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

10

Databases in real life are often neither entirely closed-world nor entirely open-world. 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 article 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 $\mathcal{T}$ is *complete* for a query Q *relative to* master data. That is, depending on how missing values in $\mathcal{T}$ are instantiated, the answer to Q in $\mathcal{T}$ 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

ACM Reference Format:
Ting Deng, Wenfei Fan, and Floris Geerts. 2016. Capturing missing tuples and missing values. ACM Trans. Database Syst. 41, 2, Article 10 (May 2016), 47 pages.
DOI: http://dx.doi.org/10.1145/2901737

1. INTRODUCTION

Incomplete information has been a long-standing 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 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

T. Deng and W. Fan are supported in part by grants 973 Program 2014CB340302 and NSFC 61421003. W. Fan is also supported in part by grants NSFC 61133002, 973 Program 2012CB316200, ERC 652976, EPSRC EP/J015377/1, and EP/M025268/1, NSF III 1302212, Shenzhen Peacock Program 1105100030834361, Guangdong Innovative Research Team Program 2011D005, Shenzhen Science and Technology Fund JCYJ20150529164656096, and Guangdong Applied R&D Program 2015B010131006.
Authors' addresses: T. Deng, School of Computer Science and Engineering, Beihang University; W. Fan, School of Informatics, University of Edinburgh; F. Geerts, Department of Mathematics and Computer Science, University of Antwerp.

DOI: http://dx.doi.org/10.1145/2901737

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] and 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 2008]. It is a closed-world database about the enterprise in certain aspects, for example, 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, for example, employees and customers, other parts of the databases are open-world, for example, sales 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) 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 $\mathcal{I}$, that is, for every $\mathcal{I}'$ such that $\mathcal{I}' \supsetneq \mathcal{I}$ 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 in which only tuples may be missing from a database. Consider a database D of UK patients, specified by the schema

MVisit(NHS, name, city, yob, GD, Date, Diag, DrID),

of which each tuple records the National Health Service (NHS) number (NHS), name, address (city), year of birthday (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 $Q_1(D)$ since tuples may be missing from D even when no attribute values of the tuples in D are missing.

Not all is lost. Suppose that there is master data D_m available, specified by schema $\mathsf{Patient}_m$(NHS, name, yob, zip, 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[NHS]='915-15-335' and p[yob]=2000. 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] and Imieliński and Lipski Jr. [1984], we use a conditional table (c-table) T to represent such a database, as shown in Figure 1. In "tuple" t_2 of T, the values of t_2[name] and t_2[yob] are missing, and the condition t_2[cond] tells us that t_2[yob] is not 2001; similarly, the

	NHS	name	city	yob	GD	Date	Diag	DrID	cond
t_1:	915-15-335	John	EDI	2000	M	15/03/2015	Flu	01	
t_2:	915-15-356	x	EDI	z	F	15/03/2015	Diabetes	01	($z \neq 2001$)
t_3:	915-15-357	Mary	w	2000	F	15/03/2015	Influenza	u	($w \neq$Edi)
t_4:	915-15-358	Jack	LON	2000	M	15/03/2015	Influenza	02	
t_5:	915-15-359	Louis	LON	2000	M	15/03/2015	Diabetes	03	

Fig. 1. A c-table of a Patient.

condition t_3[cond] tells us that t_3[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 that 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 Jr. 1984] in which certain parts are bounded by master data via a set of CCs [Fan and Geerts 2009] (see Section 2.1 for their formal definition).

Models. We propose three models to specify whether a c-instance $\mathcal{T}$ is complete for a query Q relative to master data D_m: $\mathcal{T}$ is (1) *strongly complete* if each valuation of $\mathcal{T}$ yields a ground instance that is complete for Q relative to D_m; (2) *weakly complete* if one can find in $\mathcal{T}$ the certain answers to Q over all partially closed extensions of valuations of $\mathcal{T}$; and (3) *viably complete* if there exists a valuation of $\mathcal{T}$ that is a relatively complete database for Q. A user may choose a model that best serves one's 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 $\mathcal{T}$, whether $\mathcal{T}$ is (a) consistent, that is, whether there is any partially closed database represented by $\mathcal{T}$; and (b) extensible, that is, whether there exists any partially closed extension of $\mathcal{T}$.

