [V]$ such that $h'(\bar{t}_V) = \bar{t}$, we have a tuple $\bar{s} \in \Delta$ that coincides with $\bar{t}$ on variables in $\bar{t}_V$ that occur in $\bar{x}_V$ but may be different on some other attributes. Since V is a PC query, however, we can define $h'' : [V] \to D_2$ such that $h'' = h'$ on $[V] \setminus \{\bar{t}_V\}$ and $h''(\bar{t}_V) = \bar{s}$. Clearly, $h''(\bar{x}_V) = \bar{u}$, and since this argument works for every tuple in $V(D_1)$, we have that $V(D_1) \subseteq V(D_2)$. Hence $V(D_1) = V(D_2)$.

Multiple PC *views*: We next consider VDet(CQ,PC) when $\mathbf{V}$ consists of multiple views $\{V_1(\bar{x}_1), \ldots, V_\ell(\bar{x}_\ell)\}$. We show that VDet(CQ,PC) is in PTIME by reducing the multiple view case to the single-view case.

The reduction is given as follows. First, we divide $\mathbf{V}$ into two sets: $\mathbf{V}_1$ is the set of views $V_i \in \mathbf{V}$ such that V_i contains more relation symbols than Q, and $\mathbf{V}_2 = \mathbf{V} \setminus \mathbf{V}_1$. Next, we consider the *product query* $\mathbf{V}_\otimes$ of the views in $\mathbf{V}_2$. Here we assume that the variables in the V_i's are all distinct and consider $\mathbf{V}_\otimes = \times_{V_i \in \mathbf{V}_2} V_i$ in which the free variables in $V_\otimes$ is the union of the free variables in the V_i's. We show that $\mathbf{V} \twoheadrightarrow Q$ iff $\mathbf{V}_2 \neq \emptyset$ and $\mathbf{V}_\otimes \twoheadrightarrow Q$. Note that both conditions can be checked in PTIME.

Suppose first that the conditions hold. Consider two instances D_1 and D_2 such that $\mathbf{V}(D_1) = \mathbf{V}(D_2)$. This implies that $\mathbf{V}_\otimes(D_1) = \mathbf{V}_\otimes(D_2)$. Because $\mathbf{V}_\otimes \twoheadrightarrow Q$, we can then conclude that $Q(D_1) = Q(D_2)$. In other words, $\mathbf{V} \twoheadrightarrow Q$.

Conversely, suppose that one of the conditions does not hold. Clearly, when all views V_i in $\mathbf{V}$ address more relation symbols than Q, then $V_i([Q]) = \emptyset$ for $i \in [1,\ell]$ and hence $\mathbf{V}([Q]) = \emptyset$. Proposition 3(i) then tells us that $\mathbf{V}$ cannot determine Q. In other words, $\mathbf{V}_2$ must be nonempty. Suppose next that $\mathbf{V}_\otimes$ does not determine Q. Since $\mathbf{V}_\otimes$ is a single view, this implies that the conditions for the single view case (as stated in the proof above) do not hold. As a consequence, we can construct the two instances D_1 and D_2, as in the proof for single PC views, which have the property that $\mathbf{V}_\otimes(D_1) = \mathbf{V}_\otimes(D_2)$ but $Q(D_1) \neq Q(D_2)$. Furthermore, observe that $\mathbf{V}_\otimes(D_1) = \mathbf{V}_\otimes(D_2) \neq \emptyset$ and therefore $V(D_1) = V(D_2)$ for any $D \in \mathbf{V}_2$. Since D_1 and D_2 are constructed from $[Q]$, these instances are empty for all relations not addressed by Q. As a result $V(D_1) = V(D_2) = \emptyset$ for any $V \in \mathbf{V}_1$. Putting these together, $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ but $Q(D_1) \neq Q(D_2)$. Hence $\mathbf{V}$ does not determine Q.

(2) VDet(CQ,SP). We first consider the case when $\mathbf{V}$ consists of a single view V, and then extend the result to general views.

Single SP *view*: Let $Q(\bar{x}_Q)$ be a minimal CQ query and $V(\bar{x}_V)$ be an SP view defined over relational schema $\mathcal{R} = (R_1, \ldots, R_k)$. Since V is an SP query, $[V]$ consists of a single tuple $\bar{t}_V$ over some relation R in $\mathcal{R}$.

