Capturing Missing Tuples and Missing Values

TING DENG, RCBD and SKLSDE, Beihang University
WENFEI FAN, Informatics, University of Edinburgh, and RCBD and SKLSDE, Beihang University
FLORIS GEERTS, Department of Mathematics and Computer Science, University of Antwerp

Databases in real life are often neither entirely closed-world nor entirely open-world. Indeed, databases in an enterprise are typically partially closed, in which a part of the data is constrained by master data that contains complete information about the enterprise in certain aspects. It has been shown that despite missing tuples, such a database may turn out to have complete information for answering a query.

This paper studies partially closed databases from which both tuples and attribute values may be missing. We specify such a database in terms of conditional tables constrained by master data, referred to as c-instances. We first propose three models to characterize whether a c-instance T is complete for a query Q relative to master data. That is, depending on how missing values in T are instantiated, the answer to Q in T remains unchanged when new tuples are added. We then investigate three problems, to determine (a) whether a given c-instance is complete for a query Q, (b) whether there exists a c-instance that is complete for Q relative to master data available, and (c) whether a c-instance is a minimal-size database that is complete for Q. We establish matching lower and upper bounds on these problems for queries expressed in a variety of languages, in each of the three models for specifying relative completeness.

Categories and Subject Descriptors: H.2.3 [DATABASE MANAGEMENT]: Languages

General Terms: Design, Algorithms, Theory

Additional Key Words and Phrases: Incomplete information, relative completeness, master data management, partially closed databases, complexity

1. INTRODUCTION

Incomplete information has been a longstanding issue. The scale of the problem is such that it is common to find critical information missing from databases. For instance, it is estimated that pieces of information perceived as being needed for clinical decisions were missing from 13.6% to 81% of the time [Miller Jr. et al. 2005]. Traditionally, the research community adopts either the Closed World Assumption (CWA) or the Open World Assumption (OWA). The CWA assumes that a database has collected all the tuples representing real-world entities, but the values of some attributes in those tuples are possibly missing. The OWA assumes that some tuples representing real-world entities may also be missing (see [Abiteboul et al. 1995; van der Meyden 1998] for surveys).
Real-life databases are, however, often neither entirely closed-world nor entirely open-world. This is particularly evident in Master Data Management (MDM), one of the fastest growing software markets [Microsoft 2008; Radcliffe and White 2008]. Master data is a single repository of high-quality data that provides various applications with a synchronized, consistent view of the core business entities of an enterprise [Loshin 2009]. It is a closed-world database about the enterprise in certain aspects, e.g., employees and customers. In the presence of master data, databases of the enterprise are typically partially closed [Fan and Geerts 2009; 2010b]. Whereas some parts of their data are constrained by the master data, e.g., employees and customers, other parts of the databases are open-world, e.g., sale transactions and service records.

Partially closed databases have recently been studied in [Fan and Geerts 2009; 2010b], in the absence of missing values. Certain information in a partially closed database $\mathcal{I}$ is bounded by master data $D_m$, specified by a set $V$ of containment constraints (CCs for short) from $\mathcal{I}$ to $D_m$. Relative to the master data $D_m$, the database $\mathcal{I}$ is then said to be complete for a query $Q$ if $Q(\mathcal{I}) = Q(\mathcal{I}')$ for every partially closed extension $\mathcal{I}'$ of $\mathcal{I}$, i.e., for every $\mathcal{I}'$ such that $\mathcal{I}' \supseteq \mathcal{I}$ and $(\mathcal{I}', D_m)$ satisfies $V$. That is, adding new tuples to $\mathcal{I}$ either does not change the query answer or violates the CCs. It is shown in [Fan and Geerts 2009; 2010b] that despite missing tuples, a partially closed database may still have complete information for answering queries.

The work of [Fan and Geerts 2009; 2010b] has focused on ground instances, namely, database instances from which tuples are possibly missing, but all the values of the existing tuples are in place. In practice, however, both tuples and values are commonly found missing from a database. This introduces new challenges to characterizing and determining whether a database is complete for a query relative to master data.

**Example 1.1**. Let us first recall the setting when only tuples may be missing from a database. Consider a database $D$ of UK patients, specified by the schema

$$
{\text{MVVisit}}(\text{NHS}, \text{name}, \text{city}, \text{yob}, \text{GD}, \text{Date}, \text{Diag}, \text{DrID})
$$

of which each tuple records the NHS number (NHS), name, address (city), year of birth (yob) and gender (GD) of a UK patient, as well as the date of visit to a doctor specified with ID (DrID) and the diagnosis given by the doctor. Consider a query $Q_1$ to find the names of those patients who were born in 2000 with NHS number ‘915-15-335’ and live in Edinburgh. One can hardly trust the answer $Q_1(D_m)$ since tuples may be missing from $D$, even when no attribute values of the tuples in $D$ are missing.

Not all is lost. Indeed, suppose that there is master data $D_m$ available, specified by schema $\text{Patient}_m(\text{NHS}, \text{name}, \text{yob}, \text{zip}, \text{GD})$, which provides a complete record of those patients living in Edinburgh and born after 1990. Then we can conclude that $Q_1$ finds a complete answer in $D$ provided that $Q_1(D)$ returns all the patients $p$ in $D_m$ with $p[\text{NHS}]=915-15-335$ and $p[\text{yob}]=2000$. Indeed, in this case there is no need to add new tuples to $D$ in order to find complete answers to query $Q_1$ in database $D$. Relative to master data $D_m$, the seemingly incomplete $D$ turns out to be complete for $Q_1$.

In practice, attribute values may also be missing. Following [Grahne 1991; Imieliński and Lipski 1984], we use a conditional table ($c$-table) $T$ to represent such a database, as shown in Fig. 1. In “tuple” $t_2$ of $T$, the values of $t_2[\text{name}]$ and $t_2[\text{yob}]$ are missing, and the condition $t_2[\text{cond}]$ tells us that $t_2[\text{yob}]$ is not 2001; similarly, the condition $t_3[\text{cond}]$ tells us that $t_3[\text{city}]$ is not Edinburgh (Edi). Missing values introduce additional challenges. To characterize whether $T$ is complete for $Q_1$, we have to decide how to fill in the missing values in $T$, in addition to missing tuples.

These suggest that relatively complete databases have to accommodate not only missing tuples but also missing values. In addition, there are several fundamental
questions that are not only of theoretical interest, but are also important to database users and developers. For instance, a user may be eager to know whether a database in use is complete for a query relative to master data. Furthermore, a developer may want to know what is a minimal amount of information one has to collect to build a relatively complete database. These practical needs call for a full treatment of relative information completeness.

**Relative information completeness.** To capture missing values and missing tuples, we extend the notion of partially closed databases [Fan and Geerts 2009; 2010b] to c-instances. A c-instance is a collection of c-tables [Grahne 1991; Imieliński and Lipski 1984] in which certain parts are bounded by master data, via a set of containment constraints (CCs) [Fan and Geerts 2009] (see Section 2.1 for their formal definition).

**Models.** We propose three models to specify whether a c-instance $T$ is complete for a query $Q$ relative to master data $D_m$: $T$ is (1) **strongly complete** if each valuation of $T$ yields a ground instance that is complete for $Q$ relative to $D_m$; (2) **weakly complete** if one can find in $T$ the certain answers to $Q$ over all partially closed extensions of valuations of $T$; and (3) **viably complete** if there exists a valuation of $T$ that is a relatively complete database for $Q$. A user may choose a model that best serves his or her needs.

**Data consistency.** We are interested in databases that are both relatively complete and consistent. The consistency of data is typically specified by integrity constraints, such that errors and conflicts in the data can be detected as violations of the constraints [Arenas et al. 1999; Chomicki 2007] (see [Fan and Geerts 2012] for a recent survey). We investigate the impact of integrity constraints on the analysis of relative completeness. In addition, instead of using a separate language of integrity constraints, we adopt a class of CCs that is also capable of expressing constraints commonly used in data cleaning. More specifically, we consider CCs that can be expressed in terms of conjunctive queries.

**Analysis of c-instances.** We provide complexity bounds on basic issues in connection with c-instances. These problems are to decide, given a c-instance $T$, whether $T$ is (a) consistent, i.e., whether there is any partially closed database represented by $T$, and (b) extensible, i.e., whether there exists any partially closed extension of $T$.

**Main complexity results.** We identify three fundamental problems associated with relative information completeness. Given a query $Q$ and master data $D_m$,

--- the **relatively complete database problem** (denoted by RCDP) is to decide whether a given database is complete for $Q$ relative to $D_m$;

--- the **relatively complete query problem** (RCQP) asks whether it is possible to build a database complete for $Q$ relative to $D_m$; and

--- the **minimality problem** (MINP) is to determine whether a database has a minimal size among those that are complete for $Q$ relative to $D_m$.

We investigate these problems w.r.t. several parameters:
— $L_Q$: the query language in which $Q$ is expressed, ranging over conjunctive queries, (CQ), union of conjunctive queries (UCQ), positive existential FO queries ($\exists\text{FO}^+$), first-order queries (FO), and FP, an extension of $\exists\text{FO}^+$ with an inflational fixpoint operator;
— $c$-instances vs. ground instances, i.e., in the presence or in the absence of missing values; and
— different models of relative completeness, i.e., when a $c$-instance is required to be strongly complete, weakly complete or viably complete for $Q$, relative to $D_m$ and $V$.

All these languages allow equality (=) and inequality ($\neq$), as supported by commercial DBMS; moreover, with $\neq$, we can express containment constraints (CCs) and queries in the same query language (see Section 2.1 for CCs).

We provide a comprehensive picture of these problems with different combinations of these parameters. We establish their lower and upper bounds, all matching, ranging over $O(1)$, coDP, $\Pi^p_2$, $\Sigma^p_2$, $D^p_2$, $\Pi^p_3$, $\Sigma^p_3$, $\Pi^p_4$, $\text{NEXPTIME}$, co$\text{NEXPTIME}$, and undecidable. We summarize the main complexity results in Table I, in which the complexity bounds for ground instances are also listed (enclosed in parentheses) when they differ from their counterparts for $c$-instances, annotated with their corresponding theorems. In addition, we identify tractable special cases of these problems (Section 7).

Our main conclusions are as follows.

(a) All problems are decidable for CQ, UCQ and $\exists\text{FO}^+$, but are mostly undecidable for FO and FP. However, they are decidable for FP in the weak completeness model. Moreover, some problems for CQ and UCQ exhibit different behaviors.

(b) The presence of missing values makes our lives harder when RCDP and MINP are concerned. For example, in the strong completeness model, MINP for CQ is $D^p_2$-complete for ground instances while it is $\Pi^p_3$-complete for $c$-instances; in the viable completeness model, RCDP for CQ is $\Pi^p_2$-complete for ground instances while it is $\Sigma^p_3$-complete for $c$-instances. In contrast, it does not complicate the analyses of RCQP. That is, the complexity of RCDP remains the same for ground or $c$-instances.

(c) The problems have rather diverse complexities in the three different models of relative completeness. For instance, RCQP for FP is undecidable in the strong completeness model, but is trivially decidable for weakly complete $c$-instances. Moreover, in the strong completeness model, RCQP for $c$-instances is equivalent to RCQP for ground instances, but this is no longer the case in the weak completeness model: the undecidability of RCQP for FO for ground instances can not show the undecidability for $c$-instances (see Example 5.3; the precise complexity bounds of RCQP for FO and $c$-instances remain an open problem). On the other hand, RCDP for UCQ is $\Pi^p_2$-complete for the strongly complete $c$-instances but it becomes $\Pi^p_3$-complete in the weak model.

(d) Master data and CCs do not substantially complicate the analyses of these problems. Indeed, from the proofs given in Sections 4-6, we can see that all lower bounds of RCDP, RCQP and MINP hold when master data and CCs are fixed, except of RCDP(CQ) and MINP(CQ) in the weak completeness model.

To the best of our knowledge, apart from the conference version of this paper [Fan and Geerts 2010a], this work is a first treatment of relatively complete databases in the presence of both missing values and missing tuples. We identify important problems associated with partially closed $c$-instances, and provide matching complexity bounds on these problems. A variety of techniques are used to prove these results, including finite-model theoretic constructions, characterizations of relatively complete databases and a wide range of reductions.

**Related work.** This work extends [Fan and Geerts 2009; 2010b] by dealing with missing tuples and missing values. We propose three models for relatively complete databases.
Table I. Complexity results in connection with relative completeness. Here \( \text{NEXPTIME-c} \), \( \text{coNEXPTIME-c} \), \( \text{DP}-c \) and \( \text{NP}-c \) are abbreviations for \( \text{NEXPTIME-complete} \), \( \text{coNEXPTIME-complete} \), \( \text{DP-complete} \) and \( \text{NP-complete} \), respectively.

<table>
<thead>
<tr>
<th>( \mathcal{L}_Q )</th>
<th>RCDP</th>
<th>RCQP</th>
<th>MINP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong completeness</td>
<td>Theorem 4.1</td>
<td>Corollary 4.5</td>
<td>Theorem 4.8</td>
</tr>
<tr>
<td>FO, FP, UCQ, ( \exists \bar{FO}^+ ), CQ</td>
<td>undecidable</td>
<td>( \Pi_2^p )-complete</td>
<td>( \Pi_2^p )-complete ( (\text{D}^p_2\text{-c}) )</td>
</tr>
<tr>
<td>Weak completeness</td>
<td>Theorem 5.1</td>
<td>Theorem 5.4</td>
<td>Theorem 5.6</td>
</tr>
<tr>
<td>FO</td>
<td>undecidable</td>
<td>? (undecidable)</td>
<td>undecidable</td>
</tr>
<tr>
<td>UCQ, ( \exists \bar{FO}^+ ), CQ</td>
<td>( \Pi_2^p )-complete</td>
<td>( O(1) )</td>
<td>( \Pi_2^p )-complete</td>
</tr>
<tr>
<td>viable completeness</td>
<td>Theorem 6.1</td>
<td>Corollary 6.2</td>
<td>Corollary 6.3</td>
</tr>
<tr>
<td>FO, FP, UCQ, ( \exists \bar{FO}^+ ), CQ</td>
<td>undecidable</td>
<td>( \Sigma_2^p )-complete ( (\text{M}^p_2\text{-c}) )</td>
<td>( \Sigma_2^p )-complete ( (\text{D}^p_2\text{-c}) )</td>
</tr>
</tbody>
</table>

c-instances, which were not considered in [Fan and Geerts 2009; 2010b]. For ground instances in the strong model, RCDP and RCQP have been studied in [Fan and Geerts 2009; 2010b] with several cases left open there, but neither c-instances nor ground instances and c-instances for MINP have been considered there. The data complexity of RCDP and MINP for ground instances has been studied in [Cao et al. 2014]. While we mostly focus on combined complexity in this paper, we identify tractable cases of the three problems for data complexity.

This work is an extended version of the conference version [Fan and Geerts 2010a] by including the detailed proofs of all results, which were not presented in [Fan and Geerts 2010a], and a variety of tractable cases (data complexity) in Section 7. To keep the paper within a reasonable page limit we do not consider the boundedness problem, which is studied in [Fan and Geerts 2010a]. Moreover, we set the record straight by providing correct lower and upper bounds: (a) in the strong completeness model, RCDP(\( \mathcal{L}_Q \)) for CQ is \( \Pi_2^p \)-complete instead of \( \Pi_1^p \)-complete for c-instances (see Theorem 4.1); and (b) in the strong completeness or viable completeness model, MINP(\( \mathcal{L}_Q \)) for CQ is \( \Delta_5^p \)-complete instead of \( \Delta_3^p \)-complete for ground instances (see Theorem 4.8).

There has been a host of work on incomplete information, notably representation systems (see [Abiteboul et al. 1995; van der Meyden 1998] for surveys, and more recently, [Olteanu et al. 2008]). This work adopts c-tables [Grahne 1991; Imieliński and Lipski 1984] to represent databases with missing values. Our weak model for relative completeness is based on the certain answer semantics [Imieliński and Lipski 1984], and the strong model has a resemblance to strong representation systems. In contrast, viable complete c-instances do not find a counterpart in [Grahne 1991; Imieliński and Lipski 1984]. The basic issues for c-instances (see Section 3) are similar to the problems studied in [Abiteboul et al. 1991], but in the presence of master data. As opposed to prior work in this area, we aim to model partially closed databases as found in MDM, and to settle their associated decision problems that have not been studied before.

Several approaches have been proposed to modeling databases with missing tuples (e.g., [Gottlob and Zicari 1988; Levy 1996; Motro 1989; Vardi 1986]). A notion of open null was introduced in [Gottlob and Zicari 1988] to model locally controlled open-world databases, in which tuples or values can be marked with open null, while the rest of the data is closed-world. Complete and consistent extensions of an incomplete database were studied in [Vardi 1986]. There has also been work on modeling negative information via logic programming (see [van der Meyden 1998]). Neither master data nor the decision problems studied in this work have been considered there.
Closer to this work are partially complete databases studied in [Levy 1996; Motro 1989], which assume a virtual database $D_c$ that contains complete information in all relevant aspects, and assume that any database $D$ either contains or is defined as views of $D_c$. A notion of answer completeness was proposed there to decide whether a query posed on $D_c$ can be answered in $D$. We assume neither the existence of $D_c$ with entire complete information nor views that define $D$ in terms of $D_c$. And neither missing values nor the problems studied here were considered in [Levy 1996; Motro 1989].

Certain answers have also been studied in data integration and data exchange. In data integration, for a query $Q$ posed on a global database $D_G$, one wants to find the certain answers to $Q$ over all data sources that are consistent with $D_G$ w.r.t. view definitions (see e.g., [Abiteboul and Duschka 1998; Lenzerini 2002]). In data exchange, one wants to find the certain answers to a query over all target databases transformed from data sources via schema mapping (see [Kolaitis 2005; Arenas et al. 2009]). The decision problems studied here are not considered in data exchange or data integration. There has also been work on answering queries using views to decide, e.g., whether views determine queries [Segoufin and Vianu 2005]. Our decision problems cannot be reduced to the problems studied there, and vice versa, because in MDM, one often cannot characterize databases as views of master data.

There has also been work on consistent query answering (e.g., [Arenas et al. 1999; Chomicki 2007]), to find certain answers to a query over all repairs of a database. Master data is not considered there, and we do not consider database repairs in this work. For ground instances in the strong model, RCDP is similar to the problem of query independence from updates [Elkan 1990; Levy and Sagiv 1993]. However, none of the results of [Elkan 1990; Levy and Sagiv 1993] carries over to our setting. We refer to [Fan and Geerts 2009; 2010b] for a more detailed discussion of related work on RCDP and RCQP for ground instances.

Related to this work is the recent work [Libkin 2014], which proposes a new interpretation of query answers over incomplete data. It treats incomplete databases as logical theories, and query answering as logical implication (rather than certain answers); it defines representation systems under the CWA and OWA with respect to an information ordering. In contrast to [Libkin 2014], we study relative information completeness in the presence of master data, for databases that are neither entirely closed world nor entirely open world. In this setting, we define three completeness models (strong, weak and viable), and investigate associated problems RCDP, RCQP and MINP for deciding relative completeness, which are not considered in [Libkin 2014]. Note that the models of completeness and the decision problems studied here are also meaningful under the new semantics of [Libkin 2014], although the complexity bounds may be different.

Complementary to this work is the recent work on assessing partial results, i.e., query answers computed with incomplete input due to failures in data access [Lang et al. 2014]. With respect to incomplete data sources, it proposes a framework to classify partial results (i.e., cardinality and correctness) and to determine the degree of partial result classification precision. In contrast, we study how to determine whether input data is complete for our queries relative to available master data. The problems studied in this work are not considered in [Lang et al. 2014], and vice versa. This said, after the input is found incomplete, the methods of [Lang et al. 2014] can be triggered to evaluate the quality of partial answers computed from the input.

**Organization.** Section 2 presents three models for specifying relatively complete $c$-instances. Section 3 investigates the impact of integrity constraints and basic issues in connection with $c$-instances. Problems RCDP, RCQP and MINP are studied in Sections 4, 5 and 6 for strongly complete, weakly complete and viably complete $c$-instances,
respectively. Section 7 identifies special cases with tractable data complexity. Finally, Section 8 summarizes the main results and identifies open problems.

2. RELATIVE INFORMATION COMPLETENESS REVISITED

In this section we first review relatively complete ground instances defined in [Fan and Geerts 2009; 2010b] (Section 2.1), and then present three models to characterize relatively complete c-instances (Section 2.2). Finally, we state the decision problems associated with relative information completeness (Section 2.3).

2.1. Relatively Complete Ground Instances

A database schema \( \mathcal{R} \) is a collection \((R_1, \ldots, R_n)\) of relation schemata. Each \(R_i\) is defined over a set of attributes. Its set of attributes is also denoted by \(R_i\). For each attribute \(A\) in \(R_i\), its finite or infinite domain is a set of constants, denoted by \(\text{dom}(A)\).

**Ground instances and master data.** A ground instance \(\mathcal{I}\) of \(\mathcal{R}\) is of the form \((I_1, \ldots, I_n)\), where for each \(i \in [1, n]\), \(I_i\) is an instance of \(R_i\) without missing values. That is, for each \(i \in I_i\) and each \(A \in R_i\), \(t[A]\) is a constant in \(\text{dom}(A)\).

Master data \(D_m\) is a ground instance of a database schema \(\mathcal{R}_m\). It is a consistent and closed-world database.

**Partially closed databases.** We specify the relationship between a database and master data in terms of containment constraints, or CCs for short. A CC \(\phi\) is of the form \(q(\mathcal{R}) \subseteq p(\mathcal{R}_m)\), where \(q\) is a conjunctive query (CQ) defined over schema \(\mathcal{R}\), and \(p\) is a projection query over schema \(\mathcal{R}_m\). A ground instance \(\mathcal{I}\) of \(\mathcal{R}\) and master data \(D_m\) of \(\mathcal{R}_m\) satisfy \(\phi\), denoted by \(((\mathcal{I}, D_m)) \models \phi\), if \(q(\mathcal{I}) \subseteq p(D_m)\).

Intuitively, the CWA is asserted for \(D_m\), which imposes an upper bound on the information extracted by \(q(\mathcal{I})\) from the database \(\mathcal{I}\). On the other hand, the OWA is assumed on the part of \(\mathcal{I}\) that is not constrained by CCs.

**Example 2.1.** Recall the database \(D\) and master data \(D_m\) described in Example 1.1. We specify a set \(V\) of CCs such that for each year \(y\) in the range \([1991, 2014]\), \(V\) includes the CC \(q_y(\text{MVVisit}) \subseteq p(\text{Patient}_m)\), where \(q_y(n, na, y, g) = \exists d, di, i \ (\text{MVVisit}(n, na, c, y, g, d, di, i) \land c = \text{‘EDI’})\), and \(p(n, na, y', g) = \exists z (\text{Patient}_m(n, na, y', z, g))\), which assures that \(D_m\) is an upper bound on the information in \(D\) about patients who lives in Edinburgh and are born between 1991 and 2014.

Certain integrity constraints can also be expressed as CCs. For example, consider a functional dependency (FD) \(\phi : (\text{NHS} \rightarrow \text{name}, \text{GD})\), which specifies that in the UK, the NHS number determines the name and gender of each patient. Furthermore, assume that master data contains an empty relation \(D_0\). Then the FD \(\phi\) can be enforced by including the following two CCs in \(V\): \(q_{\text{name}} \subseteq D_0\) and \(q_{\text{GD}} \subseteq D_0\), where

\[
q_{\text{name}} = \exists n, na_1, na_2, c_1, c_2, y_1, y_2, g_1, g_2, d_1, d_2, di_1, i_1, i_2
\]
\[
(\text{MVVisit}(n, na_1, c_1, y_1, g_1, d_1, di_1, i_1) \land \text{MVVisit}(n, na_2, c_2, y_2, g_2, d_2, di_2, i_2) \land n_1 \neq n_2),
\]

which detects violations of the FD \(\text{NHS} \rightarrow \text{name}\); similarly, \(q_{\text{GD}}\) is defined to detect violations of the FD \(\text{NHS} \rightarrow \text{GD}\). Note that we allow inequalities in CQ and hence also in CCs. It is shown in [Fan and Geerts 2009; 2010b] that inclusion dependencies (INDs) can be expressed as CCs \(q(\mathcal{R}) \subseteq p(\mathcal{R}_m)\) when \(q\) is in FO, referred to as CCs in FO (see more details about INDs in Section 3).

We say that \(((\mathcal{I}, D_m)) \models \phi\) for each CC \(\phi\) in \(V\).
A ground instance $\mathcal{I}$ of $\mathcal{R}$ is said to be \textit{partially closed} relative to $(D_m, V)$ if $(\mathcal{I}, D_m) \models V$. That is, the information in $\mathcal{I}$ is partially bounded by $D_m$ via the CCs in $V$.

\textbf{Relatively complete ground instances.} Consider ground instances $\mathcal{I} = (I_1, \ldots, I_n)$ and $\mathcal{I}' = (I'_1, \ldots, I'_n)$ of $\mathcal{R}$. We say that the instance $\mathcal{I}'$ extends $\mathcal{I}$, denoted by $\mathcal{I} \subseteq \mathcal{I}'$, if for all $i \in [1, n]$, $I_i \subseteq I'_i$, and furthermore, there is a $j \in [1, n]$ such that $I_j \subseteq I'_j$. The set of \textit{partially closed extensions} of $\mathcal{I}$ is defined as:

$$\text{Ext}(\mathcal{I}, D_m, V) = \{ \mathcal{I}' \mid \mathcal{I} \subseteq \mathcal{I}', (\mathcal{I}', D_m) \models V \}.$$ 

That is, for each $\mathcal{I}'$ in the set, (a) $\mathcal{I}'$ extends $\mathcal{I}$ by including new tuples, and (b) $\mathcal{I}'$ is partially closed relative to $(D_m, V)$. We write $\text{Ext}(\mathcal{I}, D_m, V)$ as $\text{Ext}(\mathcal{I})$ when $D_m$ and $V$ are clear from the context.

A ground instance $\mathcal{I}$ is said to be \textit{complete} for a query $Q$ relative to $(D_m, V)$ if (i) it is partially closed; and (ii) for each $\mathcal{I}' \in \text{Ext}(\mathcal{I})$, $Q(\mathcal{I}) = Q(\mathcal{I}')$. In other words, the answer to $Q$ in $\mathcal{I}$ remains unchanged no matter what new tuples are added to $\mathcal{I}$. Intuitively, $\mathcal{I}$ already has complete information for answering $Q$. The completeness is \textit{relative to $(D_m, V)$}: the extensions must satisfy $V$.

\textbf{Example 2.2.} Recall the ground instances $D$, $D_m$ and the query $Q_1$ from Example 1.1, and let $V$ be the set of CCs from Example 2.1. Then as shown in Example 1.1, $D$ is complete for $Q_1$ relative to $(D_m, V)$ as long as it returns all relevant tuples in $D_m$.

Consider another query $Q_2$ that is to find the names of all patients who were born in 2000 and have NHS number 915-15-321. Suppose that there are such patient records in $D_m$, but $Q_2(D)$ is empty. Then $D$ is not complete for $Q_2$. We can make $D$ complete for $Q_2$, however, by adding to $D$ a single tuple $t$ with $t[\text{NHS}] = '915-15-321'$. Indeed, $V$ includes the CCs encoding FD $\phi$, which assures that there exists at most one patient with this NHS number. Thus the extended $D$ is complete for $Q_2$ relative to $(D_m, V)$.

In contrast, consider the query $Q_3$ that is to find the names of all patients who were diagnosed as diabetics in 2000, no matter where they live. Then the master data $D_m$ does not help. Indeed, it has no information about patients living in cities other than Edinburgh. In this case we cannot make $D$ complete for $Q_3$ relative to $(D_m, V)$. \hfill $\Box$ 

\section{2.2. Accommodating Missing Values}

To specify databases with missing values, we adopt conditional tables (or c-tables) that are specified using variables and local conditions [Grahne 1991; Imielinski and Lipski 1984]. To define c-tables, for each relation schema $R_i$ and each attribute $A$ in $R_i$, we assume a countably infinite set $\text{var}(A)$ of \textit{variables} such that $\text{var}(A) \cap \text{dom}(A) = \emptyset$, $\text{var}(A) \cap \text{dom}(B) = \emptyset$, and $\text{var}(A) \cap \text{var}(B) = \emptyset$ for every attribute $B$ distinct from $A$.

\textbf{Partially closed c-instances.} A c-table of $R_i$ is a pair $(T, \xi)$, where (a) $T$ is a tableau in which for each tuple $t$ and each attribute $A$ in $R_i$, $t[A]$ is either a constant in $\text{dom}(A)$ or a variable in $\text{var}(A)$; and (b) $\xi$ is a mapping that associates a condition $\xi(t)$ with each tuple $t$ in $T$. Here $\xi(t)$ is built up from atoms $x = y$, $x \neq y$, $x = c$, $x \neq c$, by closing under conjunction $\wedge$, where $x, y$ are variables and $c$ is a constant. Denote by $(T, \text{true})$ the $c$-table without any conditions. An example of a $c$-table is shown in Figure 1.

A \textit{valuation} $\mu$ of $(T, \xi)$ is a mapping such that for each tuple $t$ in $T$ and each attribute $A$ in $R_i$, $\mu(t[A])$ is a constant in $\text{dom}(A)$ if $t[A]$ is a variable, and $\mu(t[A]) = t[A]$ if $t[A]$ is a constant. Let $\mu(t)$ be the tuple of $R_i$ obtained by substituting $\mu(x)$ for each occurrence of $x$ in $t$. Then we define

$$\mu(T) = \{ \mu(t) \mid t \in T \text{ and } \xi(\mu(t)) \text{ evaluates to true} \}.$$
Hence, $\mu(T)$ is a ground instance without variables or conditions. More specifically, $(T, \xi)$ represents a set of possible worlds $\mu(T)$ when $\mu$ ranges over all valuations of $(T, \xi)$. We write $(T, \xi)$ simply as $T$ when $\xi$ is clear from the context.

A $c$-instance $T$ of $R$ is of the form $(T_1, \ldots, T_n)$, where for each $i \in [1, n]$, $T_i$ is a $c$-table of $R_i$. A valuation $\mu$ of $T$ is of the form $(\mu_1, \ldots, \mu_n)$, where $\mu_i$ is a valuation of $T_i$.

We use $\mu(T)$ to denote the ground instance $(\mu_1(T_1), \ldots, \mu_n(T_n))$ of $R$. A partially closed $c$-instance $T$ represents a non-empty set of partially closed ground instances, denoted by $\text{Mod}(T, D_m, V)$. That is,

$$\text{Mod}(T, D_m, V) = \{ \mu(T) \mid \mu \text{ is a valuation and } (\mu(T), D_m) \models V \}.$$

We write $\text{Mod}(T, D_m, V)$ as $\text{Mod}(T)$ when $D_m$ and $V$ are clear from the context. We say that a $c$-instance $T$ is partially closed if $\text{Mod}(T, D_m, V)$ is not empty.

To simplify the discussion, in the sequel we consider only $c$-instance $T$ for which $\text{Mod}(T)$ is non-empty. The assumption has no impact on the complexity results of this paper. As will be shown by Proposition 3.3, it is in $\Sigma_2^p$ to decide whether $\text{Mod}(T)$ is non-empty. As we can see from Table I, all the complexity bounds of this paper are higher than $\Sigma_2^p$-complete except $\text{RCQ}$, $\text{UCQ}$ and $\exists FO^+$ in the strong completeness model, and $\text{MINP(CQ)}$ in the weak completeness model. For these two problems, we will show that their complexity bounds remain intact without the assumption.