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;

Table I. Complexity Results in Connection with Relative Completeness

$\mathcal{L}_Q$	RCDP	RCQP	MINP
Strong completeness	Theorem 4.1	Corollary 4.5	Theorem 4.8
FO, FP	undecidable	undecidable	undecidable
UCQ, $\exists$FO$^+$, CQ	Π_2^p-complete	NEXPTIME-c	Π_3^p-complete (D_2^p-c)
Weak completeness	Theorem 5.1	Theorem 5.4	Theorem 5.6
FO	undecidable	? (undecidable)	undecidable
FP	coNEXPTIME-c	$O(1)$	coNEXPTIME-c
UCQ, $\exists$FO$^+$	Π_3^p-complete	$O(1)$	Π_4^p-complete
CQ	Π_3^p-complete	$O(1)$	coDP-complete
Viable completeness	Theorem 6.1	Corollary 6.2	Corollary 6.3
FO, FP	undecidable	undecidable	undecidable
UCQ, $\exists$FO$^+$, CQ	Σ_3^p-complete (Π_2^p-c)	NEXPTIME-c	Σ_3^p-complete (D_2^p-c)

Note: Here, NEXPTIME-c, coNEXPTIME-c, D_2^p-c and Π_2^p-c are abbreviations for NEXPTIME-complete, coNEXPTIME-complete, D_2^p-complete and Π_2^p-complete, respectively.

—*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 with regard to several parameters:

—$\mathcal{L}_Q$: the query language in which Q is expressed, ranging over conjunctive queries, (CQ), union of conjunctive queries (UCQ), positive existential FO queries ($\exists$FO$^+$), first-order queries (FO), and FP, an extension of $\exists$FO$^+$with an inflational fix-point operator;
—c-instances versus ground instances, that is, in the presence or absence of missing values; and
—different models of relative completeness, that is 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 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, Π_2^p, Σ_2^p, D_2^p, Π_3^p, Σ_3^p, Π_4^p, NEXPTIME, coNEXPTIME, 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 $\exists$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_2^p-complete for ground instances while it is Π_3^p-complete for c-instances; in the viable completeness

model, RCDP for CQ is Π_2^p-complete for ground instances while it is Σ_3^p-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 cannot 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 Π_2^p-complete for the strongly complete c-instances, but it becomes Π_3^p-complete in the weak model.

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

To the best of our knowledge, apart from the conference version of this article [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] and Fan and Geerts [2010b] by dealing with missing tuples *and* missing values. We propose three models for relatively complete c-instances, which were not considered in these earlier works. 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 article, 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 article 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 Π_2^p-complete instead of Π_3^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 D_2^p-complete instead of Δ_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] and van der Meyden [1998] for surveys, and more recently, Olteanu et al. [2008]). This work adopts c-tables [Grahne 1991; Imieliński and Lipski Jr. 1984] to represent databases with missing values. Our weak model for relative completeness is based on the certain answer semantics [Imieliński and Lipski Jr. 1984], and the strong model has a resemblance to strong representation systems. In contrast, viably complete c-instances do not find a counterpart in Grahne [1991] and Imieliński and Lipski Jr. [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 for modeling databases with missing tuples (e.g., Gottlob and Zicari [1988], Levy [1996], Motro [1989], and 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] and 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. In addition, neither missing values nor the problems studied here were considered in Levy [1996] and 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 with regard to view definitions (e.g., see Abiteboul and Duschka [1998] and 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] and 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, for example, 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] and 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] and 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 that of 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 not 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, that is, 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. That 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. Section 8 summarizes the main results and identifies open problems.

2. RELATIVE INFORMATION COMPLETENESS REVISITED

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

2.1. Relatively Complete Ground Instances

A database schema $\mathcal{R}$ is a collection $(R_1, \ldots, R_n)$ of relation schemas. 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 $dom(A)$.