We provide necessary and sufficient conditions on V and Q to decide whether V determines Q or not. More specifically, we show that $V \twoheadrightarrow Q$ iff (1) Q only contains the relation symbol R; (2) for each tuple $\overline{t}_Q \in [Q]$, there exists a homomorphism h from $\overline{t}_V$ to $\overline{t}_Q$ and furthermore, for each variable x in $\overline{t}_V$, if (a) $h(x)$ is a constant; or (b) there exists a variable y in $\overline{t}_V$ such that $x \neq y$ but $h(x)=h(y)$; or (c) $h(x)$ appears in multiple tuples in $[Q]$; or finally, (d) if $h(x)$ appears in $\overline{x}_Q$, then x must appear in $\overline{x}_V$. These conditions can easily be checked in PTIME.

We first show that if the conditions above hold then $V \twoheadrightarrow Q$, by showing that the conditions imply that $\overline{x}_Q \in Q(V^{-1}(S))$. Let us consider $S=V([Q])$ in more detail. Suppose that $[Q]$ consists of m tuples $\overline{t}_1,\ldots,\overline{t}_m$. Since V contains only one tuple $\overline{t}_V$, one can easily verify that S is the projection of $\{h_i(\overline{t}_V) | i \in [1,m]\}$ on the attributes corresponding to $\overline{x}_V$, where h_i is the homomorphism from $\overline{t}_V$ to $\overline{t}_i$ for each $\overline{t}_i \in [Q]$. Consequently, we also have an explicit description of $V^{-1}(S)$. Indeed, $V^{-1}(S)=\{h_i'(\overline{t}_V) | i \in [1,m]\}$, where $h_i'(x)=h_i(x)$ if x is a variable in $\overline{x}_V$, and $h_i'(x)=x'$ otherwise, and x' is a distinct new variable not appearing anywhere else. We next show that the conditions imply that $\overline{x}_Q \in Q(V^{-1}(S))$ and hence by Proposition 4(i), that $V \twoheadrightarrow Q$.

We show that $\overline{x}_Q \in Q(V^{-1}(S))$ by constructing a homomorphism $\overline{h}:[Q]\to V^{-1}(S)$ such that $\overline{h}(\overline{x}_Q)=\overline{x}_Q$. Let $\overline{t}_i \in [Q]$ and consider the tuple $\overline{s}_i=h_i'(\overline{t}_V) \in V^{-1}(S)$. Let u be a variable or constant in $\overline{t}_i$ and let $h_i^{-1}(u)=\{x | h_i(x)=u\}$. Observe that conditions (2a) and (2b) imply that $h_i^{-1}(u) \subseteq \overline{x}_V$ or $h_i^{-1}(u) \cap \overline{x}_V=\emptyset$. Furthermore, in the latter case, $h_i^{-1}(u)$ consists of a single element. We define $h_i'':\overline{t}_i \to \overline{s}_i$ as follows: $h_i''(u)=u$ in case that $h_i^{-1}(u)\subseteq \overline{x}_V$, and $h_i''(u)=h_i'(h_i^{-1}(u))$ in case that $h_i^{-1}(u)\cap\overline{x}_V=\emptyset$. By the construction of $V^{-1}(S)$, the mapping h_i'' is an isomorphism from $\overline{t}_i$ to $\overline{s}_i$. Furthermore, condition (2c) guarantees that the union of all h_i'', for $i \in [1,m]$ is a homomorphism $\overline{h}''$ from $[Q]$ to $V^{-1}(S)$ which is, by condition (2d), ensured to be identity on variables in $\overline{x}_Q$. In other words, $\overline{h}''(\overline{x}_Q)=\overline{x}_Q$ and therefore, $\overline{x}_Q \in Q(V^{-1}(S))$.

We next show that the conditions above are also necessary. The necessity of condition (1) follows immediately from Proposition 3(ii). We next consider condition (2) and show that if this condition does not hold, then V does not determine Q.

First suppose that there exists a tuple $\overline{t} \in [Q]$ such that there exists no homomorphism from $\overline{t}_V$ to $\overline{t}$. In this case, the instances $D_1=[Q]\backslash\{\overline{t}\}$ and $D_2=[Q]$ provide a counterexample for view determinacy. Indeed, the lack of homomorphism implies that $V([Q])=V(D_1)$. The minimality of Q, however, implies that $Q([Q])\neq Q(D_1)$. Hence, we may assume that for any tuple $\overline{t} \in [Q]$ there exists a homomorphism h from $\overline{t}_V$ to $\overline{t}$.