Databases under the CWA or the OWA are special cases of partially closed $c$-instances. Recall that the CWA assumes that a database has collected all the tuples representing real-world entities, but the values of some attributes in those tuples are possibly missing; the OWA assumes that some tuples representing real-world entities may also be missing. Thus a $c$-instance $T$ is open-world in the absence of master data and CCs and closed-world if the master data is a possible world represented by $T$.

**Relative completeness.** We next define various notions of completeness for $c$-instances. We say that, relative to $(D_m, V)$, a partially closed $c$-instance $T$ is

- strongly complete for $Q$ if for each $I \in \text{Mod}(T)$ and for each $I' \in \text{Ext}(I)$, $Q(I) = Q(I')$;
- weakly complete for $Q$ if

$$\bigcap_{I \in \text{Mod}(T)} Q(I) = \bigcap_{I \in \text{Mod}(T), I' \in \text{Ext}(I)} Q(I'),$$

or for all $I \in \text{Mod}(T)$, $\text{Ext}(I) = \emptyset$; and
- viably complete for $Q$ if there exists $I \in \text{Mod}(T)$ such that for each $I' \in \text{Ext}(I)$, $Q(I) = Q(I')$.

Intuitively, (a) $T$ is strongly complete if no matter how missing values in $T$ are filled in, it yields a ground instance relatively complete for $Q$; (b) $T$ is weakly complete if the certain answer to $Q$ over all partially closed extensions of $T$ can already be found in $T$; and (c) $T$ is viably complete if there exists a way to instantiate missing values in $T$ that results in a ground instance relatively complete for $Q$.

We use $\text{RCQ}^w(Q, D_m, V)$, $\text{RCQ}^u(Q, D_m, V)$ and $\text{RCQ}^v(Q, D_m, V)$ to denote the set of all complete $c$-instances of $R$ for $Q$ w.r.t. $(D_m, V)$, in the strong, weak and viable completeness model, respectively. We simply use $\text{RCQ}(Q, D_m, V)$ when there is no need to distinguish the completeness models.

**Example 2.3.** Consider the $c$-instance $T$ shown in Figure 1, master data $D_m$ and query $Q_1$ of Example 1.1 and the set $V$ of CCs of Example 2.1. Then $T$ is strongly complete for $Q_1$ relative to $(D_m, V)$. Indeed, by the FD $\phi$ encoded as CCs in $V$, we have that for all valuations $\mu$ of $T$, $Q_1(\mu(T))$ returns a single tuple (name=’John’), and the answer to $Q_1$ does not change for every partially closed extension in $\text{Ext}(\mu(T))$. 

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
Consider query $Q_4$ to find the names of patients in Edinburgh who are born in 2000 and visited doctors on 15/03/2015. Suppose that $t_m^1$ and $t_m^2$ are the only patients in $D_m$ born in 2000, where $t_m^1 = (915-15-335, \text{John}, \text{M}, \text{EH8 9AB}, 2000)$ and $t_m^2 = (915-15-336, \text{Bob}, \text{M}, \text{EH8 9AB}, 2000)$. Then relative to $(D_m, V)$, $T$ is viably complete for $Q_4$, since there exists a valuation $\mu$ of $T$ such that $\mu(T)$ is complete. For instance, this happens for $\mu(x) = \text{Bob}$ and $\mu(z) = 2000$. The $c$-instance $T$ is also weakly complete, since the certain answer (name = ‘John’) can already be found over $\text{Mod}(T)$. However, $T$ is not strongly complete for $Q_4$. Indeed, consider $\mu'(T)$ with $\mu'(x) = \text{John}$ and $\mu'(z) = 2000$, and $\mu(T)$ defined as before. Then, clearly, $\mu'(T) \subseteq \mu(T)$ and moreover, $Q_4(\mu'(T))$ only returns John whereas $Q_4(\mu(T))$ returns both John and Bob.

We observe the following: (a) If $T$ is strongly complete, then it is both weakly complete and viably complete. (b) A ground instance $I$ is a $c$-instance without variables and conditions. It is strongly complete and viably complete for a query $Q$ if and only if $I$ is relatively complete for $Q$, as defined in Section 2.1. However, $I$ may be weakly complete but not relatively complete.

**Minimal complete databases.** To decide what data should be collected in a database to answer a query $Q$, we want to identify a minimal amount of information that is complete for $Q$. For this, we use the following notions of minimality.

A ground instance $I$ is a *minimal instance complete* for a query $Q$ relative to $(D_m, V)$ if it is in $\text{RCQ}(Q, D_m, V)$ and moreover, for all $I' \subseteq I$, we have that $I'$ is not in $\text{RCQ}(Q, D_m, V)$. A $c$-instance $T$ is a *minimal $c$-instance viably complete* (resp. strongly complete) for $Q$ relative to $(D_m, V)$ if there exists $I \in \text{Mod}(T)$ (resp. for all $I \in \text{Mod}(T)$) such that $I$ is a minimal instance complete for a query $Q$.

To define minimal instances in the weak model, we write $(T, \xi) \subseteq (T', \xi')$ if $T \subseteq T'$ and $\xi$ is the restriction of $\xi'$ on $T$, i.e., if for each valuation $\mu'$ of $(T', \xi')$, $\mu(T) \subseteq \mu'(T')$, and if $\mu'(T')$ satisfies $\xi'$ then $\mu(T)$ must satisfy $\xi$, where $\mu$ is the restriction of $\mu'$ on $T$. For $T = (T_1, \ldots, T_n)$ and $T' = (T'_1, \ldots, T'_n)$, we write $T \subseteq T'$ if $T_i \subseteq T'_i$ for all $i \in [1, n]$, and $T_i \subseteq T'_j$ for some $j \in [1, n]$.

A database $T$ is a *minimal instance weakly complete* for $Q$ relative to $(D_m, V)$ if $T$ is in $\text{RCQ}(Q, D_m, V)$ and there exists no $T' \subseteq T$ such that $T'$ is in $\text{RCQ}(Q, D_m, V)$. Note that $T'$ can be either a $c$-instance or a ground instance.

**Example 2.4.** Recall $D_m, V$ and $Q_2$ from Example 2.2. As argued there, a ground instance $D$ is minimally strongly complete for $Q_2$ when $D$ consists of a single tuple $t$ with $t[\text{NHS}] = '915-15-321'$. Hence, minimal complete instances may not be unique. In contrast, $D$ is a minimal instance weakly complete for $Q_2$ if $D$ is empty. As shown in Example 2.3, the $c$-instance $T$ of Figure 1 is strongly complete for $Q_1$. However, it is not minimal: removing $t_2 - t_5$ from $T$ yields a smaller complete database. □

### 2.3. Deciding Relative Completeness

We study three problems associated with relatively complete databases, parameterized with a query language $L_Q$.

<table>
<thead>
<tr>
<th><strong>RCDP($L_Q$):</strong></th>
<th>The relatively complete database problem.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT:</strong></td>
<td>A query $Q$ in $L_Q$, master data $D_m$, a set $V$ of CCs, and a partially closed $c$-instance $T$ w.r.t. $(D_m, V)$.</td>
</tr>
<tr>
<td><strong>QUESTION:</strong></td>
<td>Is $T$ in $\text{RCQ}(Q, D_m, V)$?</td>
</tr>
</tbody>
</table>

That is, does $T$ have complete information to answer $Q$?

---

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
Capturing Missing Tuples and Missing Values

RCQP($\mathcal{L}_Q$): The relatively complete query problem.
INPUT: $Q$, $D_m$, and $V$ as in RCDP.
QUESTION: Is $\text{RCQ}(Q, D_m, V)$ non-empty?

It is to determine whether there exists a $c$-instance with complete information to answer $Q$.

MINP($\mathcal{L}_Q$): The minimality problem.
INPUT: $Q$, $D_m$, $V$ and $T$ as in RCDP.
QUESTION: Is $T$ a minimal $c$-instance complete for $Q$ relative to $(D_m, V)$?

This asks whether $T$ is a minimal-size database complete for $Q$, i.e., removing any tuple from $T$ makes it incomplete.

We study these problems when $\mathcal{L}_Q$ ranges over the following query languages (see, e.g., [Abiteboul et al. 1995], for the details):

— CQ, the class of conjunctive queries built up from atomic formulas, i.e., relation atoms in the schema $R$, equality (=) and inequality ($\neq$), by closing under conjunction $\land$ and existential quantification $\exists$;
— UCQ, union of conjunctive queries of the form $Q_1 \cup \cdots \cup Q_k$, where for each $i \in [1, k]$, $Q_i$ is in CQ;
— $\exists\mathcal{FO}^+$, first-order logic (FO) queries built from atomic formulas, by closing under $\land$, disjunction $\lor$ and $\exists$;
— FO queries built from atomic formulas using $\land$, $\lor$, negation $\neg$, $\exists$ and universal quantification $\forall$; and
— FP, an extension of $\exists\mathcal{FO}^+$ with an inflationary fixpoint operator, i.e., queries defined as a collection of rules $p(\vec{x}) \leftarrow p_1(\vec{x}_1), \ldots, p_m(\vec{x}_m)$, where each $p_i$ is either an atomic formula or an IDB predicate.

We also investigate the special case for ground instances. In this setting, RCQP($\mathcal{L}_Q$) is to decide, given $Q$ in $\mathcal{L}_Q$, $D_m$ and $V$, whether there exists a ground instance in $\text{RCQ}(Q, D_m, V)$. Similarly RCDP($\mathcal{L}_Q$) and MINP($\mathcal{L}_Q$) can be stated for ground instances.

We study these problems when $\text{RCQ}(Q, D_m, V)$ is the set of instances that are strongly, weakly or viably complete, in Sections 4, 5 and 6, respectively.

The notations used in this paper are summarized in Table II.

3. ANALYSIS OF PARTIALLY CLOSED DATABASES

Before we study the decision problems for relative completeness, we investigate some basic problems in connection with integrity constraints and partially closed databases.

The impact of integrity constraints. Several classes of constraints have been used to specify data consistency, notably denial constraints and conditional functional dependencies (CFDs) (see [Chomicki 2007; Fan 2008] for surveys). As shown in [Fan and Geerts 2009; 2010b], denial constraints and CFDs can be expressed as CCs defined in Section 2 when $\mathcal{L}_Q$ is CQ. Hence we can enforce both relative information completeness and data consistency using those CCs.

One might want to adopt a class $\mathcal{C}$ of constraints that is more powerful than CCs defined in CQ. However, such $\mathcal{C}$ has an immediate impact on the analysis of relative completeness. For example, it is shown in [Fan and Geerts 2009; 2010b] that inclusion dependencies (INDs) can be expressed as CCs in FO. We show below that when $\mathcal{C}$ consists of, e.g., FDs and INDs, both RCDP($\mathcal{L}_Q$) and RCQP($\mathcal{L}_Q$) are undecidable for any language $\mathcal{L}_Q$, even in the absence of missing values.
We first introduce a couple of notions. In the presence of a set Θ of constraints in \( C \), by a partially closed database \( I \), we mean a database that is partially closed in the usual sense, and in addition, \( I \) satisfies \( Θ \). Similarly, partially closed extensions of \( I \) are also required to satisfy the additional constraints in \( Θ \). More specifically, consider master data \( D_m \), a set \( V \) of CCs, a set \( Θ \) of constraints in \( C \), and a database schema \( R \).

- A ground instance \( I \) of \( R \) is said to be partially closed relative to \((D_m, V, Θ)\) if \((D_m, I) \models V \) and \( I \models Θ \). That is, \( I \) is partially bounded by \( D_m \) via \( V \) and \( I \) is consistent with regards to the CCs in \( V \) and the additional constraints in \( Θ \).

- A ground instance \( I' \) of \( R \) is said to be a partially closed extension of \( I \) relative to \((D_m, V, Θ)\) if \((D_m, I') \models V \) and \( I' \models Θ \).

- A ground instance \( I \) of \( R \) is said to be complete for a query \( Q \) relative to \((D_m, V, Θ)\) if it is partially closed and for each partially closed extension \( I' \) of \( I \), \( Q(I') = Q(I) \).

We use RCQ\((Q, D_m, V, Θ)\) to denote the set of ground instances that are complete for a query \( Q \) relative to \((D_m, V, Θ)\).

**Proposition 3.1.** In the presence of both FDs and INDs, for ground instances, RCDP and RCQP are undecidable even when \( L_Q \) is FO, and master data \( D_m \) and the set \( V \) of CCs are both empty.

**Proof.** To prove Proposition 3.1, it suffices to show that RCDP(CQ) and RCQP(CQ) are undecidable, since CQ is contained in \( L_Q \), when \( L_Q \) is FO, FO, or FP.

We verify the undecidability of RCDP(CQ) and RCQP(CQ) by reduction from the implication problem for FDs and INDs. In particular, we consider instances \((Θ, ϕ)\) of the implication problem, where \( Θ \) is a set of FDs and INDs defined on a database schema \( R \), and \( ϕ \) is an FD \( X → A \) defined on a relation schema \( R \) in \( R \). It is undecidable to determine, given such \((Θ, ϕ)\), whether \( Θ \models ϕ \), i.e., whether for every instance \( I_R \) of \( R \), if \( I_R \models Θ \) then \( I_R \models ϕ \) (cf. [Abiteboul et al. 1995]).

(1) RCDP(CQ). Given an instance \((Θ, ϕ)\) of the implication problem, we define a Boolean query \( Q \) in CQ as follows:

\[
Q(x) = \exists y_1, y_2, w, w'(R(x, w, y_1) \land R(x, w', y_2) \land w \neq w'),
\]

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
where $\vec{x}$ corresponds to attributes $X$ in $R$, $w$ and $w'$ both correspond to attribute $A$ in $R$, as specified by the FD $\varphi : (X \rightarrow A)$ on $R$, and $\vec{y}_1$ and $\vec{y}_2$ encode attributes $R \setminus (X \cup \{A\})$.

Intuitively, for an instance $I_R$ of $R$, the query $Q$ returns true if $I_R \models \varphi$, i.e., when there exist tuples $t_1, t_2$ in $I_R$ such that $t_1[X] = t_2[X]$ but $t_1[A] \neq t_2[A]$; and otherwise, $Q$ returns false. Moreover, we set $D_m$ and $V$ both to be empty.

Consider an instance $I_\emptyset$ of $R$ consisting of empty relations only. We show that $I_\emptyset$ is in $\text{RCQ}(Q, D_m, \Theta, V)$ if and only if $\Theta \models \varphi$. First, assume that $\Theta \models \varphi$. Then for all instances of $I_R$ of $R$, if $I_R \models \varphi$, then $I_R \not\models \varphi$, and hence, $Q$ returns false. Therefore, $I_\emptyset$ is complete for $Q$ relative to $\text{D}_m$, $\text{V}$, $\Theta$.) Conversely, assume that $\Theta \models \varphi$. Then there exists an instance $I_R$ of $R$ such that $I_R \not\models \varphi$ but $I_R \not\models \varphi$. Obviously, $I_R$ is not empty. Then $Q$ returns true, which differs from $Q(I_\emptyset)$. In addition, $I_R$ is a partially closed extension of $I_\emptyset$ since $V$ is empty. From these it follows that $I_\emptyset$ is not in $\text{RCQ}(Q, D_m, V, \Theta)$.

(2) $\text{RCQP}$(CQ). Given an instance $(\Theta, \varphi)$ of the implication problem, we define a CQ query $Q$ and a set $\Theta'$ of INDs and FDs, such that $\Theta \models \varphi$ if and only if $\text{RCQP}(Q, D_m, V, \Theta')$ is non-empty, where master data $D_m$ and the set $V$ of CCs are empty.

To define $\Theta'$ and $Q$, we use a database schema $R'$ that extends $R$ by adding a new attribute $G$ to every relation schema in $R$, where $\text{dom}(G)$ is infinite. The schema $R'$ also includes the unary relation $E$ that consists of a single attribute of an infinite domain. The set $\Theta'$ consists of FDs and INDs constructed as follows.

— For each FD $Y \rightarrow B$ in $\Theta$, the FD $([G, Y] \rightarrow B)$ is in $\Theta'$.
— For each IND $R_1[Y_1] \subseteq R_2[Y_2]$ in $\Theta$, the IND $R_1[G, Y_1] \subseteq R_2[G, Y_2]$ is in $\Theta'$.

Similarly we rewrite the FD $\varphi : X \rightarrow A$ as $\varphi' : ([G, X] \rightarrow A)$. Intuitively, for each instance $I_{R'}$ of $R'$, if we group tuples of $I_{R'}$ by the attribute $G$, then $I_{R'}$ is partitioned into a collection of groups $G_q$, where $g$ ranges over elements in $\text{dom}(G)$ that appear in the $G$-attribute of $I_{R'}$. One can readily verify that $I_{R'} \models \Theta'$ if and only if for each group $I_q$, $I_q \models \Theta$. Similarly, $I_{R'} \models \varphi'$ if and only if $I_q \models \varphi$ for each group $I_q$. The CQ query $Q$ is similar to its counterpart given above. It is defined as follows:

$$Q(z) = E(z) \land \exists g, \vec{x}, \vec{y}_1, \vec{y}_2, w, w'(R(g, \vec{x}, w, \vec{y}_1) \land R(g, \vec{x}, w', \vec{y}_2) \land w \neq w').$$

This query detects whether there exist tuples $t_1$ and $t_2$ that violate the FD $\varphi'$. In other words, it checks whether there exist tuples $t_1$ and $t_2$ from the same group (with the same value in their $G$ attributes) such that $t_1$ and $t_2$ violate $\varphi$. If so, then $Q$ returns the instance $I_E$ of $E$.

We next show that $\Theta \models \varphi$ if and only if $\text{RCQP}(Q, D_m, V, \Theta')$ is non-empty. Assume first that $\Theta \models \varphi$. Then one can readily verify that for every instance $I_{R'}$ of $R'$, if $I_{R'} \models \Theta'$, then $Q(I_{R'})$ is empty. As a result, every instance $I_{R'}$ is in $\text{RCQP}(Q, D_m, V, \Theta')$. Conversely, assume that $\Theta \not\models \varphi$. Then there exists an instance $I_R$ such that $I_R \models \Theta$ but $I_R \not\models \varphi$. Assume by contradiction that there exists $I_R \in \text{RCQP}(Q, D_m, V, \Theta')$. We construct a partially closed extension $I_{R'}$, as follows. Let $g$ be a distinct value that does not appear in any $G$ column of $I_{R'}$. Define $I_{R'}$ such that for each relation $S$ in $R$, its instance in $I_{R'}$ is the union of $I'_S \cup \{(t_g) \times I_S\}$, where $I'_S$ are the instances of $S$ in $I_{R'}$, and $I_S$, respectively, and $t_g$ is a unary tuple with a single attribute $G$ such that $t[G] = g$. In addition, the instance $I_E$ of schema $E$ in $I_{R'}$ properly contains its counterpart $I_E$ in $I_{R'}$. Obviously $I_{R'} \models \Theta$, i.e., $I_{R'}$ is indeed a partially closed extension of $I_{R'}$. However, $Q(I_{R'})$ is $I_E'$, which is by no means equal to the answer to $Q$ in $I_{R'}$, since the latter is either $\Theta$ or $I_E$. This contradicts the assumption that $\text{RCQP}(Q, D_m, V, \Theta')$ is non-empty.

Note that in the proofs above, both master data and the set $V$ of CCs are empty, i.e., they are independent of the instance $(\Theta, \varphi)$ of the implication problem considered.

The undecidability result suggests that we consider integrity constraints that are expressible as CCs in CQ, to focus on the complexity incurred by the analysis of rela-
Reasoning about c-instances. As remarked earlier, the analysis of relative completeness requires decision procedures for determining some basic problems in connection with partially closed c-instances, which are stated as follows.

— The consistency problem is to determine, given master data \( D_m \), a set \( V \) of CCs and a c-instance \( T \), whether \( \text{Mod}(T, D_m, V) \) is non-empty.
— The extensibility problem is to determine, given master data \( D_m \), a set \( V \) of CCs and a ground instance \( I \), whether \( \text{Ext}(I, D_m, V) \) is non-empty, i.e., whether \( I \) can be extended without violating \( V \).

In the sequel we assume that queries are defined over a single relation. This does not lose generality due to the lemma below. For a database schema \( R \), we denote by \( \text{inst}(R) \) the set of all ground instances of \( R \).

**Lemma 3.2.** For every database schema \( R = (R_1, \ldots, R_n) \), there exist a single relation schema \( R \), a linear-time computable bijective function \( f_D : \text{inst}(R) \rightarrow \text{inst}(R) \), a linear-time computable function \( f_Q : L_Q \rightarrow L_Q \), and a linear-time computable function \( f_C \) from CCs to CCs, such that

(a) for all instances \( I \) of \( R \) and any query \( Q \in L_Q \) over \( R \), \( Q(I) = f_Q(Q(f_D(I))) \); and
(b) for every set \( V \) of CCs and master data \( D_m \), \( (I, D_m) \models V \) if and only if \( (f_D(I), D_m) \models f_C(V) \), where \( f_C(V) = \{ f_C(\psi) \mid \psi \in V \} \).

Here \( L_Q \) ranges over \( CQ, \, UCQ, \, \exists \text{FO}^+, \, \text{FO} \) and \( \text{FP} \).

**Proof.** We assume, without loss of generality, that all relations \( R_i \) in \( R \) correspond to the same schema \( R' \). Indeed, one can make the relations \( R_i \) uniform by renaming attributes and adding dummy attributes. Consider a distinct attribute \( A_R \) that takes values from \( \text{dom}(A) = [1, n] \). Define \( R \) to be an extension of \( R' \) by adding attribute \((A_R : \text{dom}(A)) \). We define \( f_D, f_Q \) and \( f_C \) as follows.

1. Define \( f_D \) such that for every instance \( I = (I_1, \ldots, I_n) \) of \( R \), \( f_D(I) = \bigcup_{j \in [1, n]} I_j \times \{(A_R = j)\} \). The function \( f_D \) is clearly bijective.
2. For a query language \( L_Q \), define \( f_Q \) such that given a query \( Q \in L_Q \) defined on \( R \), \( f_Q(Q) \) substitutes \( R(A_R = i, \bar{x}) \) for every occurrence of \( R_i(\bar{x}) \) in \( Q \), i.e., it replaces every occurrence of \( R_i \) with a Project-Select expression \( \pi_{\text{attr}(R_i)}(\sigma_{A_R = i}(R)) \), where \( \text{attr}(R_i) \) denotes the set of attributes in \( R_i \).
3. Similarly, we define \( f_C \) such that for every CC \( \psi, f_C(\psi) \) substitutes \( R(A_R = i, \bar{x}) \) for every occurrence of \( R_i(\bar{x}) \) in \( \psi \).

It is then readily verified that: (a) \( Q(f_D(I)) = f_Q(Q(f_D(I))) \); and (b) for all master data \( D_m \), \( (I, D_m) \models \psi \) if and only if \( (f_D(I), D_m) \models f_C(\psi) \), and then \( (I, D_m) \models V \) if and only if \( (f_D(I), D_m) \models f_C(V) \). Furthermore, \( f_D, f_Q \) and \( f_C \) can be computed in linear time.

**Proposition 3.3.** The consistency and extensibility problems are both \( \Sigma^p_2 \)-complete. The complexity is unchanged even in the absence of local conditions in c-instances and when master data \( D_m \) is fixed.

**Proof.** We show that the consistency problem and the extensibility problem are both \( \Sigma^p_2 \)-complete.

1. The consistency problem. We show that given master data \( D_m \), a set \( V \) of CCs and a c-instance \( T \), it is \( \Sigma^p_2 \)-complete to decide whether \( \text{Mod}(T, D_m, V) \) is non-empty.
Lower bound. We show that the complement of the consistency problem, i.e., the problem to decide whether \( \text{Mod}(T, D_m, V) \) is empty, is \( \Pi_2^p \)-hard. From this it follows that the consistency problem is \( \Sigma_2^p \)-hard. We verify the \( \Pi_2^p \)-hardness by reduction from the \( \forall^+ \exists^3 \text{SAT} \) problem, which is known to be \( \Pi_2^p \)-complete (cf. [Papadimitriou 1994]).

The \( \forall^+ \exists^3 \text{SAT} \) problem is to determine, given a sentence \( \varphi = \forall X \exists Y \psi(X, Y) \), whether \( \varphi \) is true. Here \( X = \{x_1, \ldots, x_n\} \), \( Y = \{y_1, \ldots, y_m\} \), and \( \psi \) is an instance of \( \text{3SAT} \), i.e., \( \psi = C_1 \land \cdots \land C_n \), and for each \( i \in [1, \ell] \), clause \( C_i \) is of the form \( \ell_1 \lor \ell_2 \lor \ell_3 \), where for each \( l \in [1, 3] \), \( \ell_l \) is either a variable or the negation of a variable in \( X \cup Y \).

Given \( \varphi = \forall X \exists Y \psi(X, Y) \), we define a database schema \( R \), a c-instance \( T \) of \( R \), master data \( D_m \), and a set \( V \) of CCs, such that \( \varphi \) is true if and only if \( \text{Mod}(T, D_m, V) \) is empty. We construct \( R, T, D_m \) and \( V \) as follows.

(a) The database schema \( R \) consists of five relation schemas: \( R_{(0,1)}(X) \), \( R_{\nu}(A_1, A_2, B) \), \( R_{\lambda}(A_1, A_2, B) \), \( R_{\tilde{A}}(A, \tilde{A}) \) and \( R_X(X_1, \ldots, X_n) \). Intuitively, \( R_{(0,1)}(X) \), \( R_{\nu}(A_1, A_2, B) \), \( R_{\lambda}(A_1, A_2, B) \), and \( R_{\tilde{A}}(A, \tilde{A}) \) are to store constant relations encoding truth values, disjunction, conjunction and negation of variables, respectively. We use \( R_X(X_1, \ldots, X_n) \) to generate a truth assignment for variables in \( X \).

(b) We construct a c-instance \( T = (I_{(0,1)}, I_{\nu}, I_{\lambda}, I_{\tilde{A}}, T_X) \) in which \( I_{(0,1)} \), \( I_{\nu} \), \( I_{\lambda} \) and \( I_{\tilde{A}} \) are ground relations as shown in Figure 2, to encode the Boolean domain, disjunction, conjunction, and negation, respectively, such that \( \psi \) can be expressed in CQ in terms of these relations, while \( T_X = \{(x_1, \ldots, x_n)\} \) is a c-table defined in terms of variables in \( X \), without any local conditions.

(c) Master data \( D_m \) is specified by five relation schemas: \( R_{m(0,1)}(X) \), \( R_{\nu m}(A_1, A_2, B) \), \( R_{\lambda m}(A_1, A_2, B) \), \( R_{\tilde{A} m}(A, \tilde{A}) \), and \( R_{W m}(W) \). Intuitively, \( R_{m(0,1)}(X) \), \( R_{\nu m}(A_1, A_2, B) \), \( R_{\lambda m}(A_1, A_2, B) \), and \( R_{\tilde{A} m}(A, \tilde{A}) \) are to store constant relations encoding Boolean values, disjunction, conjunction and negation of variables, respectively (denoted by \( R_{m(0,1)} = R_{(0,1)} \), \( R_{\nu m} = R_{\nu} \), \( R_{\lambda m} = R_{\lambda} \), \( R_{\tilde{A} m} = R_{\tilde{A}} \), respectively). The master data instances consist of \( I_{(0,1)}^m = I_{(0,1)} \), \( I_{\nu m}^m = I_{\nu} \), \( I_{\lambda m}^m = I_{\lambda} \), \( I_{\tilde{A} m}^m = I_{\tilde{A}} \), and \( I_{W m}^m = \emptyset \).

(d) The set \( V \) consists of the following CCs:

- \( R_{(0,1)} \subseteq R_{m(0,1)} \), \( R_{\nu} \subseteq R_{\nu m} \), \( R_{\lambda} \subseteq R_{\lambda m} \), \( R_{\tilde{A}} \subseteq R_{\tilde{A} m} \); that is, the tables in \( T \) encoding the Boolean values and operations are fixed;
- for each \( i \in [1, n] \), \( \exists x_1 \cdots x_{i-1} \exists x_{i+1} \cdots x_n R_X(x_1, \ldots, x_n) \subseteq R_{m(0,1)} \); these ensure that each instance of \( R_X \) encoding a valid truth assignment for \( X \);
- \( q(w) \subseteq R_{W m}(w) \), with \( q(w) = \exists x \exists y \exists \bar{x} \exists \bar{y} (Q_X(\bar{x}) \land Q_Y(\bar{y}) \land Q_{\psi}(\bar{x}, \bar{y}, w) \land w = 1) \). Here, \( Q_X(\bar{x}) = R_X(x_1, \ldots, x_n) \) picks a truth assignment for \( X \) and \( Q_Y(\bar{y}) = R_{(0,1)}(y_1) \cdots \cdots R_{(0,1)}(y_m) \) in CQ, i.e., it constructs all possible truth assignments of variables in \( Y \) by means of \( m - 1 \) Cartesian Products of \( I_{(0,1)} \). Furthermore, given a truth assignment \( (\mu_X, \mu_Y) \) of \( (X, Y) \), the subquery \( Q_{\psi}(\mu_X, \mu_Y, w) \) is to evaluate \( \psi(\mu_X, \mu_Y) \) by recording its truth value in \( w \), which is either 0 or 1. While CQ supports neither disjunction nor negation, \( q \) can encode \( \psi \) in CQ by leveraging relations \( I_{\nu}, I_{\lambda} \) and \( I_{\tilde{A}} \). The query \( q(w) \) returns \( \{1\} \) if and only if \( \psi(\mu_X, \mu_Y) \) evaluates true, where \( \mu_X \) and \( \mu_Y \) are the truth assignment selected by \( Q_X(\bar{x}) \) and \( Q_Y(\bar{y}) \), respectively.

Intuitively, for each ground instance \( I = (I_{(0,1)}, I_{\nu}, I_{\lambda}, I_{\tilde{A}}, I_X) \) of \( T \), \( (I, D_m) \models V \) if and only if under the truth assignment \( \mu_X \) of \( X \) variables encoded by \( I_X \), there exists no truth assignment \( \mu_Y \) of \( Y \) variables that makes \( \psi \) true.

We next show that \( \varphi \) is false if and only if \( \text{Mod}(T, D_m, V) \) is not empty.

\[ \Rightarrow \] First assume that \( \varphi \) is false. Then there exists a truth assignment \( \mu_X \) of \( X \) such that there exists no truth assignment \( \mu_Y \) that makes \( \psi(\mu_X, \mu_Y) \) true. Let \( I_{\mu} = \mu(T) = \text{Mod}(T, D_m, V) \).
Conversely, suppose that $\varphi$ is true. Then for all truth assignments $\mu_X$ of $X$, there exists a valuation $\mu_Y$ such that $\psi(\mu_X, \mu_Y)$ evaluates to true. Hence for all ground instances $I_{\mu} = (I_{(0,1)}, I_v, I_\land, I_\lnot, \mu(T_X))$, where $\mu$ sets $x_i$ to 0 or 1 (and thus all possible truth assignments of $X$ are considered), $q(I_{\mu})$ returns $\{(1)\}$. This violates the CC $q(w) \subseteq R^m_0(w)$. Hence $\text{Mod}(T, D_m, V)$ is empty.

**Upper bound.** We next provide a $\Sigma^p_2$ algorithm that, given master data $D_m$, a set $V$ of CCs and a c-instance $T$ as input, returns “yes” if $\text{Mod}(T, D_m, V)$ is non-empty.

By Lemma 3.2, we assume w.l.o.g. that $R$ consists of a single relation schema and $T$ is a c-table $(T, \xi)$. To develop the algorithm we need the following notations.