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 $t \in I_i$ and each $A \in R_i$, $t[A]$ is a constant in $dom(A)$.

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 CCs. A CC ϕ 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* ϕ, 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 $q_y(\mathsf{MVisit}) \subseteq p(\mathsf{Patient}_m)$, where $q_y(n, na, y, g) = \exists d, di, i\,(\mathsf{MVisit}(n, na, c, y, g, d, di, i) \wedge c = \text{'EDI'})$, and $p(n, na, y', g) = \exists z(\mathsf{Patient}_m(n, na, y', z, g))$, which ensures that D_m is an upper bound on the information in D about patients who live 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) $\phi : (\mathsf{NHS} \rightarrow \mathsf{name}, \mathsf{GD})$, 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 $D_\emptyset$. Then, the FD ϕ can be enforced by

including the following two CCs in V: $q_{\mathsf{name}} \subseteq D_\emptyset$ and $q_{\mathsf{GD}} \subseteq D_\emptyset$, where

$$q_{\mathsf{name}} = \exists n, na_1, na_2, c_1, c_2, y_1, y_2, g_1, g_2, d_1, d_2, di_1, di_1, i_1, i_2 \\ (\mathsf{MVisit}(n, na_1, c_1, y_1, g_1, d_1, di_1, i_1) \wedge \mathsf{MVisit}(n, na_2, c_2, y_2, g_2, d_2, di_2, i_2) \wedge n_1 \neq n_2),$$

which detects violations of the FD NHS $\rightarrow$ name; similarly, q_{GD} is defined to detect violations of the FD NHS $\rightarrow$ GD. Note that we allow inequalities in CQ, 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 $(\mathcal{I}, D_m)$ satisfies a set V of CCs, denoted by $(\mathcal{I}, D_m) \models V$, if $(\mathcal{I}, D_m) \models \phi$ for each CC ϕ in V.

A ground instance $\mathcal{I}$ of $\mathcal{R}$ is said to be *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.

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 $\mathcal{I} \subsetneq \mathcal{I}'$, if for all $i \in [1, n]$, $I_i \subseteq I'_i$, and furthermore, there is a $j \in [1, n]$ such that $I_j \subsetneq I'_j$. The set of *partially closed extensions* of $\mathcal{I}$ is defined as

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

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 $\mathsf{Ext}(\mathcal{I}, D_m, V)$ as $\mathsf{Ext}(\mathcal{I})$ when D_m and V are clear from the context.

A ground instance $\mathcal{I}$ is said to be *complete for a query Q relative to* (D_m, V) if (i) it is partially closed; and (ii) for each $\mathcal{I}' \in \mathsf{Ext}(\mathcal{I})$, $Q(\mathcal{I}) = Q(\mathcal{I}')$. 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 *relative to* (D_m, V): the extensions must satisfy V.

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, which 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 $t[\mathsf{NHS}] =$ '915-15-321'. V includes the CCs encoding FD ϕ, which ensures 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, which 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. 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).

2.2. Accommodating Missing Values

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

Partially closed *c*-instances. A *c-table* of R_i is a pair (T, ξ), 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 $dom(A)$ or a variable in $\mathsf{var}(A)$; and (b) ξ 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, true) the c-table without any conditions. An example of a c-table is shown in Figure 1.

A *valuation* μ *of* (T, ξ) 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 $dom(A)$ 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

$$\mu(T) = \{\mu(t) \mid t \in T \text{ and } \xi(\mu(t)) \text{ evaluates to } \mathsf{true}\}.$$

Hence, $\mu(T)$ is a ground instance *without* variables or conditions. More specifically, (T, ξ) represents a set of possible worlds $\mu(T)$ when μ ranges over all valuations of (T, ξ). We write (T, ξ) simply as T when ξ is clear from the context.