Suppose, however, that there exist a tuple $\overline{t}_Q \in [Q]$ and a variable x in $\overline{t}_V$ such that x does not occur in $\overline{x}_V$, but for the homomorphism h from $\overline{t}_V$ to $\overline{t}_Q$, either $h(x)$ is a constant, or $h(x)$ appears in multiple tuples in $[Q]$, or $h(x)$ appears in $\overline{x}_Q$; or there exists another variable y with $h(x)=h(y)$. Let $\overline{t}=\overline{t}_Q$. We construct a tuple $\overline{s}$ from $\overline{t}$ by replacing each occurrence of $h(x)$ in $\overline{t}$ that corresponds to each occurrence of x in $\overline{t}_V$ with a new distinct variable. Note that the replacement does not affect the existence of a homomorphism from $\overline{t}_V$ to $\overline{s}$. Since x does not appear in $\overline{x}_V$, we have that $V(\{\overline{t}\})=V(\{\overline{s}\})$. Let $D_1=[Q]\cup\{\overline{s}\}$ and let $D_2=([Q]\backslash\{\overline{t}\})\cup\{\overline{s}\}$. Since V is an SP query we may conclude that $V(D_1)=V(D_2)$. From Lemma 3 we know that $\Delta=\{\overline{s}\}$ is critical for $\overline{t}$ and $[Q]$, and hence Lemma 2 implies that $Q(D_1)\neq Q(D_2)$. In other words, V does not determine Q.

Multiple SP *views*: We next consider the case when $\mathbf{V}$ consists of a number of SP views. Let Q be a CQ query and $\mathbf{V}$ be a set of SP views. For each view $V \in \mathbf{V}$, $[V]$ consists of one tuple $\overline{t}_V$ over some relation in $\mathcal{R}$. We show that $\mathbf{V}\twoheadrightarrow Q$ iff for each tuple $\overline{t}_Q \in [Q]$, there exist a tuple $\overline{t}_V \in [\mathbf{V}]$ over the same relation as $\overline{t}_Q$ and a homomorphism h from $\overline{t}_V$ to $\overline{t}_Q$ such that the conditions (2a)–(2d) described above are satisfied for t_Q, t_V and h. As before, these conditions can be checked in PTIME.

Along the same lines as in the proof for single SP views, one can readily verify that the conditions are sufficient to determine whether $\mathbf{V}\twoheadrightarrow Q$. We next show their necessity.

Suppose that there exists a tuple $\overline{t}_Q \in [Q]$ for which no tuple $\overline{t}_V \in [\mathbf{V}]$ can be found that can be mapped onto $\overline{t}_Q$. In this case, deleting $\overline{t}_Q$ from $[Q]$ results in $\mathbf{V}([Q])=\mathbf{V}([Q]\backslash\{\overline{t}_Q\})$. The minimality of Q, however, implies that $Q([Q])\neq Q([Q]\backslash\{\overline{t}_Q\})$. Hence, $\mathbf{V}$ does not determine Q.

Next, suppose that there exists a tuple $\overline{t}_Q \in [Q]$ such that, for each tuple $\overline{t}_V \in [\mathbf{V}]$ for which there exists a homomorphism to $\overline{t}_Q$, but one of the conditions (2a)–(2d) do not hold. Let $\overline{t}=\overline{t}_Q$. For each such tuple $\overline{t}_V$ we construct a row $\overline{s}$ from $\overline{t}$, similar to the construction in the proof for single SP views. Let Δ be the set of all the constructed tuples. Let $D_1=[Q]\cup\Delta$ and $D_2=([Q]\backslash\{\overline{t}\})\cup\Delta$. It is readily verified that $\mathbf{V}(D_1)=\mathbf{V}(D_2)$, Δ is critical for $\overline{t}$ and $[Q]$ and hence, $Q(D_1)\neq Q(D_2)$. In other words, $\mathbf{V}$ does not determine Q.

(3) VDet(CQ,SC). We show that VDet(CQ,SC) is NP-hard by reduction from the graph 3-colorability problem, which is known to be NP-complete (cf. [21]). The NP upper bound holds even when queries in $\mathcal{L}_s$ are not minimal, which will be verified in the proof of Theorem 4(2).