- We define $\text{Adom}$ to be $S \cup \text{New} \cup d_f$, where (a) $S$ consists of all constants that appear in $T$, $D_m$, or $V$, (b) $\text{New}$ is a set of fresh values that are not in $S$, one for each variable in $T$ or $V$, and (c) $d_f$ is the set including all the values in the finite domains of attributes $A$ in the relation schema that have a finite domain. Intuitively, $\text{Adom}$ consists of all constants in the active domains of $T$, $D_m$ or $V$ and all constants appearing in the domains of attributes with finite domain. As will be seen shortly, for consistency checking, it suffices to consider valuations of a c-table that draws values from $\text{Adom}$ only.

- A valuation $\mu$ of a c-table $(T, \xi)$ on $\text{Adom}$ is a valuation of $(T, \xi)$ where for each variable $x$ in $T$, $\mu(x)$ is in $\text{Adom}$ and moreover, $\mu$ is the identity mapping on constants in $T$. Furthermore, if $x$ appears in an attribute of finite domain, $\mu(x)$ takes values in this finite domain. Note that finite domain values are included in $\text{Adom}$. We use $\mu(T)$ to denote the ground instance $\{\mu(t) | t \in T, \xi(\mu(t)) \text{ evaluates to true}\}$.

- We use $\text{Mod}_{\text{Adom}}(T, D_m, V)$ to denote the set $\{\mu(T) | \mu$ is a valuation on $\text{Adom}$, and $\langle \mu(T), D_m \rangle = V\}$. We also write $\text{Mod}_{\text{Adom}}(T, D_m, V)$ as $\text{Mod}_{\text{Adom}}(T)$ if $D_m$ and $V$ are clear from the context.

Using these notations we give the algorithm as follows. It checks whether there exists a valuation $\mu$ of $(T, \xi)$ such that $(\mu(T), D_m) \models V$.

1. Guess a valuation $\mu$ of $(T, \xi)$ on $\text{Adom}$.
2. Check whether $(\mu(T), D_m) \models V$. If so return “yes”; reject the guess otherwise.

The algorithm is in $\Sigma^p_2$ since it involves guessing a valuation (in NP) combined with a call to a coNP oracle in Step 2. Step 2 is in coNP since the CCs in $V$ are defined with CQ queries and hence, checking whether $(\mu(T), D_m) \not\models V$ can be done in NP as follows.

1. Guess a constraint $q(R) \subseteq p(R_m)$ in $V$ and let $(T_q, u_q)$ be the tableau representing $q$; guess a valuation $\mu_q$ of variables in $T_q$ that takes values from $\mu(T)$.
2. Check whether $\mu_q(u_q) \notin p(D_m)$; if so return “no”; otherwise reject the guess.
Obviously the algorithm above returns “no” if and only if there exists a constraint that is not satisfied by \( \mu(T) \) and \( D_m \). Moreover, the algorithm is in \( \text{NP} \) since its second step is in \( \text{PTIME} \). Hence, the consistency problem is indeed in \( \Sigma^p_2 = \text{NP}^{\text{coNP}} = \text{NP}^{\text{coNP}} \).

The correctness of the algorithm for consistency checking is assured by the following property: \( \text{Mod}(T, D_m, V) \) is non-empty if and only if \( \text{Mod}_{\text{Adom}}(T, D_m, V) \) is non-empty. Hence the algorithm returns “yes” only when \( \text{Mod}(T, D_m, V) \) is non-empty.

We next verify the property. If \( \text{Mod}_{\text{Adom}}(T, D_m, V) \) is non-empty then there exists a valuation \( \mu \) of \( \langle T, \xi \rangle \) on \( \text{Adom} \) such that \( (\mu(T), D_m) \models V \). Then obviously \( \mu(T) \) is in \( \text{Mod}(T, D_m, V) \) and hence \( \text{Mod}(T, D_m, V) \) is non-empty. Conversely, suppose that there exists an instance \( I \) in \( \text{Mod}(T, D_m, V) \). Then there exists a valuation \( \nu \) of \( \langle T, \xi \rangle \) such that \( I = \nu(T) \). We next turn \( \nu \) into a valuation \( \mu \) of \( \langle T, \xi \rangle \) on \( \text{Adom} \), showing that \( \text{Mod}_{\text{Adom}}(T, D_m, V) \) is non-empty. More precisely, we define \( \mu \) such that for every variable \( x \) in \( T \), \( \mu(x) = \nu(x) \) if \( \nu(x) \in \text{Adom} \), and otherwise, \( \mu(x) \) takes a value in \( \text{New} \), such that it preserves the equality on variables, i.e., for all other variables \( y \) in \( T \), \( \mu(x) = \mu(y) \) if and only if \( \nu(x) = \nu(y) \). This is possible since by the definition of \( \text{New} \), this set contains new constants for every variable in \( T \). We need to verify that \( (\mu(T), D_m) \models V \).

Let \( q(\mu(T)) \subseteq p(D_m) \) be any \( \text{CC} \) in \( V \), and \( \langle I_q, u_q \rangle \) be the tableau representing \( q \). Then for each valuation \( \mu_q \) of \( \langle T_q, \xi_q \rangle \) in \( T_q \), there exists a valuation \( \mu_q \) of the variables such that \( \mu_q \) agrees with \( \mu_q \) on variables that are not assigned a \( \text{New} \)-value, but takes values from \( \text{New} \) outside of \( \text{Adom} \), for the remaining variables. Recall that \( \nu(T) \in \text{Mod}(T, D_m, V) \). Thus, \( \mu_q(u_q) \) is in \( p(D_m) \).

One can readily verify that \( \mu_q(u_q) \) is in \( p(D_m) \). As a result, \( (\mu(T), D_m) \models V \) and hence \( \mu(T) \) is in \( \text{Mod}_{\text{Adom}}(T, D_m, V) \).

(2) The extensibility problem. We next show that given \( D_m \), \( V \) and a ground instance \( T \), it is \( \Sigma^p_2 \)-complete to decide whether \( \text{Ext}(T, D_m, V) \) is non-empty.

**Lower bound.** We show that it is \( \Pi^p_2 \)-hard to decide whether \( \text{Ext}(T, D_m, V) \) is empty. This is again verified by reduction from the \( \forall^e \exists^+ \text{3SAT} \) problem. Given an instance \( \varphi = \forall X \exists Y \psi(X, Y) \) of \( \forall^e \exists^+ \text{3SAT} \), we use the same \( \mathcal{R}, D_m \) and \( V \) given in (1), and let \( I_0 = (I_{0(1)}, I_{0}, I_{\lambda}, I_{\omega}, I'_{X}) \), where \( I'_{X} \) is empty.

We next show that \( \varphi \) is true if and only if \( \text{Ext}(I_0, D_m, V) \) is empty.

\[
\begin{align*}
\text{\Rightarrow} \quad & \text{First, assume that } \varphi \text{ is true. Then for all truth assignments } \mu_X \text{ of } X, \text{ there exists a truth assignment } \mu_Y \text{ of } Y \text{ such that } \psi(\mu_X, \mu_Y) \text{ evaluates to true. Hence for all extensions } I'_{0} = (I'_{0(1)}, I_{0}', I_{\lambda}', I_{\omega}', I'_{X}), \text{ as long as } I'_{0(1)} = I_{0(1)}, I_{0}' = I_{0}, I_{\lambda}' = I_{\lambda}, I_{\omega}' = I_{\omega}, \text{ and } I'_{X} \text{ encodes a truth assignment of } X, q(I'_X) \text{ returns } \{1\}. \text{ This, however, violates the } \text{CC} \text{ } q(w) \subseteq R^\emptyset(w) \text{ and hence } \text{Ext}(I_0, D_m, V) \text{ is empty.}
\end{align*}
\]

\[
\begin{align*}
\text{\Leftarrow} \quad & \text{Conversely, assume that } \varphi \text{ is false. Let } I'_0 = (I_{0(1)}, I_{0}', I_{\lambda}, I_{\omega}, I'_X), \text{ where } I'_X \text{ is an instance of } R_X \text{ consisting of a single truth assignment } \mu'_X \text{ of } X \text{ such that } \exists Y \psi(\mu'_X, Y) \text{ is false. Then along the same lines as the previous proof, one can easily verify that } I'_0 \text{ is in } \text{Ext}(I_0, D_m, V). \text{ Upper bound. We now develop a } \Sigma^p_2 \text{ algorithm that takes master data } D_m \text{, a set } V \text{ of } \text{CCs} \text{ and a ground instance } I \text{ as input, and returns “yes” if } \text{Ext}(I, D_m, V) \text{ is not empty.}
\end{align*}
\]

To present the algorithm, we assume w.l.o.g. that \( I \) is an instance of a relation schema \( \mathcal{R} \), by Lemma 3.2. Recall the definition of \( \text{Adom} \) given in the upper bound proof for the consistency problem, except that \( T \) is replaced with the ground instance \( I \). The algorithm works as follows.

(1) Guess a single tuple \( t \) of \( R \) with values from \( \text{Adom} \) that does not belong to \( I \).
(2) Check whether \( (I \cup \{t\}, D_m) \models V \). If so return “yes”; otherwise reject the guess.
4. STRONG RELATIVE INFORMATION COMPLETENESS

We next study RCDP, RCQP, and MINP for strongly relatively complete databases, i.e., databases in which neither missing values nor missing tuples prevent them from having complete information for answering queries relative to master data. We refer to these problems as RCDP\(s\), RCQP\(s\), and MINP\(s\), respectively. Recall that RCQ\(s\)(\(Q\), \(D_m\), \(V\)) denotes the set of instances that are strongly complete for \(Q\) w.r.t. \((D_m, V)\).

We establish complexity bounds on these problems for \(c\)-instances. For ground instances, we give complexity results for MINP\(s\)(\(L_Q\)) not considered in [Fan and Geerts 2010b], and for the cases of RCQP\(s\)(\(L_Q\)) that were left open in [Fan and Geerts 2010b].

Our main conclusion about the strong completeness model is that missing values make our lives harder, but not too much.

4.1. The Relatively Complete Database Problem in the Strong Model

This problem is to decide whether a given database is relatively complete for a query. It is known that for ground instances, RCDP\(s\)(\(L_Q\)) is undecidable when \(L_Q\) is FO or FP, and it is \(\Pi_2^p\)-complete when \(L_Q\) ranges over CQ, UCQ, and \(\exists\text{FO}\) [Fan and Geerts 2009]. The result below tells us that the presence of missing values does not complicates the analysis: all the results for ground instances remain the same for \(c\)-instances.

In practice, master data \(D_m\) and the set \(V\) of CCs are often predefined and fixed, and only databases and user queries vary. One might think that RCDP\(s\) would become simpler in this setting. Unfortunately, this is not the case: the complexity bounds remain intact when \(D_m\) and \(V\) are fixed.

**Theorem 4.1.** For \(c\)-instances, RCDP\(s\)(\(L_Q\)) is

- undecidable when \(L_Q\) is either FO or FP, and
- \(\Pi_2^p\)-complete when \(L_Q\) is CQ, UCQ or \(\exists\text{FO}\).

The complexity bounds remain unchanged when master data \(D_m\) and the set \(V\) of CCs are fixed.

**Proof.** We first show that RCDP\(s\)(\(L_Q\)) is undecidable when \(L_Q\) is FO or FP. We then show that the problem becomes \(\Pi_2^p\)-complete when \(L_Q\) is CQ, UCQ or \(\exists\text{FO}\).

1. When \(L_Q\) is FO or FP. The RCDP\(s\)(FO) and RCDP\(s\)(FP) for ground instances have been proved to be undecidable with fixed \(D_m\) and CCs, by reduction from the satisfiability problem of FO and the emptiness problem for 2-head DFA, respectively (Theorem 3.1 [Fan and Geerts 2010b]; see the details of these two problems in the proofs of Theorem 5.1(2) and Lemma 4.6, respectively). This undecidability carries over to RCDP\(s\)(FO) and RCDP\(s\)(FP) for \(c\)-instances since ground instances are also \(c\)-instances.

2. When \(L_Q\) is CQ, UCQ or \(\exists\text{FO}\). We next show that RCDP\(s\)(\(L_Q\)) is \(\Pi_2^p\)-complete when \(L_Q\) is CQ, UCQ or \(\exists\text{FO}\).

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
Lower bound. It is known that RCDP*(CQ) for ground instances is \( \Pi_2^p \)-hard with fixed master data and CCs, by reduction from the \( \forall \exists \exists \) 3SAT problem (Theorem 3.6 [Fan and Geerts 2010b]; see the details of the latter problem in Proposition 3.3(1)). The lower bound thus carries over since ground instances are \( c \)-instances.

Upper bound. We next show that RCDP*(CQ), RCDP*(UCQ) and RCDP*(\exists FO^+) are all in \( \Pi_2^p \) for \( c \)-instances. We first provide a \( \Pi_2^p \) algorithm for testing strongly complete \( c \)-instances for CQ queries. Later on we show how the algorithm can be extended to UCQ and \( \exists FO^+ \). We first present the algorithm for RCDP*(CQ).

RCDP*(CQ). To show that RCDP*(CQ) is in \( \Pi_2^p \), we first provide a characterization of \( c \)-instances that are strongly complete for CQ queries. Based on the characterization, we then provide a \( \Pi_2^p \) algorithm for testing strongly complete \( c \)-instances for CQ queries.

Consider a CQ query \( Q \), master data \( D_m \), a set \( V \) of CCs and a \( c \)-instance \( T = (T, \xi) \).

By Lemma 3.2, we assume without loss of generality that \( Q \) is defined over a relation schema \( R \), and \( T = (T, \xi) \) is a \( c \)-table of \( R \).

The characterization is defined in terms of the following notations.

— The CQ query \( Q \) can be expressed as a tableau query \((T_Q, u_Q)\), where \( T_Q \) denotes atomic formulas in \( Q \) and \( u_Q \) is the output summary (see e.g., [Abiteboul et al. 1995] for details). Observe that \( T_Q \) can be regarded as a \( c \)-table without local conditions.

— Similarly, we define a set of constants as in the proof of Proposition 3.3(1), also referred to as Adom, including constants in \( Q, T \), \( D_m \) or \( V \), new distinct values in \( \text{New} \) that are not in \( Q, T \), \( D_m \) and \( V \), one for each variable in \( Q \), \( T \) or \( V \), and a set of constants \( d_T \) corresponding to constants in the domains of finite domain attributes. It differs from its counterpart in the proof of Proposition 3.3 in that it includes constants in query \( Q \).

— Furthermore, a ground instance \( I \) of \( R \) is said to be bounded by \((D_m, V)\) if for each \( I' \in \text{Mod}_{\text{Adom}}(I \cup T_Q, D_m, V) \), \( Q(I) = Q(I') \) [Fan and Geerts 2010b].

The following lemma characterizes the strongly complete \( c \)-instances for CQ queries.

**Lemma 4.2.** For every query \( Q \) in CQ, master data \( D_m \), set \( V \) of CCs, and \( c \)-instance \((T, \xi)\), \((T, \xi)\) is in RCDP*(\(Q, D_m, V)\) if and only if for each \( I \in \text{Mod}_{\text{Adom}}(T, D_m, V) \), \( I \) is bounded by \((D_m, V)\).

**Proof.** It suffices to show the following. (1) For each ground instance \( I \) of \( R \), \( I \) is in RCDP*(\(Q, D_m, V)\) if and only if it is bounded by \((D_m, V)\). (2) The \( c \)-instance \((T, \xi)\) is in RCDP*(\(Q, D_m, V)\) if and only if for each \( I \in \text{Mod}_{\text{Adom}}(T, D_m, V) \), \( I \) is in RCDP*(\(Q, D_m, V)\).

(1) We first show that \( I \) is in RCDP*(\(Q, D_m, V)\) if and only if for each \( I' \in \text{Mod}_{\text{Adom}}(I \cup T_Q) \), \( Q(I) = Q(I') \). First, suppose that \( I \) is complete for \( Q \). Then for every \( I' \in \text{Ext}(I), Q(I) = Q(I') \). In particular, for every \( I' \in \text{Mod}_{\text{Adom}}(I \cup T_Q) \), \( Q(I) = Q(I') \) since \( I' \) is also in \( \text{Ext}(I) \). Conversely, suppose that \( I \) is not complete for \( Q \). Then there must exist an \( I' \in \text{Ext}(I) \) such that \( Q(I) \neq Q(I') \). More specifically, since \( Q \) is monotonic, there must exist a valuation \( \nu \) of \( T_Q \) that draws values from \( I' \), such that \( Q(\nu(T_Q)) \notin Q(I) \). Define a valuation \( \nu' \) such that for every variable \( x \) in \( T_Q \), \( \nu'(x) \) is a distinct value in \( \text{New} \) if \( \nu(x) \) is not in \( \text{Adom} \), and \( \nu'(x) = \nu(x) \) otherwise. Observe the following. (a) The valuation \( \nu' \) draws values from \( \text{Adom} \). (b) \( Q(\nu'(T_Q)) \notin Q(I) \), by the choice of the values in \( \text{New} \) and the assumption that \( Q(\nu(T_Q)) \notin Q(I) \). (c) \( I \cup \nu'(T_Q), D_m \) \models V. This follows from the assumption that \( I' \in \text{Ext}(I), I \cup \nu'(T_Q) \subseteq I' \) and \( (I', D_m) \models V \). We then also have that \( (I \cup \nu'(T_Q), D_m) \models V \) since the CCs in \( V \) are defined in terms of monotonic CQ queries. Then again by the choice of the values in \( \text{New} \), \( (I \cup \nu'(T_Q), D_m) \models V \). Putting (a), (b) and (c) together, we have that \( I \cup \nu'(T_Q) \in \text{Mod}_{\text{Adom}}(I \cup T_Q) \) but \( Q(I) \neq Q(I \cup \nu'(T_Q)) \).

(2) We next show that \((T, \xi)\) is in RCDP*(\(Q, D_m, V)\) if and only if for each \( I \in \text{Mod}_{\text{Adom}}(T,
D_m, V), I is in Mod(T, D_m, V), First, assume that (T, ξ) is in RCQ*(Q, D_m, V). Then by the definition of strongly complete c-instances, all ground instances in Mod(T, D_m, V) are complete for Q relative to (D_m, V), including those in I ∈ Mod_{Adom}(T, D_m, V). Conversely, assume that (T, ξ) is not in RCQ*(Q, D_m, V). Then there exists a valuation µ of T such that I_1 = µ(T), I_1 ∈ Mod(T, D_m, V), and there exists I_2 ∈ Ext(I_1) such that Q(I_1) ⊆ Q(I_2). Then along the same lines as the argument given for (1), one can verify that there exist valuations µ' and ν' that draw values from Adom such that I_1' = µ'(T), I_1' ∈ Mod(T, D_m, V), I_2' = I_1' ∪ ν'(T_Q), I_2' ∈ Ext(I_1'), but Q(I_1') ⊆ Q(I_2'). The valuations µ' and ν' are constructed by leveraging the choice of the values in New. That is, there exists an I ∈ Mod_{Adom}(T, D_m, V) such that I is not complete for Q relative to (D_m, V).

With this characterization in place we now present a Σ_2^p algorithm for the complement of our problem: given (T, ξ), D_m, V and CQ query Q, it returns “yes” if there exists a database I ∈ Mod_{Adom}(T, D_m, V) and a tuple s such that s /∈ Q(I) but s ∈ Q(I') for some I' ∈ Mod_{Adom}(I ∪ T_Q), and it returns “no” otherwise. More specifically, the algorithm does the following:

(1) Guess a valuation µ of (T, ξ) on Adom, a valuation ν for T_Q taking values from Adom, and a tuple s of R_Q, where R_Q is the schema of the query result Q(D).
(2) Check:
   (a) whether µ(T) ∈ Mod(T, D_m, V); if so continue; and otherwise reject the current guess; this test can be done in coNP;
   (b) whether (µ(T) ∪ ν(T_Q)) ∈ Ext(µ(T), D_m, V); if so continue; and otherwise reject the current guess; this test can be done in coNP;
   (c) whether s /∈ Q(µ(T)); if so continue, and otherwise reject the current guess; this test can be done in coNP;
   (d) whether s ∈ Q(µ(T) ∪ ν(T_Q)); if so return “yes”, and otherwise reject the current guess; this test can be done in NP

The complexity of the algorithm is thus in Σ_2^p and hence, RCDP*(CQ) is in Π_2^p. We now verify the correctness of the algorithm. Clearly, the algorithm returns “yes” if a counterexample to the strong completeness of (T, ξ) for Q is found. Indeed, the counterexample consists of I = µ(T) and I' = I ∪ ν(T_Q), where µ and ν are the guesses that lead to a successful run of the algorithm. Conversely, we show that if (T, ξ) is incomplete for Q relative to (D_m, V), then the algorithm returns “yes”. Indeed, by Lemma 4.2, if (T, ξ) is incomplete, then there exist a valuation µ of T with Adom and a valuation ν of T_Q with Adom, such that I = µ(T), I' = I ∪ ν(T_Q), I ∈ Mod_{Adom}(T, D_m, V) and I' ∈ Mod_{Adom}(I ∪ ν(T_Q), D_m, V), but there exists a tuple s ∈ Q(I') and s /∈ Q(I). Such valuations µ and ν can indeed be guessed by the algorithm. That is, the algorithm is able to find a counterexample.

RCDP(UCQ). We next show that it is in Π_2^p to decide whether a c-instance is strongly complete for queries in UCQ. Consider a query Q in UCQ: Q_1 ∪ ... ∪ Q_k, where Q_i is a query in CQ for each i ∈ [1, k]. Consider master data D_m, a set V of CCs and a c-instance (T, ξ). We represent Q_i as a tableau query (T_i, u_i) for i ∈ [1, k]. We revise the notion of bounded databases for UCQ queries as follows: a ground instance I of R is said to be bounded by (D_m, V) if for each i ∈ [1, k] and each I' ∈ Mod_{Adom}(I ∪ T_i), Q(I) = Q(I').

Along the same lines as Lemma 4.2, we have the following characterization for strongly complete c-instances for UCQ queries.

**Lemma 4.3.** For every query Q in UCQ, master data D_m, any set V of CCs, and any c-instance (T, ξ), (T, ξ) is in RCQ*(Q, D_m, V) if and only if for each I ∈ Mod_{Adom}(T, D_m, V), I is bounded by (D_m, V).
PROOF. It suffices to show that for a UCQ query $Q = Q_1 \cup \cdots \cup Q_k$, (1) for each ground instance $I$ of $R$, $I$ is complete for $Q$ relative to $(D_m, V)$ if and only if it is bounded by $(D_m, V)$, i.e., for each $i \in [1, k]$ and each $I' \in \text{Mod}_{\text{Adom}}(I \cup_i T_i)$, $Q(I) = Q(I')$, where $(T_i, u_i)$ is the tableau representation of $Q_i$, and (2) the $c$-instance $(T, \xi)$ is in $\text{RCQ}^*(Q, D_m, V)$ if and only if for each $I \in \text{Mod}_{\text{Adom}}(T, D_m, V)$, $I$ is complete for $Q$ relative to $(D_m, V)$. These can be verified along the same lines as the proof of Lemma 4.2. \qed

With this characterization we extend the $\Sigma^P_2$ algorithm given above to UCQ queries. More specifically, the algorithm presented earlier only needs a minor modification: In Step 2, we additionally guess one of the component queries $Q_i$ in $Q$ and a valuation $\nu_i$ of $Q_i$'s tableau $T_i$; furthermore, steps 2(b) and 2(d) use $\nu_i(T_i)$ rather than $\nu(T)$. In other words, the algorithm tries to find a $Q_i$ and instances $I \in \text{Mod}_{\text{Adom}}(T, D_m, V)$ and $I' \in \text{Mod}_{\text{Adom}}(I \cup_i \nu_i(T_i), D_m, V)$ for which $Q(I) \subseteq Q(I')$. That is, the algorithm verifies whether there is an $I \in \text{Mod}_{\text{Adom}}(T, D_m, V)$ that is not bounded by $(D_m, V)$.

This modification does not affect the complexity of the algorithm and thus it remains in $\Sigma^P_2$. Indeed, steps 2(c) and 2(d) remain in coNP and NP, respectively, for UCQ queries. The correctness of the algorithm can be verified along the same lines as its counterpart for $\text{RCDP}^*(CQ)$, based on Lemma 4.2. This shows that it is in $\Pi^P_2$ time to determine whether a $c$-instance is strongly complete for a query in UCQ w.r.t. $(D_m, V)$.

$\text{RCDP}^*(\exists FO^+)$.

We show that the $\Pi^P_2$ algorithm given above can be extended to $\exists FO^+$-queries. A query $Q$ in $\exists FO^+$ is equivalent to a possibly exponentially long union of CQ queries. Therefore, an unfolding of the query will bring us beyond $\Pi^P_2$. However, we can avoid unfolding $Q$ by replacing the guess in Step 2 by obtaining a single CQ by guessing disjunctions in $Q$. As before, this modification does not affect the complexity as steps 2(c) and (d) remain in coNP and NP, respectively, for $\exists FO^+$-queries. Putting these together, we have a $\Pi^P_2$ algorithm for checking $\text{RCDP}^*(\exists FO^+)$.

This completes the proof of Theorem 4.1. \qed

Theorem 4.1 is verified for $c$-instances $T$ for which $\text{Mod}(T)$ is non-empty. We next show that the complexity bounds remain unchanged without assuming that $\text{Mod}(T)$ is non-empty. This obviously holds when $L_Q$ is FO or FP, for which $\text{RCDP}^*$ is undecidable. For CQ, UCQ and $\exists FO^+$, we need an extra step to check whether $\text{Mod}(T)$ is empty; if so, algorithm terminates with "yes", and otherwise it checks $\text{RCDP}^*$ in $\Pi^P_2$. By Proposition 3.3, the initial step is also in $\Pi^P_2$. Hence $\text{RCDP}^*$ remains in $\Pi^P_2$ for CQ, UCQ and $\exists FO^+$. In other words, the assumption has no impact on the complexity of $\text{RCDP}^*$.

4.2. The Relatively Complete Query Problem in the Strong Model

This problem is to determine whether a given query has a relatively complete database at all. For $\text{RCDP}^*(L_Q)$, we do not have to worry about missing values. Indeed, $\text{RCDP}^*(L_Q)$ for $c$-instances and its counterpart for ground instances coincide.

LEMMA 4.4. For every schema $R$, query $Q$, master data $D_m$, set $V$ of CCs and number $K$, there exists a $c$-instance $T$ of $R$ such that $|T| \leq K$ and $T \in \text{RCQ}^*(Q, D_m, V)$ if and only if there exists a ground instance $I$ of $R$ such that $|I| \leq K$ and $I \in \text{RCQ}^*(Q, D_m, V)$.

PROOF. First assume that there exists a $c$-instance $T \in \text{RCQ}(Q, D_m, V)$ with $|T| \leq K$. Then by the definition of strongly complete $c$-instances, $\text{Mod}(T, D_m, V) \neq \emptyset$ and moreover, for each instance $I \in \text{Mod}(T, D_m, V)$, $I$ is complete for $Q$ relative to $(D_m, V)$. Furthermore, $|I| \leq K$ since $I = \mu(T)$ for a valuation of $T$. Conversely, assume that there exists an instance $I$ in $\text{RCQ}^*(Q, D_m, V)$ with $|I| \leq K$. Then $I$ is also a $c$-instance itself, which is complete for $Q$ relative to $(D_m, V)$. \qed
As a consequence one only needs to consider RCQP⁺(L_Q) for ground instances. Nevertheless, for ground instances, the complexity bounds on RCQP⁺(L_Q) were left open in [Fan and Geerts 2009] when L_Q is FO or FP, and when CCs are expressed in CQ. Indeed, RCQP(L_Q) was shown undecidable in [Fan and Geerts 2009] by using CCs expressed as fixed FO or FP queries. Below we settle these cases.

**Theorem 4.5.** For c-instances, RCQP⁺(L_Q) is

— undecidable when L_Q is FO or FP; and
— NEXPTIME-complete when L_Q is CQ, UCQ or ∃FO⁺.

The complexity bounds remain unchanged when D_m and V are fixed.

**Proof.** In light of Lemma 4.4, it suffices to consider ground instances of R.

(1) When L_Q is FO. We show that RCQP⁺(FO) is undecidable by reduction from the satisfiability problem for FO, which is undecidable (cf. [Trakhtenbrot 1950; Abiteboul et al. 1995]). Given an FO query q, we construct master data D_m, a set V of CCs and an FO query Q such that q is satisfiable if and only if RCQP⁺(Q, D_m, V) is empty.

We now define D_m, V and Q. The reduction does not rely on master data, i.e., V and D_m are empty. To define Q, by Lemma 3.2, assume w.l.o.g. that q is defined over a relation schema R. We define another schema R′, where R′ extends R by adding an extra attribute A with an infinite domain. We define Q as the following query over R′:

\[ Q(I) = \begin{cases} \emptyset & \text{if } \forall a(q(\pi_R(a(I))) = \emptyset) \\ I & \text{otherwise.} \end{cases} \]

We show that q is satisfiable if and only if RCQP⁺(Q, D_m, V) is empty.

First, suppose that q is satisfiable, and let I be an instance of R such that q(I) ≠ ∅. For each instance I_1 of R′, define I_2 = I_1 ∪ (\{A = b\} × I) ∪ \{t\}, where b is a distinct constant not appearing in I_1, and t is a tuple not in I_1 and with t[A] ≠ b. Here t ensures I_1 ≠ I_2 even when I is empty. Then I_2 is a partially closed extension of I_1 but Q(I_1) ≠ Q(I_2). Indeed, Q(I_2) = I_2, but Q(I_1) ≠ I_2 no matter whether Q(I_1) = I_1 or Q(I_1) = ∅. Thus I_1 is not in RCQP⁺(Q, D_m, V), and hence RCQP⁺(Q, D_m, V) is empty.

Conversely, if q is not satisfiable then Q(I) = ∅ for every instance I of R′. Thus, RCQP⁺(Q, D_m, V) is not empty, since every instance of R′ is relatively complete for Q.

(2) When L_Q is FP. To show that RCQP⁺(FP) is undecidable, we first prove the undecidability of the following problem, from which we will give a reduction to RCQP⁺(FP).

The satisfiability problem for FP in the presence of FDs is to determine, given an FP query p defined on schema R and a set Θ of FDs defined on R, whether there exists an instance I of R such that I |= Θ and p(I) is non-empty. The undecidability of this problem was claimed in [Levy et al. 1993]. Below we provide a proof for a stronger result in that the set Θ of FDs can be assumed to be fixed.

**Lemma 4.6.** The satisfiability problem of FP is undecidable in the presence of a fixed set of FDs.

**Proof.** We show the undecidability by reduction from the emptiness problem for deterministic finite 2-head automata, which is known to be undecidable [Spielmann 2000]. Our proof closely follows the reduction presented in [Spielmann 2000, Theorem 3.3.1], which shows that the satisfiability of the existential fragment of transitive-closure logic, E+TC, is undecidable over a schema having at least two non-nullary relation schemas, one of them being a function symbol. Although E+TC allows the
negation of atomic expression as opposed to FP, the undecidability proof only uses a very restricted form of negation, which can be simulated using $\neq$ and a fixed set of FDs.

We start with a review of necessary definitions from [Spielmann 2000].

A deterministic finite 2-head automaton (2-head DFA for short) is a quintuple $A = (S, \Sigma, \Delta, s_0, s_{ace})$, consisting of a finite set of states $S$, an input alphabet $\Sigma = \{0, 1\}$, an initial state $s_0$, an accepting state $s_{ace}$, and a transition function $\Delta: S \times \Sigma \times \Sigma_r \rightarrow S \times \{0, +1\} \times \{0, +1\}$, where $\Sigma_r = \Sigma \cup \{\epsilon\}$.

A configuration of $A$ is a triple $(s, w_1, w_2) \in S \times \Sigma^* \times \Sigma^*$, representing that $A$ is in state $s$, and the first and second head of $A$ are positioned on the first symbol of $w_1$ and $w_2$, respectively. On an input string $w \in \Sigma^*$, $A$ starts from the initial configuration $(s_0, w, w)$; and the successor configuration is defined as usual.