A *c-instance* $\mathcal{T}$ of $\mathcal{R}$ 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* μ *of* $\mathcal{T}$ is of the form $(\mu_1, \ldots, \mu_n)$, where μ_i is a valuation of T_i. We use $\mu(\mathcal{T})$ to denote the ground instance $(\mu_1(T_1), \ldots, \mu_n(T_n))$ of $\mathcal{R}$. A *partially closed c-instance* $\mathcal{T}$ represents a nonempty set of partially closed ground instances, denoted by $\mathsf{Mod}(\mathcal{T}, D_m, V)$. That is,

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

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

To simplify the discussion, in the sequel, we consider only c-instance $\mathcal{T}$ for which $\mathsf{Mod}(\mathcal{T})$ is nonempty. The assumption has no impact on the complexity results of this article. As will be shown by Proposition 3.3, it is in Σ_2^p to decide whether $\mathsf{Mod}(\mathcal{T})$ is nonempty. As we can see from Table I, all the complexity bounds of this article are higher than Σ_2^p-complete except RCDP for CQ, UCQ, and $\exists\mathrm{FO}^+$ in the strong completeness model, and $\mathsf{MINP}(\mathrm{CQ})$ 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 $\mathcal{T}$ is open-world in the absence of master data and CCs and closed-world if the master data is a possible world represented by $\mathcal{T}$.

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 $\mathcal{T}$ is

—*strongly complete for* Q if *for each* $\mathcal{I} \in \mathsf{Mod}(\mathcal{T})$ and *for each* $\mathcal{I}' \in \mathsf{Ext}(\mathcal{I})$, $Q(\mathcal{I}) = Q(\mathcal{I}')$;
—*weakly complete for* Q if

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

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

Intuitively, (a) $\mathcal{T}$ is strongly complete if, no matter how missing values in $\mathcal{T}$ are filled in, it yields a ground instance relatively complete for Q; (b) $\mathcal{T}$ is weakly complete if the certain answer to Q over all partially closed extensions of $\mathcal{T}$ can already be found in

$\mathcal{T}$; and (c) $\mathcal{T}$ is viably complete if there exists a way to instantiate missing values in $\mathcal{T}$ that results in a ground instance relatively complete for Q.

We use $\mathsf{RCQ}^s(Q, D_m, V)$, $\mathsf{RCQ}^w(Q, D_m, V)$, and $\mathsf{RCQ}^v(Q, D_m, V)$ to denote the set of all complete c-instances of $\mathcal{R}$ for Q with regard to (D_m, V), in the strong, weak, and viable completeness models, respectively. We simply use $\mathsf{RCQ}(Q, D_m, V)$ 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). By the FD ϕ encoded as CCs in V, we have that, for all valuations μ 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 $\mathsf{Ext}(\mu(T))$.

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, John, M, EH8 9AB, 2000) and t_m^2 = (915-15-336, Bob, M, EH8 9AB, 2000). Then, relative to (D_m, V), T is viably complete for Q_4, since there exists a valuation μ of T such that $\mu(T)$ is complete. For instance, this happens for $\mu(x)$ = Bob and $\mu(z) = 2000$. The c-instance T is also weakly complete, since the certain answer (name = 'John') can already be found over $\mathsf{Mod}(T)$. However, T is not strongly complete for Q_4. Consider $\mu'(T)$ with $\mu'(x)$ = John and $\mu'(z) = 2000$, and $\mu(T)$ defined as before. Then, clearly, $\mu'(T) \subseteq \mu(T)$; moreover, $Q_4(\mu'(T))$ only returns John, whereas $Q_4(\mu(T))$ returns both John and Bob.

We observe the following: (a) If $\mathcal{T}$ is strongly complete, then it is both weakly complete and viably complete. (b) A ground instance $\mathcal{I}$ is a c-instance without variables and conditions. It is strongly complete and viably complete for a query Q if and only if $\mathcal{I}$ is relatively complete for Q, as defined in Section 2.1. However, $\mathcal{I}$ 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 $\mathcal{I}$ is a *minimal* instance complete for a query Q relative to (D_m, V) if it is in $\mathsf{RCQ}(Q, D_m, V)$. Moreover, for all $\mathcal{I}' \subsetneq \mathcal{I}$, we have that $\mathcal{I}'$ is not in $\mathsf{RCQ}(Q, D_m, V)$. A c-instance $\mathcal{T}$ is a *minimal* c-instance viably complete (resp. strongly complete) for Q relative to (D_m, V) if there exists $\mathcal{I} \in \mathsf{Mod}(\mathcal{T})$ (resp. for all $\mathcal{I} \in \mathsf{Mod}(\mathcal{T})$) such that $\mathcal{I}$ is a *minimal* instance complete for a query Q.