The reduction is constructed as follows. Given a graph $G=(V,E)$, we define a (minimal) CQ query Q and an SC view W such that $W\twoheadrightarrow Q$ iff G is 3-colorable. More specifically, let C be a set of three variables disjoint from the set of vertices V, and R be a binary relation. We construct a CQ query $Q(\overline{x})$ such that

$$[Q]=\{(c_1,c_2) | c_1,c_2 \in C, c_1\neq c_2\}$$

and an SC view $W(\overline{x},\overline{y})$ such that

$$[W]=\{(v_1,v_2) | (v_1,v_2)\in E\}\cup\{(c_1,c_2) | c_1,c_2\in C, c_1\neq c_2\}.$$

The free variables in Q are given by $\overline{x}=(c_1,c_2,c_1,c_3,c_2,c_1,c_2,c_3,c_3,c_1,c_3,c_2)$. The free variables of W are $(\overline{x},\overline{y})$, where $\overline{x}$ is as in Q and $\overline{y}$ consists of all edges in E.

We show that W determines Q iff G is 3-colorable. Suppose that G is 3-colorable and let $\gamma: V\to C$ be a 3 coloring of V. Consider the view $W'=([W],\overline{x})$. Then, $h:[W]\to[Q]$ defined as $h(v_i)=\gamma(v_i)$ and $h(c_i)=c_i$ is a homomorphism from $[W]$ to $[Q]$ such that $h(\overline{x})=\overline{x}$. Indeed, since γ is a 3-coloring of V, we have that $h((v,w))=(\gamma(v),\gamma(w))\in[Q]$. Hence, $Q\subseteq W'$. Since $W'\subseteq Q$ we then have that $Q=\pi_{\overline{x}}(W)$. This in turn implies that $W\twoheadrightarrow Q$. On the other hand, suppose that G is not 3-colorable. Then there exists no homomorphism from $[W]$ to $[Q]$ and thus $W([Q])=\emptyset$. From Proposition 3(i) we can conclude that W does not determine Q. □

The proof of Theorem 3 also tells us that CQ is complete for rewriting CQ queries using SC, SP or PC views.

Corollary 1. *The class of conjunctive queries is complete for $\mathcal{L}$-to-CQ rewritings when $\mathcal{L}$ is SC, SP, or PC.* □

Proof. The cases when $\mathcal{L}_v$ is SP or PC follow immediately from the proofs of Theorem 3(1) and (2), respectively. Indeed, in those proofs it has been shown that if $\mathbf{V} \twoheadrightarrow Q$ then Q^{-1} is given by a CQ query. Observe that the statement remains intact for selection queries in $\mathcal{L}_s$ that are not necessarily minimal. Indeed, for a CQ query Q, $\mathbf{V} \twoheadrightarrow Q$ iff $\mathbf{V} \twoheadrightarrow Q_{\min}$, where $Q_{\min}$ is a minimal CQ query equivalent to Q. Moreover, if there exists a query Q^{-1} such that Q^{-1} is a CQ rewriting of $Q_{\min}$ using $\mathbf{V}$, then Q^{-1} is also a CQ rewriting of Q using $\mathbf{V}$, and vice versa.

The case when $\mathcal{L}_v$ is SC follows from Theorem 1 in [2] in which the completeness of CQ is shown for views that do not contain non-distinguishable variables. □

Corollary 1 and Proposition 1, when taken together, tell us that QPre(CQ,$\mathcal{L}_v$,$\mathcal{L}_q$) is equivalent to VDet(CQ,$\mathcal{L}_v$) when the language $\mathcal{L}_q$ subsumes CQ. Hence, from Theorem 3 we obtain:

Corollary 2. *When queries in $\mathcal{L}_s$ are minimal* CQ *queries,* QPre($\mathcal{L}_s$,$\mathcal{L}_v$,CQ) *is*

(1) *in* PTIME *when $\mathcal{L}_v$ is* PC *or* SP, *and*
(2) NP-*complete when $\mathcal{L}_v$ is* SC. □

5.3. Arbitrary CQ queries

We next turn our attention to the general case, that is, when the queries in $\mathcal{L}_s$ are not necessarily minimal. We first consider the invertibility problem. While general CQ queries do not make our lives harder when views are PC queries, they do complicate the invertibility analysis for SP views. Indeed, the invertibility problem becomes intractable for SP views, in contrast to PTIME when queries in $\mathcal{L}_s$ are minimal (Theorem 3(2)).