A 2-head DFA $A$ accepts $w$ if it can reach a configuration $(s_{ace}, w_1, w_2)$ from the initial configuration for $w$; otherwise $A$ rejects $w$. The language accepted by $A$ is denoted by $L(A)$.

The emptiness problem for 2-head DFAs is to determine, given a 2-head DFA $A$, whether $L(A)$ is empty.

Given a 2-head DFA $A = (S, \Sigma, \delta, s_0, s_{ace})$, we define a schema $R$, an FP-query $\Pi$, and a fixed set $\Theta$ of FDs over $R$. We show that $L(A)$ is non-empty if and only if there exists an instance $I$ of $R$ such that (i) $I \models \Theta$ and (ii) $\Pi(I)$ is non-empty.

(a) The database schema $R$ consists of two relations $P(V, A)$ and $S(W, A_1, A_2)$. Intuitively, $P(V, A)$ and $S(W, A_1, A_2)$ are to store constant relations, encoding a word $w$ in $\Sigma^*$. More specifically, an instance $I = (I_P, I_S)$ of $R$ represents a string $w \in \Sigma^*$ such that (i) elements in $\sigma_{V=1}(I_P)$ represent the positions in $w$ where a 1 occurs; (ii) $\sigma_{V=0}(I_P)$ records those positions in $w$ that are 0; and (iii) $I_S$ encodes a successor relation over these positions in $w$ by $\pi_{A_1, A_2}(\sigma_{A_1 \neq A_2}(I_S)) \cup \pi_{A_1, A_2}(\sigma_{A_1 \neq A_2}(I_S)) \cup \pi_{A_1, A_2}(\sigma_{A_1 = A_2 \land W = 1}(I_S))$ in which the last part identifies the final position in the successor relation.

We denote $\sigma_{V=1}(P) \cup \sigma_{V=0}(P)$ by FP query $\Pi_P$; and $\pi_{A_1, A_2}(\sigma_{A_1 \neq A_2}(S)) \cup \pi_{A_1, A_2}(\sigma_{A_1 = A_2 \land W = 1}(S))$ by FP query $\Pi_S$.

(b) We will use three FDs to ensure that we only consider those instances of $P$ and $S$ that indeed represent a word in $\Sigma^*$, called well-formed instances of $P$ and $S$. An instance $I = (I_P, I_S)$ is well-formed if (i) $\sigma_{V=1}(I_P)$ and $\sigma_{V=0}(I_P)$ are disjoint (i.e., a string can only have one letter at each position); and $\pi_{A_1, A_2}(\sigma_{A_1 \neq A_2}(I_S)) \cup \pi_{A_1, A_2}(\sigma_{A_1 \neq A_2}(I_S))$ must (ii) be a function and (iii) contain a unique tuple of the form $(k, k)$ for some constant $k$ indicating the final position.

To assure this, we require the presence of a tuple $(1, k, k)$ in $I_S$. We additionally require that any instance $I_S$ of $S$ contains a tuple of the form $(w, 0, i)$, where 0 represents the initial position and $i$ is some constant. The latter two requirements will be assured by FP-queries $\Pi_{ini}$ and $\Pi_{fin}$, respectively, to be defined shortly.

More specifically, the conditions (i)–(iii) will be enforced by the following set $\Theta$ of FDs:

$$A \rightarrow V,$$ enforcing that for every instance $I' = (I'_P, I'_S)$ of $R$ such that $I' \models \Theta$, condition (i) is satisfied for $I'_P$.

$$A_1 \rightarrow A_2,$$ ensuring that $\pi_{A_1, A_2}(I'_S)$ encodes a function; hence condition (ii) is satisfied.

$$W \rightarrow A_1, A_2,$$ ensuring that there can be at most one tuple with its $W$-attribute set to 1 in $I_S$. As a result, $\pi_{A_1, A_2}(\sigma_{A_1 \neq A_2 \land W = 1}(I'_S))$ contains at most one tuple, and condition (iii) is satisfied.

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
In summary, any instance $I' = (I'_p, I'_s)$ of $\mathcal{R}$ that satisfies $\Theta$ is well-formed, with the exception that we still need to check for the existence of an initial and a final position in the instance $I'_s$ of $S$ in $I'$.

(c) Before we define the query $\Pi$, we show, following [Spielmann 2000], how the non-emptiness of $L(A)$ can be expressed in terms of an E+TC-formula over $\mathcal{R}$. Consider a transition $\delta \in \Delta$ of the form $\delta = (s, i_{n1}, i_{n2}) \rightarrow (s', \text{move}_1, \text{move}_2)$. Such a transition can be encoded by means of the conjunctive query

$$\varphi_\delta(x, y, z, x', y', z') = (x = s \land x' = s' \land \alpha_1(y) \land \alpha_2(z) \land \beta_1(y, y') \land \beta_2(z, z')).$$

Intuitively, $\alpha_1(y)$ is to represent the position of $y$ based on the value of $i_{n1}$; similarly for $\alpha_2(z)$ and $i_{n2}$; and $\beta_1(y, y')$ is to decide whether $y$ and $y'$ are consecutive positions or not. More specifically,

\begin{itemize}
  \item $- \alpha_1(y) = \exists y' (\Pi_S(y, y') \land y \neq y' \land \Pi_P(1, y))$ if $i_{n1} = 1$;
  \item $- \alpha_1(y) = \exists y' (\Pi_S(y, y') \land y \neq y' \land \Pi_P(0, y))$ if $i_{n1} = 0$; and
  \item $- \alpha_1(y) = \Pi_S(y, y)$ if $i_{n1} = c$;
\end{itemize}

similarly for $\alpha_2(z)$. Furthermore,

\begin{itemize}
  \item $- \beta_1(y, y') = \Pi_S(y, y')$ if $\text{move}_1 = 1$ and
  \item $- \beta_1(y, y') = (y = y')$ if $\text{move}_1 = 0$;
\end{itemize}

similarly for $\beta_2(z, z')$. That is, $\alpha_1(y)$ enforces $y$ to be a position in the string coded by $\Pi_P(1, y)$ or $\Pi_P(0, y)$ that has a successor, unless $y$ is the final position, where $\alpha_1(y)$ demands $\Pi_S(y, y)$; similarly for $\alpha_2(z)$. Moreover, $\beta_1(y, y')$ ensures that $y$ and $y'$ are consecutive positions when $A$ makes a move (with head 1) and $y = y'$ otherwise; similarly for $\beta_1(z, z')$. Then, following [Spielmann 2000] one can show that $\Phi = \exists y_1 y_2 [\mathcal{T}_C(x, y_1, x, y_2, z, z', z') \bigvee_{\delta \in \Delta} \varphi_\delta](s_0, 0, 0, s_{acc}, y_1, y_2)$ is satisfiable if and only if $L(A) \neq \emptyset$.

Clearly, we can compute $\Phi$ using a query $\Pi_1$ in FP. Recall that we still need to assure the existence of an initial and a final position in well-formed instances of $I'_s$. The final FP-query $\Pi$ is therefore defined as $\Pi_\Phi \land \Pi_{ini} \land \Pi_{fin}$, where $\Pi_{ini} = \exists w x \exists i \exists S(w, 0, x)$ and $\Pi_{fin} = \exists x \exists S(1, x, x)$.

This concludes the construction of $\mathcal{R}$, $\Pi$ and $\Theta$. One can verify that $L(A)$ is non-empty if and only if there exists an instance $I$ of $\mathcal{R}$ such that $I \models \Theta$ and $\Pi(I) \neq \emptyset$. Note that the set $\Theta$ of FDs is fixed, independent of the 2-head DFA $L(A)$. \qed

In light of Lemma 4.6, we show that RCPQ*(FP) is undecidable by reduction from the satisfiability problem for FP in the presence of FDs. Given an FP query $p(\vec{z})$ and a fixed set $\Theta$ of FDs, we construct a database schema $\mathcal{R}'$, master data $D_m$, a set $V$ of CCs, and a FP query $Q$, such that $p$ is satisfiable in the presence of $\Theta$ if and only if RCPQ*(Q, D_m, V) is empty.

Suppose that $p$ and $\Theta$ are defined over a database schema $\mathcal{R}$. By Lemma 3.2, we assume w.l.o.g. that $\mathcal{R}$ consists of a single relation schema $R(A_1, \ldots, A_m)$.

(a) We define a database schema $\mathcal{R}' = (R', E)$, where $R'(G, A_1, \ldots, A_m)$ extends $R$ by adding a new attribute $G$ with an infinite domain, and $E(C)$ is a unary relation that consists of a single attribute $C$ with an infinite domain.

(b) We define CCs as follows. Note that for each FD $X \rightarrow A$ in $\Theta$, $(G, X) \rightarrow A$ is an FD defined over $R'$. Denote by $\mathcal{O}'$ the set of all such FDs over $R'$ deduced from FDs in $\Theta$. For each FD $(G, X) \rightarrow A$ in $\Theta$, we express it as a CC: $p_v(R') \subseteq \emptyset$, where

$$p_v(g) = \exists \vec{x}, y, y', z_1, z_2 (R'(g, \vec{x}, y, z_1) \land R'(g, \vec{x}, y', z_2) \land y \neq y'),$$

$\vec{x}, y$ and $z_1$ correspond to attributes $X$, $A$ and $R' \setminus \{X \cup \{A\}\}$, respectively; similarly for $\vec{y}, y'$ and $z_2$. That is, $p_v(R')$ extracts tuples that violate the FD $(G, X) \rightarrow A$. We
define $V$ to be the set of all CCs constructed from FDs in $\Theta'$ as above. Intuitively, we group tuples of $R'$ by the attribute $G$, such that FDs in $\Theta$ are imposed on each group individually. By Lemma 4.6, $\Theta$ is fixed, and hence, so is $V$.
(c) The master data $D_m$ is assumed to be an empty relation $\emptyset$.
(d) We define $Q$ as follows. We first construct a query $p'$ by substituting $R'(g, \vec{y})$ for each occurrence of $R(\vec{y})$ in each rule of $p$, where $g$ is a variable corresponding to attribute $G$, and is shared across all the rules in $p'$. One can verify that the following are equivalent:

— there exists an instance $I$ of $R$ such that $I \models \Theta$ and $p(I)$ is non-empty,
— there exists an instance $I'$ of $R'$ such that there exists $g \in \text{dom}(G)$, $I'_g \models \Theta$ and $p(I'_g)$ is non-empty, where $I'_g$ is the subset of $I'$ consisting of tuples $t$ with $t[G] = g$.

We define $Q(x) : -E(x), p'(g, \vec{y})$, i.e., $Q(I', E)$ returns the $E$ relation if there exists $g$ such that $I'_g \models \Theta$ and $p(I'_g)$ is non-empty.

We next show that $p$ is satisfiable in the presence of $\Theta$ if and only if $\text{RCQ}^*(Q, D_m, V)$ is empty.

$\Rightarrow$ First assume that $p$ is not satisfiable in the presence of $\Theta$, i.e., there exists no instance $I$ of $R$ such that $I \models \Theta$ and $p(I)$ is non-empty. Then $(\emptyset, \emptyset)$ is in $\text{RCQ}^*(Q, D_m, V)$, i.e., the empty instance of $R'$ and the empty instance of $E$ make a database that is complete for $Q$ relative to $(D_m, V)$. Indeed, for all instances $(I', E)$, if there exists $g \in \text{dom}(G)$ such that $I'_g \models \Theta$, then $p(I'_g)$ is empty, and hence, $Q(I', E) = \emptyset$.

$\Leftarrow$ Conversely, assume that $p$ is satisfiable. Then there exists an instance $I$ of $R$ such that $I \models \Theta$ and $p(I)$ is non-empty. We next show that $\text{RCQ}^*(Q, D_m, V)$ is empty. That is, we need to prove that for each instance $(I', E)$ of $R'(I', E) \notin \text{RCQ}^*(Q, D_m, V)$ when $((I', E), (I', E)) \models V$. We construct an extension $I''$ of $I'$ such that for each tuple $t$ in $I$, $(g, t)$ is in $I''$, for a constant $g \in \text{dom}(G)$ that does not appear in the $G$ column of $I'$. Let $E'$ be an extension of $E$. Obviously $(I'', E')$ is partially closed since $(I', E)$ is partially closed, $I''_g \models \Theta$, and the CCs in $V$ apply to tuples with the same $G$-attribute value.

Furthermore, $p'(I'')$ is non-empty. Hence $(I'', E')$ is a partially closed extension of $(I', E)$. However, $Q(I'', E') = E' \neq E = Q(I', E)$. Hence $(I', E)$ is not in $\text{RCQ}^*(Q, D_m, V)$.

(3) When $\mathcal{L}_Q$ is CQ, UCQ or $\exists F O^+$. It is known [Fan and Geerts 2010b] that $\text{RCQP}^*(\exists F O^+)$ is in NEXPTIME and that $\text{RCQP}^*(\text{CQ})$ is NEXPTIME-hard, when $D_m$ and $V$ are fixed. From this and Lemma 4.4 it follows that $\text{RCQP}(\mathcal{L}_Q)$ is NEXPTIME-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists F O^+$.

Note that in the proofs above, only fixed master data $D_m$ and fixed CCs are used. Hence the complexity bounds remain intact when $V$ and $D_m$ are fixed.

4.3. The Minimality Problem in the Strong Model

This problem is to decide whether a database is relatively complete and moreover, does not contain excessive data. The lemma below tells us how to check this when ground instances are concerned.

**Lemma 4.7.** For every ground instance $I$, query $Q$, master data $D_m$, and set $V$ of CCs, (a) if $(I, D_m) \models V$ then for every $I' \subset I$, $(I', D_m) \models V$, and (b) if $I$ is in $\text{RCQ}^*(Q, D_m, V)$, then $I$ is not minimal if and only if there exists a tuple $t \in I$ such that $I \setminus \{t\} \notin \text{RCQ}^*(Q, D_m, V)$.

**Proof.** Consider query $Q$, master data $D_m$, and a set $V$ of CCs. By Lemma 3.2, we assume w.l.o.g. that $Q$ is defined over a single relation schema $R$. Given an instance $I$ of $R$, we show the following.
(a) If $(I, D_m) \models V$ then for every $I' \subset I$, $(I', D_m) \models V$. We show that for every $\phi \in V$,
(I′, Dm) |= φ. Let ϕ be q(R) ⊆ p(Dm), where q is a CQ query. Since I′ ⊆ I, q(I′) ⊆ q(I) because CQ queries are monotonic. By (I, Dm) |= V, q(I) ⊆ p(Dm); and hence, q(I′) ⊆ p(Dm), i.e., (I′, Dm) |= φ.

(b) Suppose that I is in RCQ*(Q, Dm, V). Then I is not minimal if and only if there exists a tuple t ∈ I such that I \ {t} ∈ RCQ*(Q, Dm, V). To see this, first assume that I \ {t} ∈ RCQ*(Q, Dm, V). Then obviously I is not minimal by the definition of minimal instances. Conversely, suppose that I is not minimal, i.e., there exists I ⊆ I such that for each I2 ∈ Ext(I1, Dm, V), Q(I1) = Q(I2). Note that I ∈ Ext(I1, Dm, V). Then there must exist I2 = I \ {t} for some t ∈ I such that I1 ⊆ I2 and Q(I1) = Q(I2). By (a) above, (I2, Dm) |= V. In addition, for all I′ ∈ Ext(I2, Dm, V), I′ is also in Ext(I1, Dm, V), and hence, Q(I′) = Q(I1) = Q(I2). Thus I2 is in RCQ*(Q, Dm, V). □

Capitalizing on this lemma, below we provide complexity bounds on MINP*(LQ). Here the presence of missing values again makes the problem a little harder: MINP*(CQ) is D2-complete for ground instances, but it is Π2p-complete for c-instances. Here D2 is the class of languages recognized by oracle machines that make a call to a Σ2p oracle and a call to a Π2p oracle. That is, L is in D2 if there exist languages L1 ∈ Σ2 and L2 ∈ Π2 such that L = L1 ∩ L2 [Woolridge and Dunne 2004].

Theorem 4.8. When LQ is FO or FP, MINP*(LQ) is undecidable both for ground instances and for c-instances. When LQ is CQ, UCQ or ∃FO+, MINP*(LQ) is

— Π2p-complete for c-instances, and
— D2-complete for ground instances.

The complexity is unchanged when Dm and V are fixed.

Proof. We first show that MINP*(LQ) is undecidable when LQ is either FO or FP. We then verify that when LQ is CQ, UCQ or ∃FO+, MINP*(LQ) is Π2p-complete for c-instances but is D2-complete for ground instances. In the proofs for undecidability and lower bounds to be given below, we use fixed Dm and V.

(1) When LQ is FO or FP. To show that MINP*(LQ) is undecidable when LQ is either FO or FP, it suffices to show that these problems are undecidable for ground instances, since ground instances are c-instances themselves. In addition, it suffices to show it is undecidable to determine, given a query Q, master data Dm and a set of V of CCs, whether a special instance I0 is in RCQ*(Q, Dm, V), where I0 is the empty instance of the schema over which Q is defined. Indeed, if I0 is in RCQ*(Q, Dm, V), then it is a minimal instance complete for Q relative to (Dm, V).

MINP*(FO). We verify the undecidability of MINP*(FO) by reduction from the satisfiability of FO queries. Given an FO query q, we assume w.l.o.g. by Lemma 3.2 that q is defined over a single relation R. Consider the FO query defined in the proof of Theorem 4.5 (1), and let Dm and V both be empty. It has been shown there that if q is not satisfiable, then I0 is in RCQ*(Q, Dm, V). Conversely, if q is satisfiable, then RCQ*(Q, Dm, V) is empty, and hence, I0 is not in RCQ*(Q, Dm, V). Thus q is satisfiable if and only if I0 is minimal in RCQ*(Q, Dm, V).

MINP*(FP). Along the same lines as the proof for MINP*(FO), it suffices to show that given Q in FP, Dm and V, it is undecidable to determine whether the special instance I0 is in RCQ*(Q, Dm, V). This has already been verified by the proof of Theorem 4.5 (2). Indeed, it has been shown that deciding non-emptiness of RCQ*(Q, Dm, V) is undecidable. In particular, RCQ*(Q, Dm, V) ≠ ∅ iff I0 ∈ RCQ*(Q, Dm, V).
(2) When $\mathcal{L}_Q$ is CQ, UCQ or $\exists\forall\text{FO}^+$. We prove that $\text{MINP}^p(\mathcal{L}_Q)$ is $\Pi^p_3$-complete for $c$-instances and $\mathcal{D}^p_3$-complete for ground instances.

(2.1) For $c$-instances. To show that $\text{MINP}^p(\mathcal{L}_Q)$ is $\Pi^p_3$-complete for $c$-instances when $\mathcal{L}_Q$ is CQ, UCQ or $\exists\forall\text{FO}^+$, it suffices to verify that $\text{MINP}^p(\text{CQ})$ is $\Pi^p_3$-hard and that $\text{MINP}^p(\exists\forall\text{FO}^+)$ is in $\Pi^p_3$.

Lower bound. We show that $\text{MINP}^p(\text{CQ})$ is $\Pi^p_3$-hard by reduction from the complement of the $\exists\forall\exists\forall\text{3SAT}$ problem, which is known to be $\Sigma^p_3$-complete (cf. [Papadimitriou 1994]). The $\exists\forall\exists\forall\text{3SAT}$ problem is to determine, given a sentence $\varphi = \exists X \forall Y \exists \psi$, whether or not $\varphi$ is true. Here $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $Z = \{z_1, \ldots, z_k\}$ and $\psi$ is an instance of $\text{3SAT}$.

Given an instance $\varphi = \exists X \forall Y \exists \psi$ of the $\exists\forall\exists\forall\text{3SAT}$ problem, we define a database schema $\mathcal{R}$, a $c$-instance $T$ of $\mathcal{R}$, master data $D_m$, a set $V$ of CCs and a query $Q$ in CQ, such that $\varphi$ is true if and only if $T$ is not a minimal $c$-instance in $R\text{CQ}^R(Q, D_m, V)$.

(a) The database schema $\mathcal{R}$ consists of six relation schemas: $R_{(0,1)}(A)$, $R_{(1,1)}(A, \bar{A})$, $R_{A}(A_1, A_2, B)$, $R_{\lambda}(A_1, A_2, B)$, $R_X(id, X)$ and $R_s(W)$, where $R_{(0,1)}$, $R_{(1,1)}$, $R_{A}$ and $R_s$ are the same as their counterparts in the proof of Proposition 3.3. The relation $R_X(id, X)$ is to encode a truth assignment for variables in $X$; $R_s(W)$ is used to inspect query answers, as will become clear shortly.

(b) We construct a $c$-instance $T = (I_{(0,1)}, I_{(1,1)}, I_{A}, T_X, I_s)$, in which $I_{(0,1)}$, $I_{(1,1)}$, $I_A$ and $I_s$ are ground relations as shown in Figure 2, while $T_X = \{(1, x_1), (2, x_2), \ldots, (n, x_n)\}$ is a $c$-table, consisting of the variables in $X$, without any local condition. Finally, $I_s$ consists of two tuples $(0)$ and $(1)$. Intuitively, we use $T$ to encode basic Boolean operations, truth assignments for variables of $X$ and possible truth values of $\psi$ in the reduction.

(c) Master data $D_m$ is specified by five relation schemas: $R^m_{(0,1)} = R_{(0,1)}$, $R^m_{(1,1)} = R_{(1,1)}$, $R^m_{A} = R_{A}$ and $R^m_{\lambda}(X)$. The master data instances consist of $I^m_{(0,1)} = I_{(0,1)}$, $I^m_{(1,1)} = I_{(1,1)}$, $I^m_A = I_A$, and $I^m_\lambda = \emptyset$.

(d) The set $V$ consists of the following CCs:

- $R_{(0,1)} \subseteq R^m_{(0,1)}$, $R_{(1,1)} \subseteq R^m_{(1,1)}$, $R_{A} \subseteq R^m_{A}$, $R_{\lambda} \subseteq R^m_{\lambda}$, and $R_s \subseteq R^m_{(0,1)}$;
- $\exists id R_X(id, x) \subseteq R^m_{(0,1)}(x)$; and
- $\neg q_{ad}(x) \subseteq R^m_{\emptyset}(x)$, where $q_{ad}(x) = \exists y, y' R_X(x, y) \land R_X(x, y') \land (y \neq y')$. This is to ensure that id is a key for $R_X$.

The last two CCs given above ensure that each instance of $R_X$ is indeed a truth assignment of variables in $X$.

(e) We next define the query $Q$, such that $\varphi$ is true if and only if (i) there exists a ground instance $I$ of $T$ that encodes a truth assignment $\mu_X$ of $X$ variables by an instance of $T_X$ in $I$, and (ii) $Q(I)$ returns tuples representing all truth assignments $\mu_{Y}$ of $Y$ variables when $\psi$ is true under $(\mu_X, \mu_Y, \mu_Z)$; here $\mu_Z$ is a truth assignment of $Z$ variables, which is encoded by a tuple returned by a sub-query of $Q$ on $I$ (if it exists). More specifically, the query $Q$ is defined as follows,

$$Q(\bar{y}) = \exists \bar{x}, \bar{z} (Q_X(\bar{x}) \land Q_Y(\bar{y}) \land Q_Z(\bar{z}) \land Q_{\psi}(\bar{x}, \bar{y}, \bar{z}, w) \land R_s(w) \land Q_{aa}),$$

where $Q_X$ is a CQ query $Q_X(x_1, \ldots, x_n) = \bigwedge_{i \in [1, n]} R_X(i, x_i)$, i.e., it selects from $R_X$ the truth assignments for $X$. The query $Q_Y(\bar{y}) = R_{(0,1)}(y_1) \land \cdots \land R_{(0,1)}(y_m)$ constructs all possible truth assignments of variables in $Y$; similarly for $Q_Z(\bar{z})$ and $Z$. Given a truth assignment $(\mu_X, \mu_Y, \mu_Z)$ of $(X, Y, Z)$, the subquery $Q_{\psi}(\mu_X, \mu_Y, \mu_Z, w)$ is to evaluate $\psi(\mu_X, \mu_Y, \mu_Z)$, and it records its truth value in $w$, which is either 0 or 1. Obviously,
$Q_\psi$ can be defined in CQ by leveraging relations $I_\rightarrow$, $I_\forall$ and $I_\land$. The query $Q_{\text{all}}$ is to ensure that all the tuples in $I_{(0,1)}, I_\rightarrow, I_\forall$ and tuple (1) in $I_s$ are in place. More specifically, it is defined as

$$Q_{\text{all}} = Q_{(0,1)} \land Q_\rightarrow \land Q_\forall \land Q_\land \land Q_s,$$

where $Q_\psi = R_\forall(0,0,0) \land R_\forall(0,1,1) \land R_\forall(1,0,1) \land R_\forall(1,1,1)$ asserts that the removal of any of the four tuples in $I_\forall$ makes $Q(\vec{y})$ empty; similarly for $Q_{(0,1)}$, $Q_\rightarrow$, and $Q_\land$. In addition, $Q_s = R_s(1)$, asserting that the removal of (1) makes $Q(\vec{y})$ empty. Intuitively, query $Q$ returns all tuples encoding truth assignments $\mu_Y$ of $Y$ such that for the truth assignment $\mu_X$ of $X$ encoded by $T_X$, $\exists Z \psi(\mu_X, \mu_Y, Z)$ evaluates to a truth value in $I_s$.

We show that $\varphi$ is false if and only if $T$ is a minimal $c$-instance in RCQ$^c(Q, D_m, V)$, i.e., for each ground instance $I$ of $T$, $I$ is minimal in RCQ$^c(Q, D_m, V)$. The argument is based on the following observations: for every ground instance $I = (I_{(0,1)}, I_\rightarrow, I_\forall, I_\land, I_s)$ of $T$, (i) $I$ has no extensions by the definition of $V$, i.e., $V$ enforces an upper limit on the potential valuations of $T$; (ii) removing any tuple from $I_X$, $I_{(0,1)}$, $I_\forall$, or $I_s$, or removing tuple (1) from $I_s$ would make $Q$ empty, by the definitions of $Q$ and $Q_{\text{all}}$; i.e., $Q_{\text{all}}$ imposes a lower limit on which tuples must exist.

First assume that $\varphi$ is false. Then for each truth assignment $\mu_X$ of $X$, there exists a truth assignment $\mu_Y$ of $Y$ such that $\exists Z \psi(\mu_X, \mu_Y, Z)$ is false. It is easy to see that for all ground instances $I = (I_{(0,1)}, I_\rightarrow, I_\forall, I_\land, I_s) \in \text{Mod}(T)$ of $T$, $Q(I)$ returns all truth assignments of $Y$ since $I_s$ consists of (0) and (1). As argued in (ii) above, there exists no extension to $I$, and thus $I \in \text{RCQ}^c(Q, D_m, V)$. We next show that $I$ is minimal. As argued in (ii) above, we only need to consider the instance $I' = (I_{(0,1)}, I_\rightarrow, I_\forall, I_\land, I_s', I_1')$, where $I_1' = \{(1)\}$. Obviously, $Q(I')$ does not contain all truth assignments of $Y$ since $\varphi$ is false, making it different from $Q(I)$. Hence $T$ is a minimal $c$-instance.

Conversely, suppose that $\varphi$ is true. Then there exists a truth assignment $\mu_X^0$ of $X$ such that for every truth assignment $\mu_Y$ of $Y$, $\exists Z \psi(\mu_X^0, \mu_Y, Z)$ is true. We show that there exists a ground instance in $\text{Mod}(T, D_m, V)$ that is not minimal. Let $I_X^0$ be a ground instance of $T_X$ that agrees with $\mu_X^0$, and $I_\forall^0 = (I_{(0,1)}, I_\rightarrow, I_\forall, I_\land, I_s^0)$. As before, $Q(I_\forall^0)$ consists of all truth assignments of $Y$. However, $I_\forall^0$ is not minimal. Indeed, consider $I' = (I_{(0,1)}, I_\rightarrow, I_\forall, I_\land, I_s', I_1')$ with $I_1' = \{(1)\}$. Then, since $\varphi$ is true and $I_X^0$ encodes $\mu_X^0$, $Q(I')$ consists again of all truth assignments of $Y$. Hence, $I_\forall^0$ is not minimal and thus, $T$ is not a minimal $c$-instance in RCQ$^c(Q, D_m, V)$.

Upper bound. We next show that MINP$^c(\exists \text{FO}$) is in $\Pi_3^c$ for $c$-instances. The proof makes use of Lemma 4.7 and the $\Sigma_2^c$ algorithm [Fan and Geerts 2009] for checking whether a ground instance is not in RCQ$^c(Q, D_m, V)$.

We give an $\Sigma_2^c$ algorithm for the complement of MINP$^c(\exists \text{FO}$): given a $c$-instance $T_s$ of $Q$ in $\exists \text{FO}$, master data $D_m$ and a set $V$ of CCs, the algorithm returns “yes” if and only if $T$ is a minimal $c$-instance in RCQ$^c(Q, D_m, V)$. By Lemma 3.2, we assume w.l.o.g. that $T_s$ consists of a single $c$-table $T$. Assume that $T_s$ consists of $k$ tuples $\tau_1, \ldots, \tau_k$. Recall the notion of Adom given in the proof of Theorem 4.1. The algorithm works as follows.$$

\begin{enumerate}
\item Guess a valuation $\mu$ of $(T, \xi)$ with Adom and guess an index $i \in [1, k]$. Let $I = \mu(T)$.
\item Test the following:
\begin{enumerate}
\item whether $I \not\in \text{Mod}(T, D_m, V)$; if not, continue; otherwise reject the current guess;
\item whether $I$ is a complete ground instance; if so, return “yes”;
\item whether $\mu(\tau_i) \in I$ and $I \setminus \mu(\tau_i)$ is in RCQ$^c(Q, D_m, V)$. If so, return “yes”; otherwise reject the current guess.
\end{enumerate}
\end{enumerate}
The algorithm is in $\Sigma_2^p$. It invokes an NP oracle to check whether an instance is in $\text{Mod}(T, D_m, V)$ in step (2a), invokes an $\Sigma_2^p$ oracle to check whether $I \not\in \text{RCQ}^*(Q, D_m, V)$ in step (2b), and an $\Pi_2$ oracle to check whether $I \setminus \{\tau_i\} \in \text{RCQ}^*(Q, D_m, B)$ (see the proof of Theorem 4.1). Hence the algorithm is in NP, i.e., in $\Sigma_2^p$.

The algorithm returns "yes" when a counterexample is found that dispels $T$ as a minimal $c$-instance in $\text{RCQ}^*(Q, D_m, V)$. Conversely, suppose that $T$ is not a minimal $c$-instance in $\text{RCQ}^*(Q, D_m, V)$. Then along the same lines as the proofs of Lemmas 4.2 and 4.3 and the upper bound proof of Theorem 4.1 for $\text{RCDP}^*(\exists \forall \Omega^*)$, one can show that there must be a ground instance $I \in \text{Mod}_{\text{dom}}(T, D_m, V)$ such that $I$ is not a minimal ground instance in $\text{RCQ}^*(Q, D_m, V)$, i.e., either $I$ is not in $\text{RCQ}^*(Q, D_m, V)$, or $I$ is in $\text{RCQ}^*(Q, D_m, V)$ but there exists $I' \subseteq I$ such that $I'$ is also in $\text{RCQ}^*(Q, D_m, V)$. The former is checked by the $\Sigma_2^p$ oracle in step (2b), the latter is inspected by step (2)(c) of the algorithm above, which tests whether there exists a tuple $t = (\mu, \tau_i)$ in $I$ such that $I \setminus \{t\}$ is in $\text{RCQ}^*(Q, D_m, V)$. This suffices by Lemma 4.7. Hence the algorithm is able to find a counterexample $I$ and hence, returns "yes".