To define minimal instances in the weak model, we write $(T, \xi) \subsetneq (T', \xi')$ if $T \subsetneq T'$ and ξ is the restriction of ξ' on T, that is, if for each valuation μ' of (T', ξ'), $\mu(T) \subsetneq \mu'(T')$, and if $\mu'(T')$ satisfies ξ', then $\mu(\mathcal{T})$ must satisfy ξ, where μ is the restriction of μ' on T. For $\mathcal{T} = (T_1, \ldots, T_n)$ and $\mathcal{T}' = (T_1', \ldots, T_n')$, we write $\mathcal{T} \subsetneq \mathcal{T}'$ if $T_i \subseteq T_i'$ for all $i \in [1, n]$, and $T_j \subsetneq T_j'$ for some $j \in [1, n]$.

A database $\mathcal{T}$ is a *minimal instance weakly complete* for Q relative to (D_m, V) if $\mathcal{T}$ is in $\mathsf{RCQ}(Q, D_m, V)$ and there exists no $\mathcal{T}' \subsetneq \mathcal{T}$ such that $\mathcal{T}'$ is in $\mathsf{RCQ}(Q, D_m, V)$. Note that $\mathcal{T}'$ 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[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 $\mathcal{L}_Q$.

RCDP($\mathcal{L}_Q$):	The *relatively complete database* problem.
INPUT:	A query Q in $\mathcal{L}_Q$, master data D_m, a set V of CCs, and a partially closed c-instance $\mathcal{T}$ with regard to (D_m, V).
QUESTION:	Is $\mathcal{T}$ in RCQ(Q, D_m, V)?

That is, does $\mathcal{T}$ have complete information to answer Q?

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

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 $\mathcal{T}$ as in RCDP.
QUESTION:	Is $\mathcal{T}$ a minimal c-instance complete for Q relative to (D_m, V)?

This asks whether $\mathcal{T}$ is a minimal-size database complete for Q, that is, removing any tuple from $\mathcal{T}$ makes it incomplete.

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

—CQ, the class of conjunctive queries built up from atomic formulas, that is, relation atoms in the schema $\mathcal{R}$, equality ($=$) and inequality ($\neq$), by closing under conjunction $\wedge$ 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;
—$\exists$FO$^+$, first-order logic (FO) queries built from atomic formulas, by closing under $\wedge$, *disjunction* $\vee$ and $\exists$;
—FO queries built from atomic formulas using $\wedge$, $\vee$, negation $\neg$, $\exists$, and universal quantification $\forall$; and
—FP, an extension of $\exists$FO$^+$ with an inflational fix-point operator, that is, 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 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 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 article 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.