Theorem 4. VDet(CQ,$\mathcal{L}_v$) *is*

(1) PTIME *when $\mathcal{L}_v$ is* PC, *and*
(2) NP-*complete when $\mathcal{L}_v$ is* SP *or* SC. □

Proof. We first show that VDet is in PTIME for PC views. We then show the intractability of the problem for SP and SC views.

(1) VDet(CQ,PC). We need the following notation. Given a CQ query $Q(\bar{x}_Q)$ and a variable x in $[Q]$, we call x *typed* if x appears only in one column over a single relation table in $[Q]$. For a typed variable x, let Q_x be the set of tuples in $[Q]$ that contain x. If Q_x can be mapped to a single tuple via a homomorphism, we call x *single typed*. Checking whether a variable is typed or single typed can be done in PTIME.

We first consider a single PC view $V(\bar{x}_V)$. We show that $V \twoheadrightarrow Q$ iff the following conditions are satisfied: (1′) the relation symbols appearing in Q are exactly the same as those appearing in V; and (2′) for each row $\bar{t}_Q \in [Q]$, there exists a row $\bar{t}_V \in [V]$ over the same relation such that the following conditions are satisfied for each variable x in $\bar{t}_V$ and for the (unique) homomorphism h from $\bar{t}_V$ to $\bar{t}_Q$: x must appear in $\bar{x}_V$ (a) if $h(x)$ is a constant; or (b) if $h(x)$ is not typed; or (c) $h(x)$ is typed but not single typed; or (d) if $h(x)$ appears in $\bar{x}_Q$.

We show that these conditions on Q and V translate to the conditions described in the proof of Theorem 3(1) when considering a minimal query Q' equivalent to Q and the same view V. From this, the PTIME result follows. More specifically, let $Q'(\bar{x}_Q)$ be the minimal CQ query equivalent to Q such that $[Q'] \subseteq [Q]$. Clearly, Q' and Q access the same set of relation symbols and thus condition (1′) is equivalent to condition (1) in the proof of Theorem 3(1).

We next verify this for condition (2′). Since $Q' \equiv Q$, there exists a homomorphism $h : [Q] \to [Q']$ with $h(\bar{x}_Q) = \bar{x}_Q$. Let $\bar{t}_Q \in [Q]$ and $\bar{t}_V \in [V]$ be two tuples over the same relation. Let $\bar{t}_{Q'} = h(\bar{t}_Q)$. Consider the homomorphisms $h_1 : \bar{t}_V \to \bar{t}_Q$ and $h_2 : \bar{t}_V \to \bar{t}_{Q'}$. It is clear that $h_2 = h \circ h_1$. Let x be a variable in $\bar{t}_V$. We distinguish between the following cases, depending on the conditions stated above: (2′a) If $h_1(x)$ is a constant, then $h_2(x)$ is the same constant; (2′b) If $h_1(x)$ is not typed or (2′c) if $h_1(x)$ is typed but not single typed, then $h_2(x) = h(h_1(x))$ appearsmultiple times in $[Q']$; (2′d) if $h_1(x)$ occurs in $\bar{x}_Q$, then $h_2(x) = h(h_1(x)) = h_1(x)$ also occurs in $\bar{x}_Q$. Therefore the conditions (2′a)–(2′d) for general CQ queries correspond to the conditions (2a)–(2c) for minimal CQ queries in the proof of Theorem 3(1).

When $\mathbf{V}$ consists of multiple PC views, we can verify the statement by reduction to the single-view case, along the same lines as the proof for the multiple view case for minimal CQ queries. We omit the details here to avoid repetition.

(2) VDet(CQ,SP) *and* VDet(CQ,SC): From Theorem 3(3) we already know that VDet(CQ,SC) is NP-hard when queries in $\mathcal{L}_s$ are minimal. This lower bound trivially carries over to the general case. We therefore only need to show that VDet(CQ,SP) is NP-hard for general queries and establish a matching NP upper bound for VDet(CQ,SP) and VDet(CQ,SC).

For the lower bound, we show a stronger result. That is, we show that VDet(CQ,S) is already NP-hard by reduction from the containment problem for CQ queries, which is known to be NP-complete (cf. [8]). Let $Q_1(\bar{x})$ and $Q_2(\bar{x})$ be two CQ queries over the same n-ary relation R. We construct a CQ query Q and an S view $\mathbf{V} = \{V\}$ such that $V \twoheadrightarrow Q$ iff $Q_1 \subseteq Q_2$.