(2.2) For ground instances, we show that $\text{MINP}(\mathcal{L}_Q)$ is $D_2^p$-complete, when $\mathcal{L}_Q$ is CQ, UCQ or $\exists \forall \Omega^*$. It suffices to prove that in this setting, $\text{MINP}(\mathcal{Q})$ is $D_2^p$-hard and that $\text{MINP}(\exists \forall \Omega^*)$ is in $D_2^p$.

**Lower bound.** We first show that $\text{MINP}(\mathcal{Q})$ is $D_2^p$-hard by reduction from the $\exists \forall \exists^* \text{DNF} \land \exists \forall \exists^* \text{CNF}$ problem, which is $D_2^p$-complete [Wooldridge and Dunne 2004]. An instance of $\exists \forall \exists^* \text{DNF} \land \exists \forall \exists^* \text{CNF}$ is a pair of $\forall \exists^* \text{SAT}$ instances $\varphi_1 = \forall X_1 \exists Y_1 X_1 Y_1$ and $\varphi_2 = \forall X_2 \exists Y_2 X_2 Y_2$. It is to decide whether $\varphi_1$ is true and $\varphi_2$ is false. Given $(\varphi_1, \varphi_2)$, we define $\mathcal{R}, I, D_m, V$ and $Q$, such that $\varphi_1$ is true and $\varphi_2$ is false if and only if $I$ is a minimal instance in $\text{RCQ}^*(Q, D_m, V)$.

(a) The database schema $\mathcal{R}$ consists of six relation schemas: $R_{(0,1)}(A)$, $R_{(0,2)}(A, \bar{A})$, $R_{(1,2)}(A_1, A_2, B)$ and $R_{(2,1)}(A_1, A_2, B)$, the same as their counterparts in the proof of Proposition 3.3, as well as $R_1(W_1)$ and $R_2(W_2)$ to encode relations that will be used to select appropriate truth values, as will be detailed below.

(b) We construct a ground instance $I = (I_{(0,1)}, I_{(1,2)}, I_{(2,1)}, I_{(1,2)})$, in which $I_{(0,1)}$, $I_{(1,2)}$, and $I_{(2,1)}$ are ground relations as shown in Figure 2, $I_2 = \{(1,1)\}$ and $I_2 = \{(0,1)\}$.

(c) Master data $D_m$ is specified by six relation schemas: $R_{(0,1)}^{m} = R_{(0,1)}$, $R_{(0,2)}^{m} = R_{(0,2)}$, $R_{(1,2)}^{m} = R_{(1,2)}$, $R_{(2,1)}^{m} = R_{(2,1)}$, and $R_2^{m} = R_2$. The master data instance consists of $I_{(0,1)} = I_{(0,1)}$, $I_{(1,2)} = I_{(1,2)}$, $I_{(2,1)} = I_{(2,1)}$, and $I_2 = \{(0,1)\}$.

(d) The set $V$ consists of the following CCs: $R_{(0,1)} \subseteq R_{(0,1)}^{m}$, $R_{(1,2)} \subseteq R_{(1,2)}^{m}$, $R_{(2,1)} \subseteq R_{(2,1)}^{m}$, and $R_2 \subseteq R_2^{m}$.

(e) The CQ query $Q(x_1, x_2)$ is defined as $Q_1(x_1) \land Q_2(x_2) \land Q_{\text{all}}$, where $x_1$ and $x_2$ correspond to the $X$-variables in $\varphi_1$ and $\varphi_2$, respectively. The query $Q_{\text{all}}$ is used to ensure that (i) all tuples in $I_0$, $I_1$, $I_2$ and $I_3$ are present; and (ii) that $I_1$ and $I_2$ contain $(1)$. Otherwise, $Q_{\text{all}}$ returns false (empty set). The queries $Q_i(x_i)$ for $i = 1, 2$ are defined as

\[
Q_i(x_i) = \exists \bar{y}_i, w_i (Q_{X_i}(x_i) \land Q_{Y_i}(\bar{y}_i) \land Q_{\psi_i}(x_i, \bar{y}_i, w_i) \land R_i(w_i)),
\]

where $Q_{X_i}$ (resp. $Q_{Y_i}$) generates all truth assignments for $X_i$ (resp. $Y_i$) by means of Cartesian products of $R_{0i1}$; and $Q_{\psi_i}$ is a CQ query encoding $\psi_i$ by leveraging relations $I_1$, $I_2$, and $I_3$. More specifically, for given truth assignments $\mu_{X_i}$ and $\mu_{Y_i}$ of $X_i$ and $Y_i$, respectively, $Q_{\psi_i}(\mu_{X_i}, \mu_{Y_i}, 0)$ is true if $\psi_i(\mu_{X_i}, \mu_{Y_i})$ is false, and $Q_{\psi_i}(\mu_{X_i}, \mu_{Y_i}, 1)$ is true if $\psi_i(\mu_{X_i}, \mu_{Y_i})$ is true. The final conjunct in $Q_i$ controls what kind of truth values are returned. We consider the following cases: (a) If $I_1 = \{(1)\}$ then $Q_1(x_1)$ returns all truth
Suppose that $\varphi_1$ is true and $\varphi_2$ is false. Observe that $Q(I) = F_{X_1} \times F_{X_2}$, where $F_{X_1}$ consists of all possible truth assignments for $X_1$. Indeed, $Q_1(I) = F_{X_1}$ because $\varphi_1$ is true, whereas $Q_2(I) = F_{X_2}$ because $I_2$ consists of both (0) and (1). Furthermore, observe that $I$ is also complete. Indeed, the only possible extension of $I$ is $I' = (I_{(0,1)}, I_1, I_2)$ with $I'_1 = \{0\}$. Clearly, $Q(I') = Q(I)$ since $Q(I)$ already generates the largest possible query result. That is, $I \in \text{RCQ}^*(Q, D_m, V)$. We next show that $I$ is also minimal. Indeed, consider the unique extension $I''$ of $I$ described earlier. Clearly, $Q(I'') = F_{X_1} \times F_{X_2}$. However, since $\varphi_2$ is false and $I_1$ only contains (1), $Q_1(I'')$ will not include at least one truth assignment of $X_1$. Hence, $Q(I) \subseteq Q(I'')$. For case (ii), since $\varphi_1$ and $\varphi_2$ are both true, $Q(I) = F_{X_1} \times F_{X_2}$ and thus $Q(I'') = Q(I)$ since no more result tuples can be added. That is, $I$ is in $\text{RCQ}^*(Q, D_m, V)$. We show that $I$ is minimal. Indeed, consider the sub-instance $I''$ described earlier with $I''_1 = \{(1)\}$. We claim that $I''$ is in $\text{RCQ}^*(Q, D_m, V)$. To see this, observe that the only extensions of $I''$ are $I$, $I'$ and $I''' = (I_{(0,1)}, I_1, I_2)$. Since $\varphi_1$ is true, adding (0) to $I_1$ (as done in the extensions $I$ and $I''$) does not affect $Q_1(I'')$. Similarly, adding (0) to $I''_2$ (as done in the extensions $I$ and $I''$) does not affect $Q_2(I'')$ since $\varphi_2$ is true. Hence, $Q(I''') = Q(I) = Q(I') = Q(I'')$ and $I''$ is indeed in $\text{RCQ}^*(Q, D_m, V)$; this shows that $I$ is not minimal.

**Upper bound.** We show that for ground instances, $\text{MInNP}(\exists \text{FO}^+)$ is in $D_2^s$. Indeed, by Lemma 4.7, the set of yes-instances to $\text{MInNP}(\exists \text{FO}^+)$ is $L_1 \cap L_2$, where

- $L_1 = \{ (I, Q, D_m, V) \mid I \in \text{RCQ}^*(Q, D_m, V) \}$; and
- $L_2 = \{ (I, Q, D_m, V) \mid \forall t \in I, I \setminus \{t\} \notin \text{RCQ}(Q, D_m, V) \}.$

It now suffices to show that $L_1 \in \Pi_p^s$ and $L_2 \in \Sigma_p^s$. Clearly, $L_1 \in \Pi_p^s$ follows from Theorem 4.1. To show that $L_2 \in \Sigma_p^s$, we modify the algorithm for the complement problem of $\text{RCDP}(\exists \text{FO}^+)$ given in the proof of Theorem 4.1. More specifically, consider $(I, Q, D_m, V)$ and assume that $I = \{t_1, \ldots, t_k\}$. Let $I_i = I \setminus \{t_i\}$ for $i \in [1, k]$. We then apply the $\Sigma_p^s$-algorithm of each $(I_i, Q, D_m, V)$ “in parallel”, i.e., the algorithm guesses $k$ tuples $s$, (by means of a valuation of the query $Q$) such that $s_i \in Q(I_i) \setminus Q(I)$ for some partially closed extension $I'_i$ of $I_i$ (also identified by the valuation of the query $Q$). We can make such $k$ guesses at the same time since there are only polynomially many ($k = |I|$) instances $I_i$. The algorithm rejects a guess as long as any of the guessed $s_i \in Q(I_i)$ or $I'_i$ is not partially closed for some $i \in [1, k]$. However, when guesses are accepted, we have found $k$ witnesses showing that none of the $I_i$‘s are in $\text{RCQ}^*(Q, D_m, V)$. In other words, $(I, Q, D_m, V) \notin L_2$. \hfill \Box
5. WEAK RELATIVE INFORMATION COMPLETENESS

We next investigate RCDP, RCQP and MINP for weakly complete databases, denoted by RCDP\textsuperscript{w}, RCQP\textsuperscript{w} and MINP\textsuperscript{w}, respectively. We consider the databases from which one can find the certain answers to a query over their partially closed extensions. Here we denote by RCQ\textsuperscript{w}(Q, D, V) the set of instances that are weakly complete. In the weak completeness model, none of these problems has been studied before, neither for c-instances nor ground instances. We provide their complexity bounds here.

Compared to their counterparts in the strong model, the complexity results in the weak model are more diverse. On one hand, the certain-answer semantics simplifies the analysis of some problems, e.g., all these problems become decidable for FP, in contrast to their undecidability in the strong model. On the other hand, it makes certain problems harder, e.g., MINP\textsuperscript{w} becomes \( \Pi_3^p \)-complete for UCQ, as opposed to \( \Pi_3^p \) for MINP\textsuperscript{s}. In addition, some problems even have different bounds for CQ and UCQ, e.g., MINP\textsuperscript{w} is coDP-complete for CQ, while it is \( \Pi_4^p \)-complete for UCQ.

5.1. The Relatively Complete Database Problem in the Weak Model

As opposed to Theorem 4.1, RCDP\textsuperscript{w} is decidable for FP. In addition, RCDP\textsuperscript{w} for c-instances and RCDP\textsuperscript{w} for ground instances are both \( \Pi_3^p \)-complete when \( L_Q \) is CQ, UCQ or \( \exists FO^+ \), while their counterparts in the strong model are \( \Pi_3^p \)-complete (Theorem 4.1).

**Theorem 5.1.** For c-instances and for ground instances, RCDP\textsuperscript{w}(L\textsubscript{Q}) is

— undecidable when \( L_Q \) is FO,
— \textsc{Con}EXPTIME-complete when \( L_Q \) is FP, and
— \( \Pi_3^p \)-complete when \( L_Q \) is CQ, UCQ or \( \exists FO^+ \).

**Proof.** We show that in the weak model, RCDP\textsuperscript{w}(L\textsubscript{Q}) is undecidable when \( L_Q \) is FO, \textsc{Con}EXPTIME-complete when \( L_Q \) is FP, and it becomes \( \Pi_3^p \)-complete when \( L_Q \) is CQ, UCQ, or \( \exists FO^+ \). The lower bounds hold even when only ground instances are considered, and when \( D_m \) and \( V \) are fixed.

(1) When \( L_Q \) is FO. To prove the undecidability it suffices to consider ground instances without variables only. Indeed, a ground instance is also a c-instance.

We prove the undecidability by reduction from a variant of the satisfiability problem for FO. It is to decide, given an FO query \( q \) such that \( q(\emptyset) = \emptyset \) (i.e., \( q \) is not satisfied by the empty instance), whether \( q \) is satisfiable. It is easy to verify that this variant of FO satisfiability is also undecidable, by reduction from FO satisfiability.

Consider an FO query \( q \) such that \( q(\emptyset) = \emptyset \). Assume w.l.o.g. by Lemma 3.2 that the given FO query \( q \) is defined on a single relation \( R \). Then we define \( Q \) over \( R \) such that for every instance \( I \) of \( R \), \( Q(I) = \{i\} \) if \( q(I) = \emptyset \), and \( Q(I) = \emptyset \) otherwise. We define \( D_m \) to be an empty instance, and \( V \) to be the empty set. We show that \( q \) is not satisfiable if and only if the empty instance \( \emptyset \) is in RCQ\textsuperscript{w}(Q, D, V).

First assume that \( q \) is not satisfiable, i.e., for all instances \( I \) of \( R \), \( q(I) = \emptyset \). Then for all \( I \in \text{Ext}(\emptyset) \), \( Q(I) = \{\} \), and hence, \( \bigcap_{I \in \text{Ext}(\emptyset)} Q(I) = \{\} = Q(\emptyset) \). Thus \( \emptyset \) is in RCQ\textsuperscript{w}(Q, D, V).

Conversely, assume that \( q \) is satisfiable, i.e., there exists an instance \( I_0 \) of \( R \) such that \( q(I_0) \) is not empty. Then by the definition of \( Q \), \( Q(I_0) = \emptyset = \bigcap_{I \in \text{Ext}(\emptyset)} Q(I) \neq Q(\emptyset) \neq \{\} \), since \( q(\emptyset) = \emptyset \). Hence \( \emptyset \) is not in RCQ\textsuperscript{w}(Q, D, V).

We remark that in the proof above, the master data \( D_m \) and the set \( V \) of CCs are fixed, independent of the input query \( q \). In fact, they are even absent.
A:32 Capturing Missing Tuples and Missing Values

(2) When \( L_Q \) is FP. We show that RCDP\(^{w}\)(FP) is coNEXPTIME-complete. That is, we show that RCDP\(^{w}\)(FP) is already coNEXPTIME-hard for ground instances, and is in coNEXPTIME for arbitrary \( c \)-instances.

**Lower bound.** We show that for ground instances, RCDP\(^{w}\)(FP) is coNEXPTIME-hard by reduction from the SUCCINCT-TAUT problem, which is coNEXPTIME-complete (cf. Papadimitriou 1994). An instance of SUCCINCT-TAUT is defined by a Boolean circuit \( C \) consisting of a finite set of gates \( \{ g_i = (a_i, j, k) \mid 1 \leq i \leq M \} \), where \( a_i \in \{ \wedge, \vee, \neg, \text{in} \} \) is the type of the gate \( g_i \), and \( g_j \) and \( g_k \) for \( j, k < i \) are the inputs of the gate (unless \( g_j \) is an input gate in which case \( j = k = 0 \), or unless \( g_j \) is a \( \neg \)-gate in which case \( j = k \)). Suppose that \( C \) has \( n \) input gates, then \( C \) defines the Boolean function \( f_C : \{0, 1\}^n \rightarrow \{0, 1\} \), where \( f_C(\bar{w}) = 1 \) if and only if \( C \) evaluates to true on input \( \bar{w} \). The SUCCINCT-TAUT problem is to decide whether for all \( \bar{w} \in \{0, 1\}^n \), \( f_C(\bar{w}) = 1 \), i.e., whether \( C \) is a tautology.

Given an instance of the latter problem, we define database schemas \( R \) and \( R_m \), a ground instance \( I \) of \( R \), a set \( V \) of CCs, master data \( D_m \) of \( R_m \), and a FP query \( Q \).

(a) The database schema \( R \) consists of a single relation \( R(A_0, A_1, \ldots, A_{30}) \), where for a tuple \( t \) of \( R \), \( t[A_0, A_2] \) is to encode \( I_0(1); \) \( t[A_3, \ldots, A_{14}] \) encodes \( I_v; \) \( t[A_{15}, \ldots, A_{26}] \) encodes \( I_x \) and \( t[A_{27}, \ldots, A_{30}] \) encodes \( I_\neg \).

(b) A ground instance \( I \) of \( R \) consists of a single tuple \( t \) that is formed by juxtaposing instances \( I_0(1), I_v, I_x \) and \( I_\neg \) in \( t[A_0, \ldots, A_{30}] \) with \( t[A_0] = 1 \). Here \( I_0(1), I_v, I_x \) and \( I_\neg \) are given in Figure 2.

(c) The master data \( D_m \) and CCs in \( V \) ensure that every tuple \( t \) of \( R \) satisfies the following: (i) \( t[A_0, \ldots, A_{30}] \) encodes the instances mentioned above; and (ii) \( t[A_0] \) only takes values from \( \{0, 1\} \).

(d) The query \( Q \) in FP uses a \((n+1)\)-ary IDB predicates \( G_i \), one for each gate \( g_i = (a_i, j, k) \) in \( C \), a unary IDB \( I \) to encode \( I_0(1) \), and an \( n \)-ary IDB \( R_x \) to encode all possible \( n \)-ary binary tuples. That is, we include the following FP rules in \( Q \):

\[
I(x) \leftarrow R(A_0, x, A_2, \ldots, A_{30});
\]
\[
I(x) \leftarrow R(A_0, A_1, x, A_3, \ldots, A_{30});
\]
\[
R_x(\bar{x}) \leftarrow I(x_1), \ldots, I(x_n);
\]

If \( a_i \) is in then \( G_i(B, \bar{x}) \leftarrow R_x(\bar{x}), B = x; \)

If \( a_i = \lor \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), G_k(B_2, \bar{x}), R(A_0, A_1, A_2, B_1, B_2, B, A_6, \ldots, A_{30}); \)

\[
\vdots
\]

If \( a_i = \lor \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), G_k(B_2, \bar{x}), R(A_0, \ldots, A_{11}, B_1, B_2, B, A_{15}, \ldots, A_{30}); \)

\[
\vdots
\]

If \( a_i = \land \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), G_k(B_2, \bar{x}), R(A_0, \ldots, A_{14}, B_1, B_2, B, A_{18}, \ldots, A_{30}); \)

\[
\vdots
\]

If \( a_i = \neg \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), G_k(B_2, \bar{x}), R(A_0, \ldots, A_{23}, B_1, B_2, B, A_{27}, \ldots, A_{30}); \)

If \( a_i = \neg \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), R(A_0, \ldots, A_{26}, B_1, B, A_{29}, A_{30}); \)

If \( a_i = \neg \) then \( G_i(B, \bar{x}) \leftarrow G_j(B_1, \bar{x}), R(A_0, \ldots, A_{28}, B_1, B); \)

and finally, two more rules:

\[
G(\bar{x}) \leftarrow G_M(B, \bar{x}), R(0, A_1, \ldots, A_{30});
\]
\[
G(\bar{x}) \leftarrow G_M(B, \bar{x}), B = 1,
\]

where \( G_M \) is the IDB corresponding to the output gate \( g_M \). Intuitively, \( Q(I) \) will return all \( \bar{w} \in \{0, 1\}^n \) for which \( f_C(\bar{w}) = 1 \).
We show that \( C \) is a tautology if and only if \( \mathcal{I} \in \text{RCQ}^w(Q, D_m, V) \).

Conversely, suppose that \( C \) is not a tautology but \( \mathcal{I} \) is weakly complete. Note again that the ground instance \( \mathcal{I} \) mentioned above is the only extension of \( \mathcal{I} \), and furthermore, that \( Q(\mathcal{I}') \) contains all \( \bar{w} \in \{0, 1\}^n \). Hence, in order for \( \mathcal{I} \) to be weakly complete, \( Q(\mathcal{I}) \) must contain all \( \bar{w} \in \{0, 1\}^n \). This, however, contradicts the assumption that \( C \) is not a tautology since \( Q(\mathcal{I}) \) only contains those \( \bar{w} \in \{0, 1\}^n \) for which \( f_C(\bar{w}) = 1 \) (recall that \( \bar{I}[A_0] = 1 \)). Hence, \( \mathcal{I} \) cannot be weakly complete.

**Upper bound.** We show that \( \text{RCDP}^w(\text{FP}) \) is in \( \text{conEXPTIME} \) by providing an \( \text{NEXPTIME} \) algorithm that decides the complement problem. That is, given a \( c \)-instance \( T \), master data \( D_m \), a set \( V \) of \( CCs \) and an \( FP \) query \( Q \), the algorithm returns “yes” if \( T \) is not weakly complete for \( Q \) relative to \( (D_m, V) \), and “no” otherwise.

To do this, we first give a sufficient and necessary condition for characterizing weak completeness, by the lemma below. By Lemma 3.2 we assume \( w.l.o.g. \) that \( R \) consists of a single relation schema \( R \) and \( T \) is a \( c \)-table \((T, \xi)\). Recall the notion of Adom given in the proof of Theorem 4.1.

**Lemma 5.2.** For every \( Q \) in \( \text{FP} \), master data \( D_m \), set \( V \) of \( CCs \), and any \( c \)-instance \( T = (T, \xi) \), let \( T' = (T \cup \{(x_1, \ldots, x_n), \xi\}) \), i.e., \( T \) extended with a single tuple consisting of \( (new) \) variables only. Then \((T, \xi)\) is not in \( \text{RCQ}^w(Q, D_m, V) \) if and only if there exist a tuple \( t \) and an instance \( \mathcal{I} \in \text{ModAdom}(T) \) such that \( t \in \bigcap_{\mathcal{I} \in \text{ModAdom}(T')} Q(\mathcal{I}) \) and \( t \notin Q(\mathcal{I}). \)

**Proof.** First assume that \((T, \xi)\) is not in \( \text{RCQ}^w(Q, D_m, V) \). Then \( \bigcap_{\mathcal{I} \in \text{Mod}(T)} Q(\mathcal{I}) \neq \bigcap_{\mathcal{I} \in \text{Mod}(T'), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}'). \) By the monotonicity of \( \text{FP} \), we have that \( \bigcap_{\mathcal{I} \in \text{Mod}(T)} Q(\mathcal{I}) \subseteq \bigcap_{\mathcal{I} \in \text{Mod}(T'), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}'). \) Thus there must exist a tuple \( t' = (a_1, \ldots, a_n) \) such that \( t' \in \bigcap_{\mathcal{I} \in \text{Mod}(T), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}) \) and \( t' \notin \bigcap_{\mathcal{I} \in \text{Mod}(T)} Q(\mathcal{I}). \) In other words, there exists an \( \mathcal{I} \in \text{Mod}(T) \) such that \( t' \in \bigcap_{\mathcal{I} \in \text{Mod}(T'), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}) \) and \( t' \notin Q(\mathcal{I}). \) Note that \( \bigcap_{\mathcal{I} \in \text{Mod}(T), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}') \subseteq \bigcap_{\mathcal{I} \in \text{Mod}(T')} Q(\mathcal{I}'), \) and then \( t' \in \bigcap_{\mathcal{I} \in \text{Mod}(T')} Q(\mathcal{I}'). \) Construct a tuple \( t = (b_1, \ldots, b_n) \) from \( t' \) that takes values from Adom, such that for each \( i \in [1, n], b_i = a_i \) if \( a_i \) is a constant appearing in \( T, D_m, V \) or \( Q \); otherwise \( b_i \) takes values from new constants in Adom defined early. One can readily verify that \( t \) is in \( \bigcap_{\mathcal{I} \in \text{Mod}(T')} Q(\mathcal{I}') \) and then in \( \bigcap_{\mathcal{I} \in \text{ModAdom}(T') \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}'), \) but \( t \) is not in \( Q(\mathcal{I}). \)

Conversely, suppose that there exist a tuple \( t \) and an instance \( \mathcal{I} \in \text{ModAdom}(T) \) such that \( t \in \bigcap_{\mathcal{I} \in \text{ModAdom}(T)} Q(\mathcal{I}) \) and \( t \notin Q(\mathcal{I}). \) Observe that \( t \notin \bigcap_{\mathcal{I} \in \text{Mod}(T), \mathcal{I} \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}). \) Indeed, suppose otherwise. Then there exist \( \mathcal{I}_1 \in \text{Mod}(T) \) and \( \mathcal{I}_2 \in \text{Ext}(\mathcal{I}) \) such that \( t \notin Q(\mathcal{I}_3) \). By the monotonicity of \( \text{FP} \), \( t \notin Q(\mathcal{I}_3) \) for every \( \mathcal{I}_3 = \mathcal{I}_1 \cup \{s\} \) with \( s \in \mathcal{I}_2 \setminus \mathcal{I}_1 \). Pick such an \( \mathcal{I}_3 \) and let \( t' \) be the corresponding valuation of \( \mathcal{I}'_3 \) taking values from \( \mathcal{I}_3 \). This induces a valuation \( \mu' \) of \( \mathcal{I}'_3 \) with values in Adom such that \( t \notin Q(\mu'(\mathcal{I}')) \), contradicting the assumption that \( t \in \bigcap_{\mathcal{I} \in \text{ModAdom}(T')} Q(\mathcal{I}). \) Thus \( \mathcal{I} \) is not weakly complete.

Capitalizing on the characterization, we next present the \( \text{NEXPTIME} \) algorithm.

1. Guess a tuple \( t \) of the (output) schema of \( Q \) with values from Adom.
2. Check whether \( t \notin Q(\mu(T)) \) for some valuation \( \mu \) of \( T \) taking values from Adom; if so continue, and otherwise reject the guess. Since the valuations range over a finite domain, each \( \mu(T) \) is of polynomial size and evaluating \( \text{FP-queries} \) takes \( \text{EXPTIME} \), the total cost of this process is \( \text{EXPTIME} \).
(3) When \( L_Q \) is CQ, UCQ or \( \exists \mathbb{FO}^+ \). It suffices to show that RCDP\(^w\)(CQ) is \( \Pi_3^p \)-hard for ground instances and RCDP\(^w\)(\( \exists \mathbb{FO}^+ \)) is in \( \Pi_3^p \) for \( c \)-instances.

**Lower bound.** We show that for ground instances, RCDP\(^w\)(CQ) is \( \Pi_3^p \)-hard by reduction from the complement of \( \exists \forall \exists SAT \)-problem. Given a formula \( \varphi = \exists \forall \exists \exists \exists SAT \), \( \exists \forall \exists SAT \) is to determine whether \( \varphi \) is true. Here \( X = \{ x_1, \ldots, x_n \}, Y = \{ y_1, \ldots, y_m \} \) and \( Z = \{ z_1, \ldots, z_i \} \), which are sets of variables; and \( \psi = C_1 \land \cdots \land C_r \) is an instance of \( SAT \). It is known that \( \exists \forall \exists SAT \) is \( \Sigma_3^p \)-complete (cf. Papadimitriou 1994).

Given an instance \( \varphi = \exists \forall \exists \exists SAT \psi \), we define database schemas \( \mathcal{R} \) and \( \mathcal{R}_m \), a ground instance \( I \), a set \( V \) of \( C \)s, master data \( D_m \) and a \( Q \) query \( Q \). We show that \( \varphi \) is true if and only if \( I \) is not weakly complete for \( Q \) relative to \( (D_m, V) \).

(a) The database schema \( \mathcal{R} \) consists of five relation schemas: \( \mathcal{R}_{(0,1)}(A), \mathcal{R}_{-}(A, \bar{A}) \), \( \mathcal{R}_Y(A_1, A_2, B), \mathcal{R}_Z(A_1, A_2, B) \), which are the same as their counterparts given in the proof of Proposition 3.3, respectively. In addition, \( \mathcal{R} \) contains one extra relation \( R_Y(Y_1, \ldots, Y_m) \) to generate truth assignments of \( Y \) variables.

(b) The ground instance \( I \) is given by \((I_{(0,1)}, I_{-}, I_Y, I_Z, I_Y)\). Here \( I_{(0,1)}, I_{-}, I_Y, I_Z \) and \( I_\Lambda \) are the instances shown in Figure 2, and \( I_Y \) is an empty instance of \( R_Y \).

(c) The schema \( \mathcal{R}_m \) of master data contains \( R_{(0,1)}^m = R_{(0,1)}, R_{-}^m = R_{-}, R_{Y}^m = R_{Y}, R_{Z} = R_{Z} \), and and \( R_0(W, \bar{W}) \). The master data instance \( D_m \) consists of \( I_{(0,1)}^m = I_{(0,1)} \), \( I_Y^m = I_Y \), \( I_Z^m = I_Z \) and \( I_\Lambda^m = \emptyset \).

(d) The set \( V \) of \( C \)s consists of the following \( C \)s:

\[ R_{(0,1)} \subseteq R_{(0,1)}^m, R_{-} \subseteq R_{-}^m, R_{Y} \subseteq R_{Y}^m, R_{Z} \subseteq R_{Z}^m; \]

\[ \phi_i : q_i(y_i) \subseteq R_{(0,1)}^m, \text{ where } q_i(y_i) = \exists y_1 \ldots y_{i-1} y_{i+1} \ldots y_m (R_Y(y_1, \ldots, y_m)), i \in [1, m]; \] and

\[ \phi'_i : q'_i(y_i, y'_i) \subseteq R_0^m, \text{ where for } i \in [1, m], q'_i(y_i, y'_i) = \exists y_1 y'_1 \ldots y_{i-1} y_{i+1} y'_i \ldots y_m (R_Y(y_1, \ldots, y_m) \land R_Y(y'_1, \ldots, y'_m) \land y_i \neq y_i'). \]

It is easy to see that for each extension \( I'_Y \) of \( I_Y \), \( I'_Y \models \bigwedge_{i \in [1, m]} (\phi_i \land \phi'_i) \) if and only if \( I'_Y \) consists of a single tuple that encodes a valid truth assignment of \( Y \).

(e) We define the CQ query \( Q \) as follows:

\[ Q(x) = \exists \bar{x}, \bar{y} \exists Q_X(\bar{x}) \land R_Y(\bar{y}) \land Q_Z(\bar{z}) \land Q_\psi(\bar{x}, \bar{y}, \bar{z}, w) \land w = 1, \]

where \( \bar{x} = (x_1, \ldots, x_n), \bar{y} = (y_1, \ldots, y_m) \) and \( \bar{z} = (z_1, \ldots, z_i) \). The sub-queries \( Q_X(\bar{x}) = \bigwedge_{i=1}^m R_{01}(x_i) \) and \( Q_Z(\bar{z}) = \bigwedge_{i=1}^m R_{01}(z_i) \) generate all valid truth assignments of \( X \) variables and \( Z \) variables, respectively, by means of Cartesian products of \( R_{01} \). Given truth assignments \( (\mu_X, \mu_Y, \mu_Z) \) of \( (X, Y, Z) \), query \( Q_\psi \) is to encode the truth value of \( \psi(\mu_X, \mu_Y, \mu_Z) \) as the value of \( w \). Obviously, \( Q_\psi \) can be expressed in CQ in terms of \( R_Y, R_Z \) and \( R_{-} \). Intuitively, query \( Q \) returns all truth assignments \( \mu_X \) of \( X \) if there exists truth assignments \( \mu_Y \) and \( \mu_Z \) of \( Y \) and \( Z \), respectively, that make \( \psi \) true.
We next show that the reduction is correct, i.e., \( \varphi \) is true if and only if \( \mathcal{I} \) is not weakly complete for \( Q \) relative to \((D_m, V)\).

\[ \Rightarrow \] First assume that \( \varphi \) is true. Then there exists a truth assignment \( \mu_X^0 \) of \( X \) such that for each truth assignment \( \mu_Y \) of \( Y \), there exists a truth assignment \( \mu_Z \) of \( Z \) such that \( \psi(\mu_X, \mu_Y, \mu_Z) \) is true. We next show that \( \mathcal{I} \) is not weakly complete for \( Q \) relative to \((D_m, V)\). That is, we need to show that \( Q(\mathcal{I}) \not\in \bigcap_{\mathcal{I}' \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}') \). Observe the followings: (i) \( Q(\mathcal{I}) = \emptyset \) since \( I_Y \) is empty; and (ii) for each instance \( \mathcal{I}' \) in \( \text{Ext}(\mathcal{I}) \), \( \mathcal{I}' = (I_{(0,1)}, I_1, I_Y, I_{\lambda}, I_{\lambda}') \), where \( I_{\lambda}' \) encodes a valid truth assignment of \( Y \), and \((\vec{x}_0) \in Q(\mathcal{I}') \) representing the truth assignment \( \mu_X^0 \). Thus \((\vec{x}_0) \in \bigcap_{\mathcal{I}' \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}') \). Hence \( \mathcal{I} \) is not weakly complete for \( Q \) relative to \((D_m, V)\).

\[ \Rightarrow \] Conversely, if \( \varphi \) is false, then there exists no truth assignment \( \mu_X \) of \( X \) such that for all truth assignments of \( Y \), there exists a truth assignment of \( Z \) that makes \( \psi \) true. Recall that for each partially closed extension \( \mathcal{I}' \) of \( \mathcal{I} \), \( \mathcal{I}' \) can only be the form of \((I_{(0,1)}, I_1, I_Y, I_{\lambda}, I_{\lambda}') \), where \( I_{\lambda}' \) encodes a truth assignment of \( Y \). By the definition of \( Q \), if there exists a tuple \( t \) in \( \bigcap_{\mathcal{I}' \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}') \), \( t \) must encode a truth assignment of \( X \) such that for every truth assignment of \( Y \), there exists a truth assignment of \( Z \) that makes \( \psi \) true; thus \( \varphi \) is true, which contradict the assumption that \( \varphi \) is false. As a result, \( \bigcap_{\mathcal{I}' \in \text{Ext}(\mathcal{I})} Q(\mathcal{I}') \) must be empty, and \( \mathcal{I} \) is weakly complete for \( Q \) relative to \((D_m, V)\).