More specifically, let R' be an $(n+1)$-ary relation obtained from R by adding an extra attribute A. We define $Q(\bar{x})$ as a CQ query over R' such that $[Q] = (\{0\} \times [Q_1]) \cup (\{u\} \times [Q_2])$, where u is a variable not appearing anywhere else and $\bar{x}$ denotes the free variables in Q_1 and Q_2, respectively. Let $V(\bar{z})$ be the S-query over R' with

Table 1
Undecidability results. Bold entries are new results shown in this paper.

Problem	Complexity
VDet(FO, CQ)	Undecidable [5, Corollary 2.2]
VDet(CQ, FO)	Undecidable [5, Corollary 2.2]
VDet(**DATALOG, CQ**)	**Undecidable (Theorem 1(1))**
VDet(**CQ, DATALOG**)	**Undecidable (Theorem 1(2))**
VDet(UCQ, UCQ)	Undecidable [5, Theorem 4.1]

Table 2
Decidability results for view determinacy and conjunctive queries.

Minimal queries		General queries	
Problem	Complexity	Problem	Complexity
VDet(SPC, SPC)	Open	VDET(SPC, SPC)	Open
VDet(SPC, PC)	PTIME (Theorem 3(1))	VDET(SPC, PC)	PTIME (Theorem 4(1))
VDet(SPC, SP)	PTIME (Theorem 3(2))	VDET(SPC, SP)	NP-complete (Theorem 4(2))
VDet(SPC,SC)	NP-complete (Theorem 3(3))	VDET(SPC,SC)	NP-complete (Theorem 4(2))

$[V]=(0,z_1,z_2,\ldots,z_n)$. We know from Corollary 1 that CQ is complete for S-to-CQ rewritings and by Proposition 4(i) and (ii) that $V \twoheadrightarrow Q$ iff $\overline{x} \in Q(V^{-1}(S))$.

Observe that for the S-view V and query Q we have that $V([Q])=\{0\}\times[Q_1]=V^{-1}(S)$. We next show that there exists a homomorphism $h:[Q]\rightarrow V^{-1}(S)$ such that $h(\overline{x})=\overline{x}$ iff $Q_1 \subseteq Q_2$. Suppose that $Q_1(\overline{x}) \subseteq Q_2(\overline{x})$. Then there is a homomorphism $h':[Q_2]\rightarrow[Q_1]$ such that $h(\overline{x})=\overline{x}$. Clearly, we can extend h' to a homomorphism from $[Q]$ to $\{0\}\times[Q_1]$ by setting $h=h'$ for variables in $[Q_2]$ and $\overline{h}(u)=0$, and by letting h be the identity for variables in Q_1. Then clearly $h(\overline{x})=\overline{x}$. Conversely, suppose that $h:[Q]\rightarrow\{0\}\times[Q_1]$ is a homomorphism such that $h(\overline{x})=(\overline{x})$. Then h induces a homomorphism h' from Q_2 to Q_1 such that $h'(\overline{x})=\overline{x}$, hereby showing that $Q_1 \subseteq Q_2$.

To see that VDet(CQ, SP) and VDet(CQ, SC) are in NP, observe that by Corollary 1, $\mathbf{V} \twoheadrightarrow Q$ iff the inverse is a CQ query and moreover, this query is an equivalent rewriting of Q using $\mathbf{V}$. In other words, testing determinacy reduces to testing for an equivalent CQ rewriting, which is known to be in NP [11]. □

Along the same lines as Corollary 2, we can readily get the following for the query preservation problem.

Corollary 3. QPre(CQ,$\mathcal{L}_v$,CQ) *is*

(1) *in* PTIME *when* $\mathcal{L}_v$ *is* PC, *and*
(2) NP-*complete when* $\mathcal{L}_v$ *is* SP *or* SC. □

6. Conclusion

We have introduced the notions of query preservation and invertibility to specify the preservation of selected information in data transformations. We have shown that invertibility coincides view determinacy, establishing the connection between selected information preservation and query rewriting using views. We have also investigated two important problems associated with selected information preservation, namely, VDet($\mathcal{L}_s$,$\mathcal{L}_v$) and QPre($\mathcal{L}_s$,$\mathcal{L}_v$,$\mathcal{L}_q$), and provided their complexity bounds for a variety of query languages for expressing selection queries ($\mathcal{L}_s$), views ($\mathcal{L}_v$) and user queries ($\mathcal{L}_q$). We expect that these results will help practitioners determine whether their data transformations are lossless *w.r.t.* important information. In addition, the results are new additions to the study of view determinacy and complete rewriting languages.

We summarize the main complexity results for invertibility (view determinacy) in Tables 1 and 2, annotated with their corresponding theorems. All the results in Table 2 and the highlighted results in Table 1 have not appeared in the literature.

The study of selected information preservation is still preliminary. One open problem is to establish the complexity of VDet(CQ,CQ) and QPre(CQ,CQ,CQ). However, these are by no means trivial: for example, VDet(CQ,CQ) is equivalent to the view determinacy problem for CQ queries and CQ views, whose decidability remains unknown [5]. Another issue is to study VDet($\mathcal{L}$,CQ) and QPre($\mathcal{L}$,CQ,CQ), for CQ views and selection queries Q in $\mathcal{L}$ ranging over SP, PC and SC. A third topic is to identify practical cases of VDet(CQ,CQ) and QPre(CQ,CQ,CQ) that are tractable. In particular, our conjecture is that the analysis would become simpler for key preserving CQ views, *i.e.,*views that retain the keys of base relations involved [22]. Data transformations in practice are either key preserving or can be naturally extended to preserve keys.

Acknowledgments

Fan and Geerts are supported in part by an IBM scalable data analytics for a smarter planet innovation award, and the RSE-NSFC Joint Project Scheme. Fan is also supported in part by the National Basic Research Program of China (973 Program) 2012CB316200 and NSFC 61133002. Zheng is supported in part by NSFC 61070038.

References

[1] L. Segoufin, V. Vianu, Views and queries: determinacy and rewriting, in: PODS, 2005, pp. 49–60.
[2] F. Afrati, Determinacy and query rewriting for conjunctive queries and views, Theoretical Computer Science 412 (2011) 1005–1021.
[3] M. Marx, Queries determined by views: pack your views, in: PODS, 2007, pp. 23–30.
[4] A. Nash, L. Segoufin, V. Vianu, Determinacy and rewriting of conjunctive queries using views: a progress report, in: ICDT, 2007, pp. 59–73.
[5] A. Nash, L. Segoufin, V. Vianu, Views and queries: determinacy and rewriting, Transactions on Database Systems, TODS 35 (3) (2010) 21:1–21:41.
[6] D. Pasailă, Conjunctive queries determinacy and rewriting, in: ICDT, 2011, pp. 220–231.
[7] L. Zheng, H. Chen, Determinacy and rewriting of conjunctive queries over unary database schemas, in: SAC, 2011, pp. 1044–1049.
[8] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[9] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, Lossless regular views, in: PODS, 2002, pp. 247–258.
[10] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, View-based query processing: on the relationship between rewriting, answering and losslessness, Theoretical Computer Science 371 (3) (2007) 169–182.
[11] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, D. Srivastava, Answering queries using views, in: PODS, 1995, pp. 95–104.

[12] C. Chekuri, A. Rajaraman, Conjunctive query containment revisited, in: ICDT, 1997, pp. 56–70.
[13] R. Hull, Relative information capacity of simple relational database schemata, SIAM Journal on Computing 15 (1986) 856–886.
[14] S. Abiteboul, R. Hull, Restructuring hierarchical database objects, Theoretical Computer Science 62 (1988) 3–38.
[15] R.J. Miller, Y.E. Ioannidis, R. Ramakrishnan, The use of information capacity in schema integration and translation, in: VLDB, 1993, pp. 120–133.
[16] R.J. Miller, Y.E. Ioannidis, R. Ramakrishnan, Schema equivalence in heterogeneous systems: bridging theory and practice, Information Systems 19 (1994) 3–31.
[17] D. Barbosa, J. Freire, A.O. Mendelzon, Designing information-preserving mapping schemes for XML, in: VLDB, 2005, pp. 109–120.
[18] W. Fan, P. Bohannon, Information preserving xml schema embedding, Transactions on Database Systems 33 (2008) 4:1–4:44.
[19] O. Shmueli, Equivalence of datalog queries is undecidable, Journal of Logic Programming 15 (1993) 231–241.
[20] A.K. Chandra, P.M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: STOC, 1977, pp. 77–90.
[21] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.
[22] G. Cong, W. Fan, F. Geerts, J. Li, J. Luo, On the complexity of view update analysis and its application to annotation propagation, Transactions on Knowledge and Data Engineering. ⟨http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.27⟩, in press.