**Upper bound.** It suffices to show that \( \text{RCDP}^\varphi(\exists \text{FO}^\ast) \) is in \( \Pi_p^p \). To do this, we use the same algorithm given for \( \text{RCDP}^\varphi(\text{FP}) \). We show the algorithm is in \( \Sigma_p^p \), then \( \text{RCDP}^\varphi(\exists \text{FO}^\ast) \) is in \( \Pi_p^p \). Indeed, Step (2) of the algorithm can be done in \( \Sigma_p^p \) by the following procedure.

1. Guess a valuation \( \mu \) of \( T \) by taking values from \( \text{Adom} \).
2. Check whether \( \mu(T) \) satisfies \( \xi \). If so continue; otherwise reject the guess.
3. Check whether \( t \notin Q(\mu(T)) \). If so return “yes”; otherwise reject the guess.

Obviously, it return “yes” if and only if there exists a valuation of \( T \) such that \( t \notin Q(\mu(T)) \). It is in \( \Sigma_p^p \) since step (2) is in \( \text{PTIME} \) and step (3) is in \( \text{coNP} \) for \( \text{CQ} \).

Moreover, step (3) of the algorithm is also in \( \Sigma_p^p \) for \( \exists \text{FO}^\ast \). We give a \( \Pi_p^p \) procedure for the complement of step (3), i.e., it returns “no” if and only if \( t \notin \bigcap_{\mathcal{I}' \in \text{Mod}_{\text{Adom}}(T')} Q(\mathcal{I}') \).

1. Guess a valuation \( \mu' \) of \( (T', \xi') \) by taking values from \( \text{Adom} \).
2. Check whether \( \mu'(T') \) satisfies \( \xi' \). If so continue; otherwise reject the guess.
3. Check whether \( (\mu'(T'), D_m) \models V \) and whether \( (\mu'(T), D_m) \models V \). If so continue; otherwise reject the guess.
4. Check whether \( \mu'(T') \subseteq \mu'(T') \). If so continue; otherwise reject the guess.
5. Check if \( t \notin Q(\mu'(T')) \). If so return “no”; otherwise reject the guess.

It returns “no” if and only if \( t \notin \bigcap_{\mathcal{I}' \in \text{Mod}_{\text{Adom}}(T')} Q(\mathcal{I}') \). It is in \( \Pi_p^p \) since step (2) is in \( \text{PTIME} \), step (3) is in \( \text{NP} \), and steps (4) and (5) are both in \( \text{coNP} \), for \( \exists \text{FO}^\ast \) queries.

### 5.2. The Relatively Complete Query Problem in the Weak Model

Recall that \( \text{RCQP}^c \) for \( c \)-instances is equivalent to \( \text{RCQP}^c \) for ground instances, as verified by Lemma 4.4. However, the example below tells us that it is no longer the case in the weak completeness model.

**Example 5.3.** Consider an FO query \( Q \) defined on a pair of relations: \( Q(I_1, I_2) = \{ (a) \} \) if \( I_1 \subseteq I_2 \), and it is \( \{ (b) \} \) otherwise, where \( a \) and \( b \) are distinct constants. For empty \( D_m \) and \( V \), no ground instances are in \( \text{RCQP}^c(Q, D_m, V) \) since \( Q(I_1, I_2) \neq \emptyset \) for all \( (I_1, I_2) \) while \( \bigcap_{\mathcal{I}' \in \text{Ext}(I_1, I_2)} Q(\mathcal{I}') = \emptyset \). In contrast, consider a \( c \)-instance \( T = (T_1, T_2) \),
where \( T_1 = \{ (x), \emptyset \} \) and \( T_2 = \{ (y), \emptyset \} \). Obviously, \( T \) is in \( \text{RCQP}^w(Q, D_m, V) \) since 
\[
Q(T) = \bigcap_{I \in \text{Mod}(T)} Q(I) = \emptyset = \bigcap_{I \in \text{Mod}(T), I' \in \text{Ext}(I)} Q(I').
\]

This tells us that from the undecidability of \( \text{RCQP}^w(\text{FO}) \) for ground instances we cannot conclude the undecidability for \( c \)-instances. Nevertheless, \( \text{RCQP}^w(L_Q) \) becomes trivially decidable when \( L_Q \) is \( \text{FP} \), \( \text{CQ} \), \( \text{UCQ} \) or \( \exists \text{FO}^+ \), for \( c \)-instances and for ground instances, in contrast to Theorem 4.5.

**Theorem 5.4.** \( \text{RCQP}^w(L_Q) \) is

— undecidable for ground instances if \( L_Q \) is \( \text{FO} \), and

— decidable in \( O(1) \)-time for \( c \)-instances and ground instances when \( L_Q \) is \( \text{FP} \), \( \text{CQ} \), \( \text{UCQ} \) or \( \exists \text{FO}^+ \).

The complexity is unchanged when \( D_m \) and \( V \) are fixed.

**Proof.** We refer to Appendix A for the proof of this Theorem. In a nutshell, we show that \( \text{RCQP}^w(\text{FO}) \) is undecidable for ground instances by reduction from the satisfiability problem for \( \text{FO} \), and give a constructive proof showing that there always exists a database (ground instance) that is weakly complete for \( Q \) relative to \( (D_m, V) \) when \( Q \) is an \( \text{FP} \) query. From this it follows that \( \text{RCQP}^w(\text{FP}) \) is trivially decidable.

**5.3. The Minimality Problem in the Weak Model**

In contrast to the strong completeness model, Lemma 4.7 no longer holds in the weak completeness model, i.e., to decide whether an instance \( I \) is minimal, it does not suffice to inspect \( I \setminus \{ t \} \) only.

**Example 5.5.** Consider a CQ query \( Q \) defined on a pair of unary relations \( (R_1, R_2) \): 
\[ Q(x) = \exists z (R_1(y) \land R_2(z) \land x = a). \]
That is, on an instance \( (I_1, I_2) \) of \( (R_1, R_2) \), \( Q \) returns \( \{ (a) \} \) if \( I_1 \) and \( I_2 \) are both non-empty. Consider an instance \( I_0 = (\{ (0) \}, \{ (1) \}) \), an empty set \( V \) of CCs and any master data \( D_m \). Then \( I_0 \) is in \( \text{RCQP}^w(Q, D_m, V) \). Nevertheless, it is not minimal: the empty instance \( (\emptyset, \emptyset) \) of \( (R_1, R_2) \) is also in \( \text{RCQP}^w(Q, D_m, V) \). Indeed, \( (\{ 0 \}, \emptyset) \) is a partially closed extension of \( (\emptyset, \emptyset) \) and \( Q((\{ 0 \}, \emptyset)) = \emptyset \). Then \( \bigcap_{I' \in \text{Ext}(\emptyset, \emptyset)} = \emptyset \) and thus \( (\emptyset, \emptyset) \) is in \( \text{RCQP}^w(Q, D_m, V) \). However, removing one tuple from \( I_0 \) does not make it a weakly complete instance, i.e., a counterexample to the minimality of \( I_0 \) cannot be found by removing only one tuple from \( I_0 \).

In the weak model, the minimality analysis is quite different from its counterpart in the strong model (Theorem 4.8). (a) The absence of missing values does not simplify the analysis, as opposed to their counterparts in the strong model (\( \Pi_0^\text{c} \) for ground instances vs. \( \Pi_0^\text{w} \) for \( c \)-instances). (b) It is much easier to check \( \text{MINP}^w(\text{CQ}) \) than \( \text{MINP}^w(\text{UCQ}) \) (co\text{DP}-complete vs. \( \Pi_0^\text{w} \)-complete), whereas \( \text{MINP}^w(\text{CQ}) \) and \( \text{MINP}^w(\text{UCQ}) \) have the same complexity. Recall that co\text{DP} = \text{NP} \cup \text{coNP} (cf. [Papadimitriou 1994]).

**Theorem 5.6.** For \( c \)-instances and ground instances, \( \text{MINP}^w(L_Q) \) is

— undecidable when \( L_Q \) is \( \text{FO} \),

— \( \text{coNEXPTIME} \)-complete when \( L_Q \) is \( \text{FP} \),

— \( \Pi_0^\text{w} \)-complete when \( L_Q \) is \( \text{UCQ} \) or \( \exists \text{FO}^+ \), and

— co\text{DP}-complete when \( L_Q \) is \( \text{CQ} \).

**Proof.** We show that \( \text{MINP}^w(L_Q) \) is undecidable when \( L_Q \) is \( \text{FO} \), co\text{NEXPTIME}-complete when \( L_Q \) is \( \text{FP} \), and \( \Pi_0^\text{w} \)-complete when \( L_Q \) is \( \text{UCQ} \) or \( \exists \text{FO}^+ \). When \( L_Q \) is \( \text{CQ} \) the problem is shown to be co\text{DP}-complete. All lower bounds remain intact when only ground instances are considered.
(1) When $L_Q$ is FO. It suffices to show that $\text{MINP}_w^{w}(\text{FO})$ is undecidable for ground instances. More specifically, it suffices to show that it is undecidable to determine, given an FO query $Q$, master data $D_m$ and a set $V$ of CCs, whether $\mathcal{I}_0$ is in $\text{RCQ}_w^{w}(Q, D_m, V)$, where $\mathcal{I}_0$ is the empty database instance of the schema over which $Q$ is defined. This is because if $\mathcal{I}_0$ is in $\text{RCQ}_w^{w}(Q, D_m, V)$, then it is a minimum instance complete for $Q$ relative to $(D_m, V)$. The undecidability of this problem has already been verified in the proof of Theorem 5.4.

(2) When $L_Q$ is FP. We show that $\text{MINP}_w^{w}(\text{FP})$ is coNEXPTIME-hard for ground instances and provide a coNEXPTIME algorithm for deciding $\text{MINP}_w^{w}(\text{FP})$ for $\text{c}$-instances.

Lower bound. We show that for ground instances, $\text{MINP}_w^{w}(\text{FP})$ is coNEXPTIME-hard by reduction from the SUCCINCT-TAUT problem (see a description of the problem in the proof of Theorem 5.1 (2)). Given an instance of the latter problem, we define the same database schemas $R$ and master data $D_m$, and set $V$ of CCs, master data $D_m$ of $R_m$, and FP query $Q$ as their counterparts in the proof of Theorem 5.1 (2). We show that $C$ is a tautology if and only if $I$ is the minimal in $\text{RCQ}_w^{w}(Q, D_m, V)$.

$\Rightarrow$ Suppose that $C$ is a tautology. We first show that $I \in \text{RCQ}_w^{w}(Q, D_m, V)$, i.e., $Q(I) = \bigcap_{\vec{w} \in \text{Ext}(I)} Q(I')$. As shown in the proof of Theorem 5.1 (2), $Q(I)$ returns all $\vec{w} \in \{0, 1\}^n$ since $C$ is a tautology. Moreover, the only extension $I'$ of $I$ is $(t, t')$, where $t$ is in $I$ and $t'$ is the same as $t$ except that $t'[A_0] = 0$ while $t[A_0] = 1$, and $Q(I')$ returns all $\vec{w} \in \{0, 1\}^n$ as well. Then $I \in \text{RCQ}_w^{w}(Q, D_m, V)$. One can verify that $\emptyset \not\in \text{RCQ}_w^{w}(Q, D_m, V)$. Indeed, as discussed above, $Q(I') = \{0, 1\}^n$. Hence, $I$ is weakly complete and minimal.

$\Leftarrow$ Conversely, if $C$ is not a tautology, then by the proof of Theorem 5.1 (2), $I$ is not even in $\text{RCQ}_w^{w}(Q, D_m, V)$, and hence is not minimal.

Upper bound. We provide a coNEXPTIME algorithm that, given a $c$-instance $T$, master data $D_m$, an FP query $Q$ and a set $V$ of CCs, the algorithm returns "yes" if $T$ is weakly complete and minimal. The algorithm works as follows:

(1) Check whether $T \in \text{RCQ}_w^{w}(Q, D_m, V)$. This is a coNEXPTIME process by Theorem 5.1(2). If so, continue. Otherwise return "no".

(2) For each $\Delta \subseteq T$, $\Delta \neq \emptyset$, check whether $T \setminus \Delta \in \text{RCQ}_w^{w}(Q, D_m, V)$. If such a $\Delta$ is found, return "no", otherwise return "yes". The enumeration of the subsets $\Delta$ and checking whether $T \setminus \Delta \in \text{RCQ}_w^{w}(Q, D_m, V)$ are again in coNEXPTIME.

Hence $\text{MINP}_w^{w}(\text{FP})$ is in coNEXPTIME. The correctness of the algorithm is immediate.

(3) When $L_Q$ is UC or $\exists \text{FO}^+$. We next show that $\text{MINP}_w^{w}(L_Q)$ is $\Pi_2^p$-complete when $L_Q$ is UC or $\exists \text{FO}^+$. It suffices to show that $\text{MINP}_w^{w}(\text{UC})$ is $\Pi_2^p$-hard, and $\text{MINP}_w^{w}(\exists \text{FO}^+)$ is in $\Pi_2^p$. The lower bound holds when considering ground instances only.

Lower bound. We show the $\Pi_2^p$-hardness by reduction from the $\forall \exists \forall \exists \exists \text{SAT}$ problem, which is known to be $\Pi_2^p$-complete (cf. [Papadimitriou])]. The $\forall \exists \forall \exists \exists \text{SAT}$ problem is to determine, given a sentence $\varphi = \forall X \exists Y \forall Z \exists W \psi$, whether $\varphi$ is true. Here $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $Z = \{z_1, \ldots, z_k\}$, $W = \{w_1, \ldots, w_p\}$, and $\psi$ is an instance of $3\text{SAT}$. Given an instance $\varphi = \forall X \exists Y \forall Z \exists W \psi$ of the $\forall \exists \forall \exists \exists \text{SAT}$ problem, where $\psi = C_1 \land \cdots \land C_r$, we define a database schema $\mathcal{R}$, a ground instance $\mathcal{I}_0$ of $\mathcal{R}$, master data $D_m$, a set $V$ of CCs and a query $Q$ in UCQ, such that $\varphi$ is true if and only if either $\mathcal{I}_0$ is not in $\text{RCQ}_w^{w}(Q, D_m, V)$ or $\mathcal{I}_0 \in \text{RCQ}_w^{w}(Q, D_m, V)$ but it is not minimal.

(a) The database schema $\mathcal{R}$ consists of the following six relation schemas: $R_{(0,1)}(X)$, $R_{i}(A_1, A_2, B)$, $R_{s}(A_1, A_2, B)$, $R_{-}(A_1, A_2)$, $R_{x}(\text{id, } X)$, and $R_{z}(Z_1, \ldots, Z_k)$. Instances of $R_{x}(\text{id, } X)$ are to encode truth assignments of $X$, and instances of $R_{z}$ are singleton
sets, each encoding a truth assignment of $Z$. We let $I_0 = (I_X, I_{(0,1)}, I_V, I_\Lambda, I_\tau, I_2)$, where $I_X = \{(1,0), (1,1), \ldots, (n,0), (n,1)\}, I_Z = \emptyset$, and $I_{(0,1)}, I_V, I_\Lambda$, and $I_\tau$ are as shown in Figure 2.

(b) The master data $D_m$ and CCs $V$ ensure the following: (i) any instance $I_X$ of $R_X$ satisfies $I_X \subseteq I_1$; (ii) any instance $I_Z$ of $R_Z$ consists of a single tuple with values taken from \{0, 1\}; and (iii) instances of $R_{(0,1)}, R_V, R_\Lambda$, and $R_\tau$ are subsets of $I_{(0,1)}, I_V, I_\Lambda$, and $I_\tau$, respectively. Clearly $(I, D_m) \models V$.

(c) The query $Q$ is defined as $Q_1 \cup \cdots \cup Q_{2n+12} \cup P_1 \cup P_2 \cup P_3$, where the $Q_i$'s guarantee the existence of certain tuples in the query result when the input instance consists of at least $i$ tuples, for $i \in [1, 2n + 12]$; query $P_1$ generates an additional tuple when the instance of $R_X$ contains a proper truth assignment of $X$; query $P_2$ is to eliminate the effect on the certain answers of extensions that do not correspond to proper truth assignments of $X$; and finally $P_3$ generates truth assignments of $Y$ that satisfy certain properties related to $\varphi$. As we will see below, $I_0 \in \text{RCQ}^w(Q, D_m, V)$ but it is only minimal when there does not exist a proper subset $I_X$ of $I_X$ such that $I_0 = (I_X, I_{(0,1)}, I_V, I_\Lambda, I_\tau, \emptyset) \in \text{RCQ}^w(Q, D_m, V)$. Furthermore, $Q$ is defined in such a way that only $I_X$ that encode valid truth assignments of $X$ need to be considered. In particular, $I_0 \in \text{RCQ}^w(Q, D_m, V)$ if and only if the truth assignment $\mu_X$ encoded in $I_X$ is such that there does not exist a $\mu_Y$ of $Y$ such that for every $\mu_Z$ of $Z$, $\exists W \psi(\mu_X, \mu_Y, \mu_Z, W)$ evaluates to true. In other words, $I_0 \in \text{RCQ}^w(Q, D_m, V)$ if and only if $\varphi$ is false. As a consequence, $I_0$ is minimal if and only if $\varphi$ is true. In the rest of the lower bound proof, we use $\mathcal{I}$ and $\mathcal{I}'$ to range over the instances of $R, \mathcal{I}^-$ and $(\mathcal{I}')^-$ to denote sub-instances (proper subsets) of $\mathcal{I}$ and $\mathcal{I}'$, respectively.

We next explain the disjuncts in $Q$ in more detail. All queries have output arity $m$, the number of variables in $Y$. Observe that the maximal size of partially closed instances is $2n + 13$, i.e., there are at most $2n$ tuples in instances of $R_X, 12$ tuples in the instances corresponding to $R_{(0,1)}, R_V, R_\Lambda$, and $R_\tau$, and at most one tuple in instances of $R_Z$.

For $i \in [1, 2n + 12]$, we define $Q_i(\bar{u})$ as a UCQ that returns $\bar{a}_i = (a_i, \ldots, a_i)$ whenever the input instance has size at least $i$. Here $a_i$ is a fresh new constant not used anywhere else. Clearly, such a query can be expressed in UCQ by using $\neq$.

Consider $\bar{Q}' = Q_1 \cup \cdots \cup Q_{2n+12}$ and any instance $\mathcal{I}$ of $R$ of size $i$. Then, $Q'(\mathcal{I}) = \{\bar{a}_1, \ldots, \bar{a}_i\}$. However, for any extension $\mathcal{I}'$ of $\mathcal{I}$ (i.e., for $\mathcal{I}' \in \text{Ext}(\mathcal{I})$), we have that $Q'(\mathcal{I}) \subseteq Q'(\mathcal{I}')$ since the latter surely contains $\{\bar{a}_1, \ldots, \bar{a}_i, \bar{a}_{i+1}\}$. In other words, if we were to use only $Q'$ instead of $Q$, no strict sub-instance $\mathcal{I}^-$ of $\mathcal{I}_0$ can be in $\text{RCQ}^w(Q', D_m, V)$. We will see shortly how the additional query $P_1$ in $Q$ provides the opportunity for specific sub-instances of $\mathcal{I}$, i.e., those that correspond to valid truth assignments of $X$, to be in $\text{RCQ}^w(Q, D_m, V)$. Observe that $Q'(\mathcal{I}_0) = \{\bar{a}_1, \ldots, \bar{a}_{2n+12}\}$. Similarly, $Q'(\mathcal{I}^+_0) = \{\bar{a}_1, \ldots, \bar{a}_{2n+12}\}$ for any extension $\mathcal{I}^+_0$ of $\mathcal{I}_0$. Indeed, $Q'$ stops adding fresh tuples to the query result once the instance grows in size beyond $2n + 12$. Hence, $Q'$ helps us ensure that $\mathcal{I}_0 \in \text{RCQ}^w(Q', D_m, V)$.

We next define the queries $P_1, P_2$ and $P_3$. First, we let

$$P_1(\bar{u}) = (\exists \bar{x} \bigwedge_{i \in [1,n]} R_X(i, x_i) \land Q_{all} \land \bar{u} = \bar{a}_{n+13},$$

where $Q_{all}$ is to ensure that all the tuples in $I_{(0,1)}, I_\Lambda, I_V$ and $I_\tau$ are in place. That is, query $P_1$ puts $\bar{a}_{n+13}$ into the query result on instances $\mathcal{I}'$ of $R$ for which $\mathcal{I}'$ contains a truth assignment $\mu_X$ of $X$, i.e., when $\mathcal{I}'$ contains tuples of the form $(i, v)$ for each $i \in [1,n]$, and when all instances encoding Boolean domain and operations are present.
Observe that \( P_1 \) has effect only when \( T' \) has size \( n + 12 \) (when its \( R_X \) instance encodes a valid truth assignment for \( X \)). Indeed, \( \bar{a}_{n+13} \) is already in the query result for larger instances because of the \( Q_2 \)'s is described above. On the other hand, it cannot affect instances \( T' \) of smaller size. Indeed, \( I_X \) must contain at least \( n \) tuples and all 12 tuples in \( I_0(1), I_1, I_2 \) and \( I_3 \), must be present.

It is easily verified that for \( Q'' = Q' \cup P_1, I_0'' \in RCQ^w(Q'', D_m, V) \) if and only if either (a) \( I = I_0 \) or (b) \( I_0'' \) consists of precisely \( n + 12 \) tuples and satisfies the condition in \( P_1 \). We refer to such sub-instances \( I_0'' \) of \( I_0 \) as weakly complete candidates. That is, we use \( Q' \) and \( P_1 \) to distinguish weakly complete candidates. Observe that the certain answers of \( Q'' \) on extensions of \( I_0'' \) is equal to \( \{\bar{a}_1, \ldots, \bar{a}_{n+12}, \bar{a}_{n+13}\} \), which equals \( Q''(I_0) \).

What remains to show is that no weakly complete candidates can be weakly complete if and only if \( \varphi \) is true. For if this holds, \( I_0'' \) is minimal in \( RCQ^w(Q, D_m, V) \) if and only if \( \varphi \) is true, as desired. To show this, we need the two additional queries \( P_2 \) and \( P_3 \), which are defined as follows.

Query \( P_2 \) is to ensure that for any extension \( (I_0')^+ \) of weakly complete candidates \( I_0'' \), if its \( R_X \) instance \( (I_X')^+ \) does not encode a truth assignment of \( X \), then \( (I_0')^+ \) does not affect the certain answer result. More specifically, \( P_2 \) is a disjunction of \( n \) queries \( P_2,i(y) = R_X(i, 0) \wedge R_X(i, 1) \wedge R_R(0, 1)(y) \wedge \cdots \wedge R_R(0, m)(y) \). That is, whenever \( P_2 \) is applied on an instance \( (I_0')^+ \) such that \( (I_X')^+ \) contains two possible values for a variable \( x_i \) (encoded by \( i, 0 \) and \( i, 1 \)), it puts all truth assignments of \( Y \) in the query result. We denote by \( F_Y \) the set of all possible truth assignments of \( Y \).

Consider \( Q''' = Q' \cup P_2 \). Observe that \( Q'''(I_0) = \{\bar{a}_1, \ldots, \bar{a}_{n+12}\} \cup F_Y = Q''(I_0') \) for any extension of \( I_0'' \) of \( I_0 \). Hence, \( I_0'' \in RCQ^w(Q'', D_m, V) \). Similarly, for a weakly complete candidate \( I_0'' \), \( Q''(I_0'') = \{\bar{a}_1, \ldots, \bar{a}_{n+13}\} = \bigcap_{(I_0')^+ \in Ext(I_0')} Q'''(I_0')^+ \). Indeed, this follows from the fact that \( Ext(I_0'') \) contains at least one instance \( I_1 \) of size \( n + 13 \) on which \( P_2 \) does not produce \( F_Y \). For example, \( I_1 \) could be \( I_0'' \) extended with an additional tuple in its \( R_Z \) instance \( I_0'' \). Since \( Q''(I_1) = Q(I_0) = \{\bar{a}_1, \ldots, \bar{a}_{n+13}\} \subseteq \bigcap_{(I_0')^+ \in Ext(I_0')} Q'''(I_0')^+ \), we may conclude that \( P_2 \) alone does not prevent weakly complete candidates to be in \( RCQ^w(Q'', D_m, V) \). The relevance of \( P_2 \) comes only in play together with query \( P_3 \), which we define next.

More specifically,

\[
P_3(g) = \exists \bar{x}, \bar{z}, \bar{w}(Q_X(\bar{x}) \land Q_Y(\bar{y}) \land Q_Z(\bar{z}) \land Q_W(\bar{w}) \land Q_\psi(\bar{x}, \bar{y}, \bar{z}, \bar{w}, 1)),
\]

where \( Q_X(\bar{x}) = \bigwedge_{i \in [1, n]} R_X(i, x_i) \), and \( Q_Y(\bar{y}) \) and \( Q_W(\bar{w}) \) generate all \( k \) and \( p \) binary tuples by means of Cartesian products of \( R_R(0, 1) \). Furthermore, \( Q_\psi(\bar{x}, \bar{y}, \bar{z}, \bar{w}, v) \) is a CQ query that encodes the truth value of \( \psi(\mu_X, \mu_Y, \mu_Z, \mu_W, v) \) for given truth assignment \( \mu_X \) of \( X \), \( \mu_Y \) of \( Y \), \( \mu_Z \) of \( Z \) and \( \mu_W \) of \( W \) as encoded by \( \bar{x}, \bar{y}, \bar{z} \) and \( \bar{w} \), respectively. In other words, \( v = 1 \) if \( \psi(\mu_X, \mu_Y, \mu_Z, \mu_W, v) \) holds and \( v = 0 \) otherwise. Query \( Q_\psi \) is encoded by means of \( R_R(0, 1), R_Y, R_Z \) and \( R_W \), as before.

We now have that \( Q = Q'' \cup P_3 \). Observe that \( Q(I_0) = Q''(I_0) \). Furthermore, since \( Q''(I_0) \) already contains \( F_Y \), no further tuples can be added to the query result in any extension of \( I_0 \). Hence, \( I_0 '' \) remains a weakly complete database for \( Q, D_m \) and \( V \).

We next investigate the impact of \( P_3 \) on weakly complete candidates \( I_0'' \). Since \( I_Z = \emptyset \) in \( I_0, I_Z = \emptyset \) in any weakly complete candidate \( I_0'' \); therefore, \( P_3 \) does not add additional tuples to \( Q''(I_0'') \), i.e., \( Q(I_0'') = Q''(I_0'') \). On the other hand, consider an extension \( (I_0')^+ \) of \( I_0'' \) on which \( P_3 \) does not apply (otherwise \( P_2 \) would already have added \( F_Y \) to the query result and hence \( P_3 \) does not have an impact). Recall that such extensions exist by simply adding a tuple to \( V \). Then \( P_3((I_0')^+) \) is either (a) empty, in which case \( \bigcap_{(I_0')^+ \in Ext(I_0')} Q((I_0')^+) = \{\bar{a}_1, \ldots, \bar{a}_{n+13}\} = Q(I_0) \), or (b) \( P_3((I_0')^+) \) returns \( F_Y \),
where $F'_v$ consists of all truth assignments of $Y$ that satisfy the condition in $P_3$. It is easily verified that $\bigcap_{(\mathcal{I}_0)^+} Q((\mathcal{I}_0)^+) = \{\bar{a}_1, \ldots, \bar{a}_{n+13}\} \cup C_Y$, where $C_Y$ denotes the set of truth assignments returned by $P_3$ on all extensions $(\mathcal{I}_0)^+$ of $\mathcal{I}_0$. Hence, the weakly complete candidate $\mathcal{I}_0^*$ is weakly complete if and only if $C_Y$ is empty. Indeed, recall that $Q(\mathcal{I}_0^*) = \{\bar{a}_1, \ldots, \bar{a}_{n+13}\}$. That is, for the truth assignment $\mu_Y$ encoded in $J'_X$, there is no truth assignment $\mu_Y$ of $Y$ such that for all truth assignments $\mu_Z$ of $Z$ (note that all $\mu_Z$ are considered as a tuple in $J'_Z$ in some extension $(\mathcal{I}_0)^+$ of $\mathcal{I}_0$), there exists a truth assignment $\mu_W$ of $W$ that makes $\psi$ true. That is, a weakly complete candidate is actually in $\text{RCQ}^w(Q, D_m, V)$ if and only if $\varphi$ is false, as desired. As a result, $\mathcal{I}$ is a minimal instance in $\text{RCQ}^w(Q, D_m, V)$ if and only if $\varphi$ is true.

**Upper bound.** We show that $\text{MINP}^w(\exists \text{FO}^+)$ is in $\text{NP}^\Sigma_4^P$ or $\Sigma_4^P$ and therefore, $\text{MINP}(\exists \text{FO}^+)$ in $\Pi_3^p$. The correctness of the algorithm is immediate.

(1) Check whether $T \notin \text{RCQ}^w(Q, D_m, V)$. If so, then return “yes”. Otherwise continue.

By Theorem 5.1(3) this step is in $\Sigma_4^P$.

(2) We guess $\Delta \subseteq T$ and make a call to a $\Pi_3^p$-oracle, to check whether $T \setminus \Delta \in \text{RCQ}^w(Q, D_m, V)$. If not, reject the current guess, otherwise return “yes”.

Hence, the overall complexity of the algorithm is $\text{NP}^\Sigma_4^P$ or $\Sigma_4^P$ and therefore, $\text{MINP}(\exists \text{FO}^+)$ is in $\Pi_3^p$. The correctness of the algorithm is immediate.

(4) When $L_Q$ is CQ. We show that $\text{MINP}^w(\text{CQ})$ is coDP-complete for ground instances. By Lemma 3.2, we assume w.l.o.g. that $T$ is defined over a single relation schema. When $L_Q$ is CQ, the minimal weakly complete databases are rather restrictive as verified by the following Lemma:

**Lemma 5.7.** Given a CQ query $Q$, master data $D_m$, a set $V$ of CCs and a $c$-instance $T$, $T$ is a minimal instance in $\text{RCQ}^w(Q, D_m, V)$ if and only if either $T = \emptyset$ is in $\text{RCQ}^w(Q, D_m, V)$, or when $\emptyset \notin \text{RCQ}^w(Q, D_m, V)$, $|T|$ is a singleton set and $\text{Mod}(T, D_m, V) \neq \emptyset$.

**Proof.** Let $(T_Q, u_Q)$ denote the tableau representation of Q. We distinguish between the following two cases: (i) $\emptyset \notin \text{RCQ}^w(Q, D_m, V)$; and (ii) $\emptyset \notin \text{RCQ}^w(Q, D_m, V)$. Clearly, in case (i), $T$ is a minimal instance in $\text{RCQ}^w(Q, D_m, V)$ if and only if $T = \emptyset$. Suppose that case (ii) holds. We then show that $\{\tau\} = \text{RCQ}^w(Q, D_m, V)$ for every $\tau \in T$, where $\text{Mod}(\tau, D_m, V) \neq \emptyset$. Hence one can readily verify that in case (ii), $T$ is a minimal instance in $\text{RCQ}^w(Q, D_m, V)$ if and only if $|T| = 1$ and $\text{Mod}(T, D_m, V) \neq \emptyset$.

Suppose that $\emptyset \notin \text{RCQ}^w(Q, D_m, V)$. Then $\bigcap_{\{t\}, D_m \models Q} Q(\{t\}) \neq \emptyset$. Indeed, once $\bigcap_{\{t\}, D_m \models Q} Q(\{t\}) = \emptyset$, then $\bigcap_{\mathcal{I}, T \models Q} E(\emptyset) = Q(\emptyset)$, and thus $\emptyset \notin \text{RCQ}^w(Q, D_m, V)$, which contradicts the assumption. Moreover, it is easy to verify that $Q(\{t\})$ consists of a single tuple, and all $Q(\{t\})$ must return the same answer tuple, say $u$. Let $T$ be a tuple in $T$ such that $\text{Mod}(\tau, D_m, V) \neq \emptyset$. First note that $\text{Mod}(\tau, D_m, V)$ is a subset of $\{t \mid \{t\}, D_m \models Q\}$ and moreover, for each $t \in \text{Mod}(\tau, D_m, V)$ we have that $Q(\{t\}) = u$. We show that $\{\tau\}$ is in $\text{RCQ}^w(Q, D_m, V)$. To this aim, we only need to show that the intersection of $Q(\{t, s\})$, where $t \in \text{Mod}(\tau, D_m, V)$ and $\{t, s\}, D_m \models Q$, is equal to $u$. In fact, we show a stronger result: $Q(\{t, s\}) = u$ for each $t$ and $s$ as above. Suppose by contradiction that there exists a valuation $\mu'$ of $T_Q$ with values from $t$ and $s$ such that $\mu'(u_Q) \neq u$. However, every variable $x \in u_Q$ such that $\mu'(x) \neq u[x]$ is witnessed by an attribute either in $t$ or $s$ (or both), as specified by $\mu'$. Observe, however,
that also \( \{s\}, D_m \models V \) and thus also \( Q(\{s\}) = u \). This implies in turn, that \( \mu'(x) \) is already witnessed by a valuation of \( Q \) with values in \( t \), or by a valuation of \( Q \) with values in \( s \), both of which result in \( u \). Thus \( \mu'(u_Q) \neq u \) cannot be true. A contradiction.

From this lemma, it follows that we only need to consider \( c \)-instances \( T \) such that either \( T = \emptyset \) or \(|T| = 1\). Furthermore, for \( T \) with \(|T| \leq 1 \), the problem of testing minimality reduces to testing whether \( \emptyset \in \text{RCQ}^w(\{Q, D, m\}, V) \).

**Lower bound.** We show that for ground instances \( \text{MNP}^w(\text{CQ}) \) is \( \text{coDP-hard} \) by reduction from the complement of the \( \text{SAT-UNSAT} \) problem, which is \( \text{DP-complete} \) (cf. [Papadimitriou 1994]). An instance of \( \text{SAT-UNSAT} \) is to determine whether for a pair of 3SAT-instances \((\phi, \phi')\), \( \phi \) is satisfiable and \( \phi' \) is not satisfiable. Here \( \phi = C_1 \land \cdots \land C_r \) and \( \phi' = C'_1 \land \cdots \land C'_s \), i.e., for each \( i \in [1, r] \) (resp. \( i \in [1, s] \)), clause \( C_i \) (resp. \( C'_i \)) is of the form \( \ell_1 \lor \ell_2 \lor \ell_3 \), where for each \( i \in [1, 3] \), \( \ell_i \) is either a variable or the negation of a variable in \( X = \{x_1, \ldots, x_n\} \).

Given an instance of the latter problem, we define a database schema \( \mathcal{R} \), a ground instance \( I \) of \( \mathcal{R} \), master data \( D_m \), a set \( V \) of CCs and a CQ query \( Q \) such that \( I \) is a minimal weakly complete instance for \( Q \) relative to \((D_m, V)\) if and only if \( \phi \) is not satisfiable or \( \phi' \) is satisfiable.

(a) The database schema \( \mathcal{R} \) consists of a single relation \( R(X_1, \ldots, X_n, X'_1, \ldots, X'_n, Y) \) and we set the instance \( I \) to be the empty set.

(b) The master data \( D_m \) consists of two relations \((I_{(0,1)} = \{(0), (1)\}, I_0 = \emptyset)\).

(c) The set \( V \) of CCs consists of the following: (i) a constraint enforcing that every attribute in \( R \) takes values from \( I_{(0,1)} \); (ii) for each clause \( C_i \), for \( i \in [1, r] \), and each truth assignment \( \mu_X \) of the variables in \( C_i \) that makes \( C_i \) false, we add a selection condition of the form \( \sigma_{u_X}(R) \subseteq I_0 \), ensuring that the projection of \( R \) on \( X_1, \ldots, X_n \) contains no tuples satisfying the selection condition. For example, for \( C = \bar{x}_1 \lor \bar{x}_2 \lor x_3 \) and \( \mu_X = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\} \), we have the constraint \( \sigma_{x_1=\bar{1}} \land x_2 \land x_3 = 0 \subseteq I_0 \); and finally, (iii) for each \( C'_i \), for \( i \in [1, s] \), we add similar constraints to ensure that no tuples in \( R \) satisfy \( C'_i \). In addition, these constraints always include \( Y = 1 \). For example, for \( C'_1 = x_3 \lor x_4 \lor \bar{x}_5 \) and \( \mu_X = \{x_3 \mapsto 1, x_4 \mapsto 0, x_5 \mapsto 1\} \), we have the constraint \( \sigma_{x_3=\bar{1}} \land x_4 \land x_5 = 0 \subseteq I_0 \); note that there are \( r + s \) such CCs. Observe the following: (i) \( \{t[y] \mid \{t\}, D_m \models V\} = \emptyset \) is empty if and only if \( \phi \) is not satisfiable; (ii) it is \( \{(0)\} \) if \( \phi \) is satisfiable and \( \phi' \) is unsatisfiable; and (iii) it is \( \{(0), (1)\} \) if both \( \phi \) and \( \phi' \) are satisfiable. Moreover, clearly \( I = \emptyset \) is partially closed relative to \((D_m, V)\).

(d) Finally, the CQ query \( Q \) simply returns \( \pi_Y(R) \).

We next show that \( I = \emptyset \) is in \( \text{RCQ}^w(Q, D, m, V) \) if and only if \( \phi \) is unsatisfiable or \( \phi' \) is satisfiable. Note that \( I = \emptyset \) is in \( \text{RCQ}^w(Q, D, m, V) \) if and only if \( \bigcap_{t' \neq \emptyset, (t', D_m) \models V} \{Q(t')\} = \emptyset \). Since \( Q \) is monotonic, the latter is equivalent to \( \bigcap_{(t, D_m) \models V} \{Q(t)\} = \emptyset \), which only happens when either \( \{t[y] \mid \{t\}, D_m \models V\} = \emptyset \) or \( \{t[y] \mid \{t\}, D_m \models V\} \neq \emptyset \) and moreover there exist two elements in this set. Indeed, in this case \( Q(t_1) = \{t_1\} \neq \{t_2\} = Q(t_2) \). Hence, the only case when \( I \notin \text{RCQ}^w(Q, D, m, V) \) is when \( \phi \) is satisfiable and \( \phi' \) is not satisfiable.

**Upper bound.** Based on Lemma 5.7, the following algorithm decides whether a given \( c \)-instance \( T \) is a minimal instance weakly complete for a CQ query \( Q \) relative to \((D_m, V)\):

1. Check whether \(|T| > 1\). If so return \( \text{“no”} \); otherwise continue.
2. Check whether \( \emptyset \in \text{RCQ}^w(Q, D, m, V) \) and \( T = \emptyset \). If so return \( \text{“yes”} \).
3. Check whether \( \emptyset \notin \text{RCQ}^w(Q, D, m, V) \) and \(|T| = 1\). If so return \( \text{“yes”} \); otherwise return \( \text{“no”} \).
It is in \( \text{NP} \) to check whether \( \emptyset \notin \text{RCQ}^w(Q, D_m, V) \). Indeed, \( \emptyset \notin \text{RCQ}^w(Q, D_m, V) \) if and only if \( \bigcap_{t \in \text{Ext}(\emptyset)} Q(t') \neq \emptyset \). One can verify that the latter holds if and only if \( \bigcap_{(t_1, D_m) \in V} Q(\{t\}) \neq \emptyset \). As discussed before, we have that \( \emptyset \notin \text{RCQ}^w(Q, D_m, V) \) if and only if for every pair of tuples \( t_1 \) and \( t_2 \) that possibly refer to the same tuple, \( Q(t_1) = Q(t_2) \) if \( \{t_1\}, D_m \models V \) and \( \{t_2\}, D_m \models V \). Accordingly, we give a \( \text{coNP} \) algorithm for its complement problem, i.e., checking whether \( \emptyset \in \text{RCQ}^w(Q, D_m, V) \), which returns “no” if and only if there exist a pair of tuples \( t_1 \) and \( t_2 \) such that \( Q(t_1) \neq Q(t_2) \) if \( \{t_1\}, D_m \models V \) and \( \{t_2\}, D_m \models V \). More specifically, the algorithm works as follows.

1. Guess one pair of tuples \( (t_1, t_2) \) with values in \( \text{Adom} \).
2. Check whether \( \{t_1\}, D_m \models V \) and \( \{t_2\}, D_m \models V \). If so continue; otherwise reject the guess. This can be done in \( \text{PTIME} \) for one-tuple instances.
3. Check whether \( Q(t_1) \neq Q(t_2) \). If so return “no”; otherwise reject the guess. This can be done again in \( \text{PTIME} \) for one-tuple instances.

It is in \( \text{coNP} \) to check whether \( \emptyset \in \text{RCQ}^w(Q, D_m, V) \), and thus it is in \( \text{NP} \) to check whether \( \emptyset \notin \text{RCQ}^w(Q, D_m, V) \). Thus the algorithm for deciding \( \text{MINP}^w(CQ) \) is in \( \text{coNP} \cap \text{NP} \).

In the proof above we consider only those \( T \) such that \( \text{Mod}(T, D_m, V) \neq \emptyset \). However, without assuming \( \text{Mod}(T, D_m, V) \neq \emptyset \), \( \text{MINP}^w(CQ) \) is still in \( \text{coNP} \). Indeed, it is in \( \text{NP} \) to check whether \( \text{Mod}(T, D_m, V) \neq \emptyset \) since \( |T| = 1 \). That is, adding an extra step to check whether \( \text{Mod}(T, D_m, V) \neq \emptyset \) to the algorithm will not complicate the analysis.

### 6. Viable Relative Information Completeness

We next investigate \( \text{RCQP} \), \( \text{RCQ} \) and \( \text{MINP} \) for viably complete \( c \)-instances, denoted by \( \text{RCQP}^v \), \( \text{RCQ}^v \) and \( \text{MINP}^v \), respectively. That is, we now focus on databases that can be made relatively complete when their missing values are correctly instantiated. In this model we provide complexity results on these problems, for various query languages. The results tell us that missing values complicate the analysis of these problems, to an extent. As opposed to their counterparts in the weak model, the complexity bounds are not very diverse (we defer the proofs of the results of this section to the electronic appendix). We use \( \text{RCQ}^v(Q, D_m, V) \) to represent the set of all viably complete instances.

In contrast to Theorem 4.1, \( \text{RCQP}^v(CQ) \) for \( c \)-instances is \( \Sigma_2^p \)-complete rather than \( \Pi_2^p \)-complete. Here \( \text{RCQP}^v(CQ) \) remains undecidable, as opposed its counterpart in the weak model (Theorem 5.1).

**Theorem 6.1.** For \( c \)-instances, \( \text{RCQP}^v(L_Q) \) is

- undecidable when \( L_Q \) is \( \text{FO} \) or \( \text{FP} \), and
- \( \Sigma_2^p \)-complete when \( L_Q \) is \( \text{CQ} \), \( \text{UCQ} \) or \( \exists \text{FO}^+ \).

The complexity is unchanged when \( D_m \) and \( V \) are fixed.

In contrast to Theorem 5.1, \( \text{RCQP}^v(L_Q) \) is no longer trivial for viably complete \( c \)-instances when \( L_Q \) is \( \text{FP} \). One can verify that Lemma 4.4 still holds in this setting. As a result, \( \text{RCQP}^v \) for relatively viably complete \( c \)-instances coincides with \( \text{RCQP}^v \) for ground instances. For the latter, the complexity results are already established by Theorem 4.5. From these the corollary below follows.

**Corollary 6.2.** For \( c \)-instances, \( \text{RCQP}^v(L_Q) \) is

- undecidable when \( L_Q \) is \( \text{FO} \) or \( \text{FP} \), and
- \( \text{NEXPTIME} \)-complete when \( L_Q \) is \( \text{CQ} \), \( \text{UCQ} \) or \( \exists \text{FO}^+ \).

The complexity is unchanged when \( D_m \) and \( V \) are fixed.
For $c$-instances, $\text{MINP}^c(\mathcal{L}_Q)$ becomes $\Sigma^b_3$-complete for $\text{CQ}$, $\text{UCQ}$ or $\exists \text{FO}^+$, rather than $\Pi^b_3$-complete as in the strong model. The complexity bound is rather robust: it is the same for $\text{CQ}$, $\text{UCQ}$ and $\exists \text{FO}^+$, as opposed to their counterparts in the weak model.

**Corollary 6.3.** $\text{MINP}^c(\mathcal{L}_Q)$ is

— undecidable for $c$-instances and for ground instances when $\mathcal{L}_Q$ is FO or FP, and
— $\Sigma^b_3$-complete for $c$-instances and $\Pi^b_2$-complete for ground instances, when $\mathcal{L}_Q$ is $\text{CQ}$, $\text{UCQ}$ or $\exists \text{FO}^+$.

The complexity is unchanged when $D_m$ and $V$ are fixed.

7. TRACTABLE SPECIAL CASES

The results of Sections 4, 5 and 6 tell us that RCDP, RCQP and MINP have rather high complexity. For practical use to emerge from the study of relative information completeness, we need to develop effective and efficient heuristic algorithms RCDP, RCQP and MINP, and moreover, identify their special cases that are practical and tractable.

In practice, we often deal with fixed sets of queries and constraints. That is, the queries and CCs are predefined in advance, and only the underlying databases and master data may vary. Indeed, we often have a fixed query load, e.g., in e-commerce, certain fixed Web forms are used, which are fixed queries in which some designated variables may take various value parameters. Moreover, people typically first design constraints based on schemas, and then populate and maintain database instances. This highlights the need for studying the data complexity of relative information completeness (see [Abiteboul et al. 1995] for details about data complexity).

In this section we identify tractable cases for RCDP, RCQP and MINP when queries $Q$ and CCs $V$ are fixed, while the underlying databases $D$ and master data $D_m$ vary, for $c$-instances in the strong, weak and viable completeness models. That is, we study their tractable cases under data complexity (the proofs of the results are given in the electronic appendix). In contrast, Sections 4, 5 and 6 have studied the combined complexity of the problems, when data, queries and CCs may all vary.

One might be tempted to think that the data complexity analyses of these problems would be much simpler. Unfortunately, the study of data complexity is non-trivial! For ground instances in the strong completeness model, the data complexity of RCDP and MINP has recently been studied [Cao et al. 2014]. It is shown that RCDP and MINP remain undecidable for FO even in the absence of CCs, and for FP when $V$ is a set of FDs! These undecidability results obviously carry over to $c$-instances in the strong model. While these problems are in $\text{PTIME}$ for $\text{CQ}$, $\text{UCQ}$ and $\exists \text{FO}^+$ when ground instances are considered, the $\text{PTIME}$ algorithms of [Cao et al. 2014] no longer work on $c$-instances. To the best of our knowledge, no previous work has studied RCQP for ground instances or $c$-instances, or RCDP and MINP for $c$-instances.

**The relatively complete database problem.** To get tractable cases for $c$-instances, we consider $c$-instances with a constant number of variables. That is, when our databases have a small number of missing (null) values. Under this condition and data complexity, RCDP becomes tractable for most positive query languages.

**Corollary 7.1.** For $c$-instances with a constant number of variables, and for fixed query $Q$ and a fixed set $V$ of CCs,

— $\text{RCDP}^s$ and $\text{RCDP}^v$ are in $\text{PTIME}$ for $\text{CQ}$, $\text{UCQ}$ and $\exists \text{FO}^+$; and
— $\text{RCDP}^w$ is in $\text{PTIME}$ for $\text{CQ}$, $\text{UCQ}$, $\exists \text{FO}^+$ and FP.

**The relatively complete query problem.** When we use INDs as CCs, i.e., for CCs of the form $q \subseteq p$ when $q$ and $p$ are both projection queries, $\text{RCQP}^s$ and $\text{RCQP}^v$ become...
Capturing Missing Tuples and Missing Values

much simpler. The positive results hold even when the set $V$ of CCs is not fixed. Moreover, $\text{RCDP}^w$ is in constant time for FP for general CCs defined in CQ, by Theorem 5.4.

**Corollary 7.2.** For fixed queries

— $\text{RCQP}^s$ and $\text{RCQP}^v$ are in $\text{PTIME}$ for CQ, UCQ and $\exists FO^+$ when CCs are INDs; and
— $\text{RCQP}^w$ is in $O(1)$ time for CQ, UCQ, $\exists FO^+$ and FP.

The minimality problem. Similar to $\text{RCDP}$, we get tractable cases of $\text{MINP}$ when queries and CCs are fixed, for $c$-instances with a constant number of variables.

**Corollary 7.3.** For $c$-instances with a constant number of variables, and for fixed query $Q$ and a fixed set $V$ of CCs,

— $\text{MINP}^s$ and $\text{MINP}^v$ are in $\text{PTIME}$ for CQ, UCQ and $\exists FO^+$; and
— $\text{MINP}^w$ is in $\text{PTIME}$ for CQ.

8. CONCLUSIONS

We have proposed three models to specify the relative information completeness of databases in the presence of both missing values and missing tuples. We have studied the interaction between the analysis of relative completeness and the analysis of data consistency. We have also identified three problems associated with relative completeness, namely, $\text{RCQP}$, $\text{RCDP}$ and $\text{MINP}$. For a variety of query languages, we have established upper and lower bounds on these problems, all matching, in each of the three completeness models, both for $c$-instances and for ground instances. We have also identified tractable cases of these problems under data complexity. We expect that these results will help database users decide whether their queries can find complete answers in a database, and moreover, help developers of MDM or databases identify a minimal amount of information to collect in order to answer queries commonly issued.

The main complexity results are summarized in Table I, annotated with their corresponding theorems. From the table we can see that different combinations of query languages, completeness models, and the presence and the absence of missing values lead to a spectrum of decision problems with different complexity bounds.

The study of relative information completeness is still in its infancy. An open issue is about the complexity of $\text{RCQP}$ for FO in the weak model. We only know that it is undecidable for ground instances, and our conjecture is that it is also undecidable for $c$-instances. Another open issue concerns whether the complexity bounds remain intact when master data and CCs are fixed. A third topic is to develop representation systems for relatively complete databases, possibly under the semantics introduced by [Libkin 2014]. A fourth topic is to figure out the impact of other constraints on the analysis of relative completeness, such as tuple generating dependencies. Finally, to make practical use of the study, we need to develop efficient heuristic algorithms for the problems with certain performance guarantees.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES


ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYYY.
Capturing Missing Tuples and Missing Values


ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.


A. PROOFS OF SECTION 5

Proofs of Theorem 5.4

THEOREM 5.4. RCQP\(^w\)\((\mathcal{L}_Q)\) is
— undecidable for ground instances if \(\mathcal{L}_Q\) is FO, and
— decidable in \(O(1)\)-time for \(c\)-instances and ground instances when \(\mathcal{L}_Q\) is FP, CQ, UCQ or \(\exists FO^*\).

The complexity is unchanged when \(D_m\) and \(V\) are fixed.

PROOF. We start with a proof of the undecidability of RCQP\(^w\)\((FO)\) for ground instances. We then show that RCQP\(^w\)\((\mathcal{L}_Q)\) is trivially decidable when \(\mathcal{L}_Q\) is CQ, UCQ, \(\exists FO^*\) or FP, for ground instances and for \(c\)-instances.

(1) When \(\mathcal{L}_Q\) is FO. We show that RCQP\((FO)\) is undecidable for ground instances by reduction from the satisfiability problem for FO. Given an FO query \(q\), we define \(Q, D_m\) and \(V\), such that \(q\) is not satisfiable if and only if RCQP\(^w\)\((Q, D_m, V)\) is non-empty. By Lemma 3.2 we may assume w.l.o.g. that the query \(q\) is defined on a single relation \(R\).

The query \(Q\) is defined over a database schema \(\mathcal{R} = (R', E_1, E_2)\), where \(R'\) extends \(R\) by adding an attribute \(A\) with an infinite domain; and \(E_1\) and \(E_2\) are unary relations, each having a single attribute with an infinite domain. Then for an instance \(\mathcal{I} = (I, I_{E_1}, I_{E_2})\) of \(\mathcal{R}\), we define \(Q(\mathcal{I})\) as follows:

\[
Q(\mathcal{I}) = \begin{cases} 
\{(b)\} & \text{if } \forall a(q(\pi_R(\sigma_{A=a}(I))) = \emptyset) \\
\{(c)\} & \text{if } \exists a(q(\pi_R(\sigma_{A=a}(I))) \neq \emptyset) \text{ and } I_{E_1} = I_{E_2} \\
\{(d)\} & \text{if } \exists a(q(\pi_R(\sigma_{A=a}(I))) \neq \emptyset) \text{ and } I_{E_1} \neq I_{E_2}
\end{cases}
\]

where \(b, c\) and \(d\) are distinct constants. We define \(D_m\) to be an empty relation, and \(V\) to be the empty set.

We show that \(q\) is satisfiable if and only if RCQP\((Q, D_m, V)\) is empty.

\[\Rightarrow\] First assume that \(q\) is satisfiable, i.e., there exists an instance \(I\) of \(R\) such that \(q(I) \neq \emptyset\). We next show that RCQP\(^w\)\((Q, D_m, V)\) is empty, i.e., for every instance \(\mathcal{I}' = (I', I_{E_1}', I_{E_2}')\) of \(\mathcal{R}\), \(\mathcal{I}' \notin \text{RCQP}^w\((Q, D_m, V)\) if \((\mathcal{I}', D_m) \models V\). To achieve this, we construct two extensions of \(\mathcal{I}'\): \(\mathcal{I}'_1 = (I' \cup \{\{A = e\} \times I\}), I_1, I_2\), and \(\mathcal{I}'_2 = (I' \cup \{\{A = e\} \times I\}, I_1', I_2')\), where \(e\) is a distinct constant not appearing in \(I\). (2) \(I_1\) and \(I_2\) are extensions of \(I_{E_1}\) and \(I_{E_2}\), respectively, such that \(I_1 = I_2\), and (3) \(I_1'\) and \(I_2'\) are extensions of \(I_{E_1}'\) and \(I_{E_2}'\), respectively, such that \(I_1' \neq I_2'\). Obviously, by the definition of \(Q\), \(Q(\mathcal{I}'_1) = \{(c)\}\), \(Q(\mathcal{I}'_2) = \{(d)\}\). Then \(\bigcap_{I' \in \text{Ext}(\mathcal{I}_0)} Q(\mathcal{I}') = \emptyset \neq Q(\mathcal{I}_0)\). Thus \(\mathcal{I}'\) is not in RCQP\(^w\)\((Q, D_m, V)\), and hence RCQP\((Q, D_m, V)\) is empty.

\[\Leftarrow\] Conversely, suppose \(q\) is not satisfiable. Then for every instance \(\mathcal{I}\) of \(\mathcal{R}\), \(Q(\mathcal{I}) = \{(b)\}\) and hence, \(\mathcal{I}\) is in RCQP\(^w\)\((Q, D_m, V)\). That is, RCQP\(^w\)\((Q, D_m, V)\) is not empty.
(2) When \( L_Q \) is FP. We show that for every FP query \( Q \) defined over schema \( \mathcal{R} \), any set \( V \) of CCs and master data \( D_m \), \( \text{RCQ}^m(Q, D_m, V) \) is non-empty, i.e., there always exists a database (ground instance) that is complete for \( Q \) relative to \( (D_m, V) \). From this it follows immediately that \( \text{RCQP}^m(L_Q) \) is trivially decidable for \( c \)-instances in the weak model, when \( L_Q \) is CQ, UCQ, \( \exists \text{FO}^* \) or FP.

These languages have the following properties: for every query \( Q \) in FP,

(a) **Monotonicity**: for every pair of instances \( I_1 \) and \( I_2 \), if \( I_1 \subseteq I_2 \) then \( Q(I_1) \subseteq Q(I_2) \);

(b) **Small Extension**: for every instance \( I, I \in \text{RCQ}^m(Q, D_m, V) \) if and only if \( Q(I) = \cap_{t \in \text{Ext}(I)} Q(I \cup \{t\}) \), i.e., it suffices to consider partially closed extensions of \( I \) by including a single tuple. Indeed, \( I \in \text{RCQ}^m(Q, D_m, V) \) if and only if \( Q(I) = \cap_{t \in \text{Ext}(I)} Q(I') \). Obviously, \( \cap_{t \in \text{Ext}(I)} Q(I') \subseteq \cap_{t \in \text{Ext}(I)} Q(I \cup \{t\}) \), and \( \cap_{t \in \text{Ext}(I)} Q(I \cup \{t\}) \subseteq \cap_{t \in \text{Ext}(I)} Q(I') \) since FP is monotonic.

Recall that by Lemma 3.2, we may assume that \( \mathcal{R} \) consists of a single relation \( R(A_1, \ldots, A_n) \). We now leverage the two previous properties to construct a ground instance \( I_0 \) of schema \( R \) such that \( I_0 \) is in \( \text{RCQ}^m(Q, D_m, V) \). Observe that the FP query \( Q \) can be written as an equivalent query \( \text{lfp}(Q'(S)) \), where \( \text{lfp} \) is the inflationary fixpoint operator (see, e.g., [Abiteboul et al. 1995] for more details about \( \text{lfp} \)), and \( Q' \) is a UCQ \( Q_1 \cup \cdots \cup Q_m \). Given an instance \( I \) of \( R \), the query \( Q \) computes relations \( S \) as follows:

\[
S_0 = \emptyset; \quad S_{j+1} = S_j \cup Q'(S_j, I);
\]

Then \( Q(I) \) is the relation \( S_l \) when the sequence converges, i.e., when \( S_l = S_{l+1} \).

To construct \( I_0 \), we first define its active domain, denoted by \( \text{Adom}(I_0) \), \( \text{Adom}(I_0) \) in the same way as its counterpart defined in the proof of Theorem 4.1. We then construct a set \( L \) of tuples: \( \text{Adom}(A_1) \times \cdots \times \text{Adom}(A_n) \), where for each \( i \in \{1, n\} \), \( \text{Adom}(A_i) \) is either \( D_i \) if \( \text{dom}(A_i) \) is finite domain, or \( \text{Adom}(I_0) \) otherwise.

We define \( I_0 \) to be a maximum subset of \( L \) such that \( (I_0, D_m) \models V \), i.e., adding any more tuples from \( L \) would violate \( V \). There are possibly multiple such maximum subsets of \( L \), and any one of them will serve our purpose.

We show that \( I_0 \) is in \( \text{RCQP}^m(L_Q) \), i.e., \( Q(I_0) = \cap_{t \in \text{Ext}(I_0)} Q(I') \). By the Monotonicity property of FP mentioned earlier, for every \( I' \in \text{Ext}(I_0) \), \( Q(I_0) \subseteq Q(I') \). Hence it suffices to show that \( \cap_{t \in \text{Ext}(I_0)} Q(I') \subseteq Q(I_0) \). By the Small Extension property of FP, it suffices to consider \( I' = I_0 \cup \{t\} \), where \( t \) is a tuple not in \( I_0 \).

In light of these, we only need to show that \( \cap_{t \in \text{Ext}(I_0)} Q(I_0 \cup \{t\}) \subseteq Q(I_0) \).

First, we can treat \( I_0 \cup \{t\} \) as a \( c \)-instance \( T \), where \( t \) is denoted by a tuple template \( (x_1, \ldots, x_n) \) with distinct variables. Let \( t_0 \) be any tuple in \( \cap_{t \in \text{Ext}(I_0)} Q(I_0 \cup \{t\}) \). With the weakly completeness semantics, \( c \)-instances are a strong representation system for FP [Grahne 1991], i.e., there exists a \( c \)-instance \( T_Q \) representing \( Q(T) \) such that \( t_0 \) is in the certain answer of \( Q(T) \). Observe that any tuple template \( t_0 \) in \( T_Q \) consists of values in \( \text{Adom}(I_0) \) or variables in \( \{x_1, \ldots, x_n\} \). However, if \( t_0 \) contains a variable, then it cannot be part of \( \cap_{t \in \text{Ext}(I_0)} Q(I') \). Indeed, one can instantiate the variable with distinct values. In light of this, we only need to consider \( t_0 \) composed of values in \( \text{Adom}(I_0) \).

We next show that \( t_0 \) is in \( Q(I_0) \). Consider any \( I' = I_0 \cup \{t\} \), where \( t = (a_1, \ldots, a_n) \), and some \( a_i \) may not be in \( \text{Adom}(I_0) \), such that \( (I', D_m) \models V \) and \( t \notin I_0 \). For all \( k \geq 1 \), note that \( S_k = S_{k-1} \cup Q'(S_{k-1}, I_0 \cup \{t\}) \), where \( Q \) is treated as query \( \text{lfp}(Q'(S)) \) as mentioned above. Define \( S_k' \) as follows: for each tuple \( s = (p_1, \ldots, p_m) \in S_k \), replace \( s \) with another tuple \( s' = (p_1', \ldots, p_m') \) such that for every \( i \in [1, m] \), \( p_i' = p_i \) if \( p_i \) is in \( \text{Adom}(I_0) \), and otherwise set \( p_i' \) to be a distinct value in \( \text{New} \). Similarly, we construct \( t' \) from \( t \). We preserve the equality on variables in \( Q' \). Then we can show the following.
**Lemma A:** $S'_k \subseteq Q(I_0)$. □

**Proof.** We show the lemma by induction on $k$. When $k = 1$, $S_1 = Q'(\emptyset, I_0 \cup \{t\}) = \bigcup_{j \in \{1, \ldots, m\}} Q_j(\emptyset, I_0 \cup \{t\})$ (recall that $Q'$ is a UCQ $Q_1 \cup \cdots \cup Q_m$). Then $S'_1 = \bigcup_{j \in \{1, \ldots, m\}} Q_j(\emptyset, I_0 \cup \{t\})$, where $Q_j$ is a query in CQ as mentioned above. It can be readily verified that $(I_0 \cup \{t\}, D_m) \models V$ since $(I_0 \cup \{t\}, D_m) \models V$. Hence $t'$ is in $I_0$ since $I_0$ is the maximum instance taking values from $\text{Adom}(I_0)$ such that $(I_0, D_m) \models V$. Thus $S'_1 \subseteq Q(I_0)$. Assume that $S'_k \subseteq Q(I_0)$ for $k \leq l$. When $k = l + 1$, $S_{l+1} = S_l \cup Q'(S_l, I_0 \cup \{t\})$. By the induction hypothesis, we know that $S'_l \subseteq Q(I_0)$. Then $S'_{l+1} = S'_l \cup Q'(S'_l, I_0 \cup \{t\}) \subseteq Q(I_0) \cup Q'(Q(I_0), I_0 \cup \{t\}) \subseteq Q(I_0)$, since $Q'(Q(I_0), I_0 \cup \{t\}) \subseteq Q(I_0)$. □

We next show that $t_0$ is in $Q(\emptyset)$. Consider any $I' = I_0 \cup \{t\}$ given above and tuple $t_0$ in $Q(I')$. Assume w.l.o.g. that $t_0$ is in $S_k = S_{k-1} \cup Q'(S_{k-1}, I_0 \cup \{t\})$, where $k \geq 1$. Consider the following two cases. (1) If $t_0 \in S_{k-1}$, then $t_0$ must be in $S'_{k-1}$ since $S'_{k-1}$ contains all tuples $t$ in $S_{k-1}$ that take values from $\text{Adom}(I_0)$. Thus $t_0 \in Q(I_0)$ as discussed above. (2) If $t_0 \in Q'(S_{k-1}, I_0 \cup \{t\})$ and $t_0 \notin S_{k-1}$, then $t_0$ is in $Q'(S'_{k-1}, I_0 \cup \{t\})$. Indeed, there exists $i \in \{1, \ldots, m\}$ such that $t_0 \in Q_i(S_{k-1}, I_0 \cup \{t\})$, where $Q_i$ is a query in CQ as mentioned above. Write $Q_i$ in the SP normal form $\pi_X(\sigma_C E)$, where $E$ is the Cartesian product of (multiple occurrences of) $S_{k-1}$ and $R$. Then there must exist $t_1, \ldots, t_l$ such that each $t_s$ is in $S_{k-1}$ or is in $R$ for $s \in \{1, \ldots, l\}$, such that $t_0 = \pi_X(\sigma_C(t_1 \times \cdots \times t_l))$. For each such $t_s$ (no matter whether it equals $t$ or is from $S_{k-1}$), we construct tuple $t'_s$ in the same way as shown above, with values from $\text{Adom}(I_0)$. Obviously, $t_0$ can be still generated when replacing each such $t_s$ with $t'_s$. In other words, $t_0$ is in $Q(I_0)$. □

**B. PROOFS OF SECTION 6**

**Proofs of Theorem 6.1**

**Theorem 6.1.** For $c$-instances, RCDP$^c(L_Q)$ is

— undecidable when $L_Q$ is FO or FP, and
— $\Sigma_2^p$-complete when $L_Q$ is CQ, UCQ or $\exists$FO$^+$.

The complexity is unchanged when $D_m$ and $V$ are fixed.

**Proof.** We show that for viably complete $c$-instances, RCDP$^c(L_Q)$ remains undecidable when $L_Q$ is FO or FP, but it becomes $\Sigma_2^p$-complete when $L_Q$ is CQ, UCQ or $\exists$FO$^+$.

(1) When $L_Q$ is FO. The undecidability follows from the fact that RCDP$^c$(FO) is already undecidable for ground instances [Fan and Geerts 2009], a special case of $c$-instances. Indeed, for every ground instance $I$, master data $D_m$ and set $V$ of CCs, $\text{Mod}(I, D_m, V)$ consists of a single instance, namely, $I$ itself, if $(I, D_m) \models V$. Hence the following problems are equivalent: (a) deciding whether $I$ is in $\text{RCQ}^c(Q, D_m, V)$ for a query $Q$, and (b) whether there exists an instance in $\text{Mod}(I, D_m, V)$ that is in $\text{RCQ}^c(Q, D_m, V)$.

(2) When $L_Q$ is FP. Observe that RCDP$^c$(FP) is undecidable for ground instances (see the proof of Theorem 4.1). From this it follows that RCDP$^c$(FP) is undecidable for viably complete $c$-instances.

(3) When $L_Q$ is CQ, UCQ or $\exists$FO$^+$. It suffices to show that RCDP$^c(\exists$FO$^+)$ is in $\Sigma_2^p$ and that RCDP$^c$(CQ) is $\Sigma_2^p$-hard.

**Lower bound.** We show that RCDP$^c$(CQ) is $\Sigma_2^p$-hard by reduction from the $\exists \forall \exists \forall$ 3SAT-problem, which is $\Sigma_2^p$-complete (cf. Papadimitriou 1994). Given an instance $\varphi = \exists X \forall Y \exists Z \psi(X, Y, Z)$ of the latter problem, we construct the same $R$, $\mathcal{R}_m$, $D_m$ and $V$ as their counterparts given in the proof of $\text{MINP}^c$(CQ) in Theorem 4.8(2). Furthermore, the
First assume that

Conversely, assume that

L — undecidable when Mod

there exists

Adding any tuples to

I

R

instances of

that the instance

C

App–4 Capturing Missing Tuples and Missing Values

Finally, query Q is defined as follows:

Q(\bar{y}) = \exists \bar{z}, \bar{w} \left( Q_X(\bar{x}) \land Q_Y(\bar{y}) \land Q_Z(\bar{z}) \land Q_\psi(\bar{x}, \bar{y}, \bar{z}, \bar{w}) \land R_s(\bar{w}) \right),

where

Q_X = \bigwedge_{i \in [1..n]} R_X(i, x_i)

is to select from R_X the truth assignments for X; and query

Q_Y = R_{(0.1)}(y_1) \land \cdots \land R_{(0.1)}(y_m)

is to construct all possible truth assignments of variables in Y; similarly,

Q_Z = R_{(1.0)}(z_1) \land \cdots \land R_{(1.0)}(z_p).

Given a truth assignment (\mu_X, \mu_Y, \mu_Z) of (X, Y, Z), the subquery

Q_\psi(\mu_X, \mu_Y, \mu_Z, w)

is to evaluate \psi(\mu_X, \mu_Y, \mu_Z), and it records its truth value in w, which is either 0 or 1. Intuitively, query Q returns all truth assignments \mu_Y of Y under which \psi(\mu_X, \mu_Y, \mu_Z) evaluates to a truth value in I_s for given \mu_X, \mu_Y and \mu_Z.

We next verify the correctness of the reduction.

\[ \Rightarrow \] First assume that \varphi is true. We show that there exists a ground instance I \in \text{Mod}(T, D_m, V) that is complete. Since \varphi is true, there exists \mu_X such that for all \mu_Y, there exists \mu_Z that makes \psi true. Define a ground instance I \in \text{Mod}(T, D_m, V) such that the instance I_X of R_X encodes \mu_X, i.e.,

\[ I_X = \{ (1, \mu_X(x_1)), \ldots, (n, \mu_X(x_n)) \} \]

and the instances of R_{(0.1)}, R_{(1.0)}, R_n, R_i, and R_s are fixed as given above. Clearly, (I, D_m) \models V.

In addition, I is complete. Indeed, Q(I) consists of all possible truth assignments of Y. Adding any tuples to I either violates V or does not change the answer to Q.

\[ \Leftarrow \] Conversely, assume that \varphi is not true. Then for every \mu_X of X, there exists \mu_Y such that for every \mu_Z of Z, \psi is false. We show that for each ground instance I = (I_0, I_1, I_i, I_s) \in \text{Mod}(T, D_m, V),

where the instance I_X of R_X encodes a truth assignment of X, I is not in RCQ\psi(Q, D_m, V). Indeed, there exists at least one truth assignment of Y that is not in Q(I). However, if we extend I by adding (0) to I_s and refer to this extension as I', then one can readily show that (I', D_m) \models V and Q(I') consists of all truth assignments of Y. That is, Q(I) \neq Q(I'). Thus there exists no instance of T that is complete.

Upper bound. We prove the \Sigma^p_3-upper bound by providing an algorithm for testing the relative completeness of c-instances. We first consider queries in UCQ, and then extend the algorithm to \exists\Sigma^p_3. By Lemma 3.2, we assume w.l.o.g. that the queries are defined over a single relation schema. Given (T, \xi), D_m, V and a UCQ query Q, the algorithm returns “yes” if there exists a ground instance I \in \text{Mod}(T, D_m, V) such that I is complete relative to (D_m, V), and it returns “no” otherwise. More specifically, the algorithm does the following:

(1) Guess a valuation \mu of (T, \xi) with Adom (recall the definition of Adom from the proof of Theorem 4.1).

(2) Check whether \mu(T) \not\in \text{Mod}(T, D_m, V). If so, then reject the current guess. Otherwise, let I = \mu(T). Note that the checking can be done with an NP-oracle.

(3) We next call a \Sigma^p_3 oracle: check whether I is not in RCQ\psi(Q, D_m, V). If so, reject the guess. Otherwise return “yes”.

The overall complexity is thus NP^{\Sigma^p_3} or \Sigma^p_3. The correctness of the algorithm can be verified along the same lines as its counterpart in the proof of Theorem 4.1.

The algorithm given above can be extended to \exists\Sigma^p_3, in the same way as the extension given in the proof of Theorem 4.1.

\[ \square \]

Proofs of Corollary 6.2

\textbf{Corollary 6.2.} For c-instances, RCQP\psi(\mathcal{L}_Q) is

— undecidable when \mathcal{L}_Q is FO or FP, and

ACM Transactions on Database Systems, Vol. V, No. N, Article A, Publication date: January YYYY.
— \textsc{Nexptime}-complete when \( \mathcal{L}_Q \) is CQ, UCQ or \( \exists \mathcal{F} \mathcal{O}^+ \).

The complexity is unchanged when \( D_m \) and \( V \) are fixed.

\textbf{Proof.} It suffices to show that Lemma 4.4 holds for viably complete \( c \)-instances. For if it holds, then \( \text{RCQP}^{\exists}(\mathcal{L}_Q) \) for \( c \)-instances is equivalent for \( \text{RCQP}^{\forall}(\mathcal{L}_Q) \) for ground instances. As a result, the complexity bounds of \( \text{RCQP}^{\exists}(\mathcal{L}_Q) \) for ground instances established by Theorem 4.5 carry over to \( \text{RCQP}^{\forall}(\mathcal{L}_Q) \) for viably complete \( c \)-instances.

We next prove Lemma 4.4 for viably complete \( c \)-instances. If there exists a \( c \)-instance \( T \in \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \) with \(|T| \leq K\), then by the definition of viably complete \( c \)-instances, there exists a ground instance \( I \in \text{Mod}(T) \) such that \( I \) is also in \( \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \). Moreover, \(|I| \leq K\) since \( I = \mu(T) \) for some valuation of \( T \). Conversely, if there is a ground instance \( I \) in \( \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \) with \(|I| \leq K\), then there also exists a \( c \)-instance \( T \) in \( \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \) with \(|T| \leq K\), i.e., \( I \) itself. Hence there exists a viably complete \( c \)-instance \( T \) in \( \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \) with \(|T| \leq K\) if and only if there exists a ground instance \( I \) in \( \text{RCQP}^{\mathcal{F}}(Q, D_m, V) \) with \(|I| \leq K\).

\textbf{Proofs of Corollary 6.3}

\textbf{Corollary 6.3.} \( \text{MINP}^{\forall}(\mathcal{L}_Q) \) is

— undecidable for \( c \)-instances and for ground instances when \( \mathcal{L}_Q \) is \( \mathcal{F} \mathcal{O} \) or \( \mathcal{F} \mathcal{P} \), and

— \( \Sigma_3^{\forall} \)-complete for \( c \)-instances and \( \Sigma_3^{\mathcal{F}} \)-complete for ground instances, when \( \mathcal{L}_Q \) is CQ, UCQ or \( \exists \mathcal{F} \mathcal{O}^+ \).

The complexity is unchanged when \( D_m \) and \( V \) are fixed.

\textbf{Proof.} We first show that \( \text{MINP}^{\forall}(\mathcal{L}_Q) \) is undecidable when \( \mathcal{L}_Q \) is \( \mathcal{F} \mathcal{O} \) or \( \mathcal{F} \mathcal{P} \), and then show that it becomes \( \Sigma_3^{\forall} \)-complete for \( c \)-instances when \( \mathcal{L}_Q \) is CQ, UCQ or \( \exists \mathcal{F} \mathcal{O}^+ \).

For ground instances, it has already been shown in the proof of Theorem 4.8 that the problem is \( \Pi_3^{\mathcal{F}} \)-complete for CQ, UCQ and \( \exists \mathcal{F} \mathcal{O}^+ \).

(1) When \( \mathcal{L}_Q \) is \( \mathcal{F} \mathcal{O} \) or \( \mathcal{F} \mathcal{P} \). We show that \( \text{MINP}^{\forall}(\mathcal{L}_Q) \) remains undecidable when \( \mathcal{L}_Q \) is either \( \mathcal{F} \mathcal{O} \) or \( \mathcal{F} \mathcal{P} \), for ground instances. For ground instances, the undecidability of \( \text{MINP}(\mathcal{F} \mathcal{O}) \) and \( \text{MINP}(\mathcal{F} \mathcal{P}) \) has already been verified in the proof of Theorem 4.8.

(2) When \( \mathcal{L}_Q \) is CQ, UCQ or \( \exists \mathcal{F} \mathcal{O}^+ \). We show that \( \text{MINP}^{\forall}(\text{CQ}) \) is \( \Sigma_3^{\forall} \)-hard and that \( \text{MINP}^{\forall}(\exists \mathcal{F} \mathcal{O}^+) \) is in \( \Sigma_3^{\forall} \), for \( c \)-instances.

\textbf{Lower bound.} We show that \( \text{MINP}^{\forall}(\text{CQ}) \) is \( \Sigma_3^{\forall} \)-hard by reduction from the \( \exists \forall^* \exists \forall^* \text{3SAT} \) problem (see the proof of Theorem 4.8(3) for a description of this problem). Given \( \varphi = \exists \forall \forall \exists Z \psi \), we define the same \( R, D_m, V, Q \) and \( T \) as their counterparts given in the proof of \( \text{MINP}^{\forall}(\text{CQ}) \) in Theorem 4.8(2) except that in the \( c \)-instance \( T, I_s = \{(1)\} \).

We show that \( \varphi \) is true if and only if there exists \( I \in \text{Mod}(T) \) such that \( I \) is a minimal instance complete for \( Q \).

\[ \text{[=]} \] If \( \varphi \) is true, then there exists a truth assignment \( \mu_X \) of \( X \) such that for every truth assignment \( \mu_Y \) of \( Y \), \( \exists \psi(\mu_X, \mu_Y, z) \) is true. Let \( I_X^0 \) be a ground instance of \( T_X \) that agrees with \( \mu_X \), and \( I = (I_{0,1}, I_X, I_Y, I_\psi, I_0, I_\psi, I_s) \). The argument given in the proof of \( \text{MINP}^{\forall}(\text{CQ}) \) for Theorem 4.8 suffices to show that \( I \) is in \( \text{RCQP}^{\forall}(Q, D_m, V) \). Furthermore, \( I \) is minimal, since removing any tuple from \( I \) makes the answer to \( Q \) empty, and hence makes the instance incomplete. By the definition of viably complete minimal \( c \)-instances, \( T \) is indeed a minimal \( c \)-instance in \( \text{RCQP}^{\forall}(Q, D_m, V) \).

\[ \text{[\leq]} \] Conversely, if \( \varphi \) is false, then for all truth assignments \( \mu_X \) of \( X \), there exists a truth assignment \( \mu_Y \) of \( Y \), such that for all truth assignments \( \mu_Z \) of \( Z \), \( \psi \) is false. Hence for
every instance $I = (I_{0,1}, I_\land, I_\lor, I_\neg, I_X, I_s)$ of $T$, where $I_X$ encodes a truth assignment of $X$, there exists at least one truth assignment of $Y$ that is not in $Q(I)$. However, let $I' = (I_{0,1}, I_\land, I_\lor, I_\neg, I_X, I'_s)$, where $I'_s = \{(1), (0)\}$. Obviously, $I'$ is in Ext$(T)$ and $Q(I')$ consists all truth assignments of $Y$. That is, $I$ is not in RCQ$^v(Q, D_m, V)$.

In the reduction above, only fixed $D_m$ and $V$ are used. Hence MINP$^v$(CQ), MINP$^v$(UCQ) and MINP$^v$(\exists\text{FO}+) remain $\Sigma_3^p$-hard when $D_m$ and $V$ are fixed.

Upper bound. We show that MINP$^v$(\exists\text{FO}+) is in $\Sigma_3^p$ for $c$-instances by giving a $\Sigma_3^p$ algorithm that, taking a $c$-instance $T$, a query $Q$ in $\exists\text{FO}^+$, master data $D_m$ and a set $V$ of CCs as input, returns “yes” if and only if $T$ is a minimal $c$-instance in RCQ$^v(Q, D_m, V)$. The algorithm works as follows.

(1) Guess a valuation $\mu$ of $T$ with values in Adom. Let $I = \mu(T)$.

(2) Test the following:
   a) whether $I \not\in \text{Mod}(T, D_m, V)$; if so reject the current guess;
   b) whether $I$ is not a complete ground instance; if so, reject the current guess;
   c) for each $t \in I$, whether $I \setminus \{t\}$ is not a complete ground instance. Return “yes” if none of $I \setminus \{t\}$ is in RCQ$^v(Q, D_m, V)$.

Here Adom is defined in the same way as its counterparts used in previous proofs. One can readily verify that the algorithm is in $\Sigma_3^p$. Furthermore, by Lemma 4.7, it can be easily verified that the algorithm returns “yes” if and only if $T$ is a minimal $c$-instance in RCQ$^v(Q, D_m, V)$. \hfill $\square$

C. PROOFS OF SECTION 7

Proofs of Corollary 7.1

Corollary 7.1. For $c$-instances with a constant number of variables, and for fixed query $Q$ and a fixed set $V$ of CCs,

--- RCDP$^c$ and RCDP$^v$ are in PTIME for CQ, UCQ and $\exists\text{FO}^+$; and
--- RCDP$^w$ is in PTIME for CQ, UCQ, $\exists\text{FO}^+$ and FP.

Proof. We first show that for $c$-instances with a constant number of variables, and for fixed queries and fixed sets of CCs, RCDP$^c(L_Q)$ and RCDP$^v(L_Q)$ are both in PTIME when $L_Q$ is CQ, UCQ or $\exists\text{FO}^+$; we then prove that RCDP$^w(L_Q)$ is in also in PTIME when $L_Q$ is CQ, UCQ, $\exists\text{FO}^+$ or FP. By Lemma 3.2, we assume w.l.o.g. that $Q$ is defined over a relation schema $R$, and $T = (T, \xi)$ is a $c$-table of $R$.

RCDP$^c(L_Q)$. We give a PTIME algorithm for RCDP$^c(\exists\text{FO}^+)$ that given $(T, \xi)$, $D_m$, fixed $V$ and fixed $\exists\text{FO}^+$ query $Q$, returns “yes” if $(T, \xi)$ is in RCDP$^c(Q, D_m, V)$, i.e., for every valuation $\mu$ of $(T, \xi)$, $\mu(T)$ is a ground instance complete for $Q$; otherwise it returns “no”. The algorithm works as follows.

(1) Enumerate all valuations $\mu$ of $(T, \xi)$.

(2) For every such valuation $\mu$, do the following.
   a) Check whether $\mu(T)$ is in Mod$(T, D_m, V)$. If not, check the next valuation.
   b) Check whether $\mu(T)$ is a ground instance complete for $Q$. If not, return “no”.

(3) Return “yes” after all the valuations are inspected.

The algorithm checks whether $(T, \xi)$ is in RCDP$^c(Q, D_m, V)$ by the definition of RCDP$^c$, and is hence correct. The algorithm is in PTIME. Indeed, since $T$ has only a constant number of variables, there are only polynomially many valuations $\mu$ in step (1). Step (2) is also in PTIME: checking whether $\mu(T)$ is in Mod$(T, D_m, V)$ is in PTIME for fixed $V$, and checking whether $\mu(T)$ is not a ground instance complete for $Q$ is also in PTIME when $Q$ and $V$ are both fixed, by Theorem 4 of [Cao et al. 2014].
RCDP\textsuperscript{w}(\mathcal{L}_Q). We next give a PTIME algorithm for RCDP\textsuperscript{w}(\exists\text{FO}^+). Given \((T, \xi), D_m, \) fixed \(V\) and fixed \(\exists\text{FO}^+\) query \(Q,\) the algorithm returns “yes” if \((T, \xi)\) is in RCDP\textsuperscript{w}(\(Q, D_m, V\)), i.e., there exists a valuation \(\mu\) of \((T, \xi)\) such that \(\mu(T)\) is a ground instance complete for \(Q;\) otherwise it returns “no”. It works as follows.

1. Enumerate all valuations \(\mu\) of \(T\) that takes values from Adom, which is defined in the proof of Theorem 4.1.
2. For every such valuation \(\mu\), do the following.
   (a) Check whether \(\mu(T)\) is a ground instance complete for \(Q.\) If so, return “yes”.
   (b) Enumerate all valuations \(\mu'\) of \(T' = T \cup \{(x_1, \ldots, x_n)\},\) where \((x_1, \ldots, x_n)\) is a tuple consisting of (new) variables not in \(T.\) The algorithm returns “no” if \((T, \xi)\) is not in RCDP\textsuperscript{w}(\(Q, D_m, V\)), i.e., there exist a tuple \(t\) and an instance \(I \in \text{Mod}_{\text{Adom}}(T)\) such that \(t \in I\) and \(I \notin Q;\) otherwise it return “yes”. We use \(R_Q\) to denote the schema of query result \(Q(I),\) which has a constant arity when \(Q\) is fixed.
3. Return “no” after all the valuations are inspected.

Along the same lines as the argument for RCDP\textsuperscript{w} given above, we can show that the algorithm is correct and is in PTIME.

RCDP\textsuperscript{w}(\mathcal{L}_Q). We next show that the algorithm is in PTIME. In step (1) there are polynomially many tuples \(t\) since \(Q\) (hence \(R_Q\)) is fixed, and polynomially many valuations \(\mu\) since \(T\) has only a constant number of variables. Furthermore, step (2)(a) is in PTIME for fixed queries, and step (2)(b) is in PTIME since there are polynomially many valuations \(\mu'\) of \(T',\) for which step (i) is in PTIME for fixed \(V\) and step (ii) is also in PTIME for fixed \(Q.\) □

Proofs of Corollary 7.2

Corollary 7.2. For fixed queries

— RCDP\textsuperscript{w} and RCDP\textsuperscript{w} are in PTIME for CQ, UCQ and \(\exists\text{FO}^+\) when CCs are INDs; and
— RCDP\textsuperscript{w} is in \(O(1)\) time for CQ, UCQ, \(\exists\text{FO}^+\) and FP.

Proof. We first show that RCDP\textsuperscript{w}(\mathcal{L}_Q) and RCDP\textsuperscript{w}(\mathcal{L}_Q) are both in PTIME for fixed queries in CQ, UCQ and \(\exists\text{FO}^+\), when all CCs are INDs. Then we prove that RCDP\textsuperscript{w}(\mathcal{L}_Q) is in PTIME for fixed queries in CQ, UCQ, \(\exists\text{FO}^+\) and FP when CCs are not fixed. By Lemma 3.2, we assume that \(Q\) is defined over a relation schema \(R.\)
RCQP⁺(ŁQ). By Lemma 4.4, we only need to consider the problem for ground instances. We first give a PTIME algorithm for RCQP⁺(CQ), and extend it to RCQP⁺(UCQ) and RCQP⁺(∃FO⁺). To do these, we first review some notations introduced in [Fan and Geerts 2009].

— For a CQ Q in its tableau form (TQ, uQ), and for each variable y in TQ, the domain of y, denoted by dom(y), is the domain of attribute A when y appears in column A in TQ.

— A valuation μ for variables in CQ Q = (TQ, uQ) is said to be valid if for each variable y in TQ, μ(y) is a value from dom(y) and Q(μ(TQ)) is non-empty.

— A CQ query Q = (TQ, uQ) is bounded by (Dm, V) if for all variables y in uQ, either dom(y) is finite, or there exists an IND π(A,...)(R) ⊆ p in V such that y appears in column A in TQ, where π is the projection operator.

By Proposition 4.3 of [Fan and Geerts 2009], for each CQ query Q = (TQ, uQ), master data Dm and each set V of INDs, RCQP⁺(Q, Dm, V) is non-empty if and only if either Q is bounded by (Dm, V), or there exists no valid valuation μ of TQ taking values from Adom such that (μ(TQ), Dm) |= V. Accordingly, we give the PTIME algorithm as follows, which is a minor revision of the one given in [Fan and Geerts 2009].

1. Check whether Q is bounded by (Dm, V). If so return “yes”; otherwise continue. Obviously, this step can be done in PTIME when Q is fixed and V is a set of INDs.
2. Check whether there exists no valid valuation μ of TQ taking values from Adom such that (μ(TQ), Dm) |= V. If so, return “yes”; otherwise return “no”. This step can be done in PTIME since there exist polynomially many valuations μ given that Q is fixed, and moreover, for each such μ, it is in PTIME to check whether (μ(TQ), Dm) |= V when V consists of only INDs.

The algorithm is in PTIME, and thus RCQP⁺(CQ) is in PTIME.

As shown in [Fan and Geerts 2009], for a UCQ Q = Q₁ ∪⋯∪ Qₙ, RCQP⁺(Q, Dm, V) is non-empty if and only if each CQ query Qi is bounded, or there exists no valid valuation of Q. One can readily extend the PTIME algorithm given above to RCQP⁺(UCQ), and the revised algorithm is still in PTIME since Q is fixed. Moreover, a fixed ∃FO⁺ query Q can be transformed to an equivalent UCQ query of polynomial size in |Q|. Thus RCQP⁺(∃FO⁺) is also in PTIME.

RCQP⁺(ŁQ). By the proof of Corollary 6.2, Lemma 4.4 holds for viably complete c-instances. Thus RCQP⁺(ŁQ) for c-instances is equivalent to RCQP⁺(ŁQ) for ground instances. Hence the PTIME results for RCQP⁺(ŁQ) given above carry over to RCQP⁺(ŁQ).

RCQP⁺⁺(ŁQ). As shown in Theorem 5.4, RCQP⁺⁺(ŁQ) is decidable in O(1)-time for c-instances when ŁQ is CQ, UCQ, ∃FO⁺ or FP. The results obviously remain intact for fixed FP queries.

Proof of Corollary 7.3.

COROLLARY 7.3. For c-instances with a constant number of variables, and for fixed query Q and a fixed set V of CCs,

— MINP⁺ and MINP⁺⁺ are in PTIME for CQ, UCQ and ∃FO⁺; and
— MINP⁺⁺ is in PTIME for CQ.

PROOF. We first study MINP⁺(ŁQ) and MINP⁺⁺(ŁQ) when ŁQ is CQ, UCQ or ∃FO⁺, and then investigate MINP⁺⁺(ŁQ) when ŁQ is CQ. We assume that Q is defined over a relation schema R, and T = (T, ξ) is a c-table of R, by Lemma 3.2.
MINP\(^*\)(\(L_Q\)). We only need to show that MINP\(^*\)(\(\exists\)FO\(^+\)) is in PTIME. To do this, we give a PTIME algorithm as follows, which returns “yes” if for all valuations \(\mu\) of \(T\), \(\mu(T)\) is a minimal ground instance complete for \(Q\); otherwise it returns “no”.

1. Enumerate all valuations \(\mu\) of \(T\) with values drawn from \(\text{Adom}\), which is defined in the proof of Theorem 4.1. There are polynomially many such valuations since \(T\) has only a constant number of variables.
2. For each such valuation \(\mu\), do the following.
   
   (a) Check whether \(\mu(T) \in \text{Mod}(T, D_m, V)\). If so continue; otherwise check the next valuation. This can be done in PTIME for fixed \(V\).
   
   (b) Check whether \(\mu(T)\) is not a minimal instance complete for \(Q\). If so return “no”; otherwise check the next valuation. This can be done in PTIME for fixed \(Q\) and \(V\) by Theorem 5 of [Cao et al. 2014].
3. Return “yes” after all the valuations are inspected.

The algorithm is in PTIME and thus MINP\(^*\)(\(\exists\)FO\(^+\)) is in PTIME.

MINP\(^*\)(\(L_Q\)). We prove that MINP\(^*\)(\(\exists\)FO\(^+\)) is in PTIME by giving the following algorithm, which returns “yes” if and only if there exists a valuation of \(T\) such that \(\mu(T)\) is a minimal ground instance complete for \(Q\).

1. Enumerate all valuations \(\mu\) of \(T\) with values from \(\text{Adom}\). There are polynomially many such valuations since \(T\) has only a constant number of variables.
2. For each such valuation \(\mu\), do the following.
   
   (a) Check whether \(\mu(T) \in \text{Mod}(T, D_m, V)\). If so continue; otherwise check the next valuation. This can be done in PTIME for fixed \(V\).
   
   (b) Check whether \(\mu(T)\) is a minimal instance complete for \(Q\). If so return “yes”; otherwise check the next valuation. This can be done in PTIME for fixed \(Q\) and \(V\) by Theorem 5 of [Cao et al. 2014].
3. Return “no” after all the valuations are inspected.

The algorithm is in PTIME and thus MINP\(^*\)(\(\exists\)FO\(^+\)) is in PTIME.

MINP\(^*\)(\(L_Q\)). We show that the algorithm for MINP\(^*\)(CQ) given in the proof of Theorem 5.6 is in PTIME when \(Q\) and \(V\) are fixed and when \(T\) has a constant number of variables. Observe that it suffices to prove that it is in PTIME to check whether \(\emptyset\) is in RCQ\(^*\)(\(Q, D_m, V\)). As discussed in the proof of Theorem 5.6(4), \(\emptyset\) is not in RCQ\(^*\)(\(Q, D_m, V\)) if and only if for every pair of tuples \(t_1\) and \(t_2\) that possibly refer to the same tuple, \(Q(t_1) = Q(t_2)\) if \((\{t_1\}, D_m) \models V\) and \((\{t_2\}, D_m) \models V\). Accordingly, we give an algorithm as follows.

1. Enumerate all pairs of tuples \((t_1, t_2)\) with values from \(\text{Adom}\).
2. For every pair of such tuples \((t_1, t_2)\), do the following.
   
   (a) Check whether \((\{t_1\}, D_m) \models V\) and \((\{t_2\}, D_m) \models V\). If so continue; otherwise check the next pair of tuples.
   
   (b) Check whether \(Q(t_1) \neq Q(t_2)\). If so return “no”; otherwise check the next pair of tuples.
3. Return “yes” after all the pairs of tuples are inspected.

The algorithm is in PTIME. Indeed, polynomially many pairs of tuples \((t_1, t_2)\) in step (1) need to be checked for fixed \(Q\), since we only need to check those attributes that appear in \(Q\). Hence, step (2)(a) is in PTIME for fixed \(V\), and step (2)(b) is also in PTIME for fixed \(Q\). Thus it is in PTIME to check whether \(\emptyset\) is in RCQ\(^*\)(\(Q, D_m, V\)). As a result, MINP\(^*\)(\(L_Q\)) is in PTIME for fixed queries in CQ, by the proof of Theorem 5.6. \(\square\)