Table II. Notations

Symbols	*Notations*
(T, ξ)	*c-table*, where T is a tableau and $\xi(t)$ is a condition for $\forall t \in T$
$\mu(T)$	valuation: $\{\mu(t) \mid t \in T$ and $\xi(\mu(t))$ evaluates to true$\}$
$\mathcal{T}$	*c-instance* $(T_1, \ldots, T_n)$ of schema $\mathcal{R}$
$(\mathcal{I}, D_m) \models \phi$	a ground instance $\mathcal{I}$ and master data D_m *satisfy* a CCϕ
$\mathsf{Ext}(\mathcal{I}, D_m, V)$	*partially closed extensions* of $\mathcal{I}$: $\{\mathcal{I}' \mid \mathcal{I} \subsetneq \mathcal{I}', (\mathcal{I}', D_m) \models V\}$
$\mathsf{Mod}(\mathcal{T}, D_m, V)$	$\{\mu(\mathcal{T}) \mid \mu$ is a valuation and $(\mu(\mathcal{T}), D_m) \models V\}$
$\mathsf{RCQ}^s(Q, D_m, V)$	the set of all strongly complete c-instances $\mathcal{T}$ of $\mathcal{R}$ for Q with regard to (D_m, V): $\{\mathcal{T} \mid \forall \mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \forall \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})\ (Q(\mathcal{I}) = Q(\mathcal{I}'))\}$
$\mathsf{RCQ}^w(Q, D_m, V)$	the set of all weakly complete c-instances $\mathcal{T}$ of $\mathcal{R}$ for Q with regard to (D_m, V): $\bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T})} Q(\mathcal{I}) = \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$
$\mathsf{RCQ}^v(Q, D_m, V)$	the set of all viably complete c-instances $\mathcal{T}$ of $\mathcal{R}$ for Q with regard to (D_m, V): $\exists \mathcal{I} \in \mathsf{Mod}(\mathcal{T})$ such that for $\forall \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})$, $Q(\mathcal{I}) = Q(\mathcal{I}')$.
$\mathsf{RCDP}^s(\mathcal{L}_Q)$ ($\mathsf{RCDP}^w(\mathcal{L}_Q)$ or $\mathsf{RCDP}^v(\mathcal{L}_Q)$)	the *relatively complete database* problem in strong (weak or viable) completeness model
$\mathsf{RCQP}^s(\mathcal{L}_Q)$ ($\mathsf{RCQP}^w(\mathcal{L}_Q)$ or $\mathsf{RCQP}^v(\mathcal{L}_Q)$)	the *relatively complete query* problem in strong (weak or viable) completeness model
$\mathsf{MINP}^s(\mathcal{L}_Q)$ ($\mathsf{MINP}^w(\mathcal{L}_Q)$ or $\mathsf{MINP}^v(\mathcal{L}_Q)$)	the *minimality* problem in strong (weak or viable) completeness model
$\mathcal{L}_Q$	CQ, UCQ, $\exists$FO$^+$, FOor FP

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] and 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 later that, when $\mathcal{C}$ consists of, for example, FDs and INDs, both $\mathsf{RCDP}(\mathcal{L}_Q)$ and $\mathsf{RCQP}(\mathcal{L}_Q)$ 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 $\mathcal{C}$, by a partially closed database $\mathcal{I}$, we mean a database that is partially closed in the usual sense, and $\mathcal{I}$ satisfies Θ. Similarly, partially closed extensions of $\mathcal{I}$ 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 $\mathcal{C}$, and a database schema $\mathcal{R}$.

—A ground instance $\mathcal{I}$ of $\mathcal{R}$ is said to be *partially closed* relative to (D_m, V, Θ) if $(D_m, \mathcal{I}) \models V$ and $\mathcal{I} \models \Theta$. That is, $\mathcal{I}$ is partially bounded by D_m via V and $\mathcal{I}$ is consistent with regard to the CCs in V and the additional constraints in Θ.
—A ground instance $\mathcal{I}'$ of $\mathcal{R}$ is said to be a *partially closed extension* of $\mathcal{I}$ relative to (D_m, V, Θ) if $\mathcal{I} \subsetneq \mathcal{I}'$, $(D_m, \mathcal{I}') \models V$, and $\mathcal{I}' \models \Theta$.
—A ground instance $\mathcal{I}$ of $\mathcal{R}$ 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 $\mathcal{I}'$ of $\mathcal{I}$, $Q(\mathcal{I}) = Q(\mathcal{I}')$. We use $\mathsf{RCQ}(Q, D_m, V, \Theta)$ 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 $\mathcal{L}_Q$ is CQ, 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 $\mathcal{L}_Q$, when $\mathcal{L}_Q$ is UCQ, $\exists$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 $\mathcal{R}$, and φ is an FD $X \to A$ defined on a relation schema R in $\mathcal{R}$. It is undecidable to determine, given such (Θ, φ), whether $\Theta \models \varphi$, that is, whether, for every instance $\mathcal{I}_\mathcal{R}$ of $\mathcal{R}$, if $\mathcal{I}_\mathcal{R} \models \Theta$, then $\mathcal{I}_\mathcal{R} \models \varphi$ (see 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:

$$Q(\,) = \exists \vec{x}, \vec{y}_1, \vec{y}_2, w, w' (R(\vec{x}, w, \vec{y}_1) \wedge R(\vec{x}, w', \vec{y}_2) \wedge w \neq w'),$$

where $\vec{x}$ corresponds to attributes X in R, w and w' both correspond to attribute A in R, as specified by the FD $\varphi : (X \to A)$ on R, and $\vec{y}_1$ and $\vec{y}_2$ encode attributes $R \setminus (X \cup \{A\})$. Intuitively, for an instance $\mathcal{I}_\mathcal{R}$ of $\mathcal{R}$, the query Q returns true if $I_R \not\models \varphi$, that is, when there exist tuples t_1, t_2 in I_R such that $t_1[X] = t_2[X]$ but $t_1[A] \neq t_2[A]$; otherwise, Q returns false. Moreover, we set D_m and V both to be empty.

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

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

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

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

Similarly, we rewrite the FD $\varphi : X \to A$ as $\varphi' : ([G, X] \to A)$. Intuitively, for each instance $\mathcal{I}_{\mathcal{R}'}$ of $\mathcal{R}'$, if we group tuples of $\mathcal{I}_{\mathcal{R}'}$ by the attribute G, then $\mathcal{I}_{\mathcal{R}'}$ is partitioned into a collection of groups $\mathcal{I}_g$, where g ranges over elements in $dom(G)$ that appear in the G-attribute of $\mathcal{I}_{\mathcal{R}'}$. One can readily verify that $\mathcal{I}_{\mathcal{R}'} \models \Theta'$ if and only if for each group $\mathcal{I}_g$, $\mathcal{I}_g \models \Theta$. Similarly, $\mathcal{I}_{\mathcal{R}'} \models \varphi'$ if and only if $\mathcal{I}_g \models \varphi$ for each group $\mathcal{I}_g$.

The CQ query Q is similar to its counterpart given these facts. It is defined as follows:

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

This query detects whether there exist tuples t_1 and t_2 violating the FD φ'. That is, it checks whether there exist 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 φ. If so, then Q returns the instance I_E of E.

We next show that $\Theta \models \varphi$ if and only if RCQ(Q, D_m, V, Θ') is nonempty. Assume first that $\Theta \models \varphi$. Then, one can readily verify that, for every instance $\mathcal{I}_{\mathcal{R}'}$ of $\mathcal{R}'$, if $\mathcal{I}_{\mathcal{R}'} \models \Theta'$,

then $Q(\mathcal{I}_{\mathcal{R}'})$ is empty. As a result, every instance $\mathcal{I}_{\mathcal{R}'}$ is in $\mathsf{RCQ}(Q, D_m, V, \Theta')$. Conversely, assume that $\Theta \not\models \varphi$. Then, there exists an instance $\mathcal{I}_{\mathcal{R}}$ such that $\mathcal{I}_{\mathcal{R}} \models \Theta$ but $\mathcal{I}_{\mathcal{R}} \not\models \varphi$. Assume, by contradiction, that there exists $\mathcal{I}_{\mathcal{R}'} \in \mathsf{RCQ}(Q, D_m, V, \Theta')$. We construct a partially closed extension $\mathcal{I}'_{\mathcal{R}'}$, as follows. Let g be a distinct value that does not appear in any G column of $\mathcal{I}_{\mathcal{R}'}$. Define $\mathcal{I}'_{\mathcal{R}'}$ such that, for each relation S in $\mathcal{R}$, its instance in $\mathcal{I}'_{\mathcal{R}'}$ is the union of $I' \cup (\{t_g\} \times I)$, where I', I are the instances of S in $\mathcal{I}_{\mathcal{R}'}$ and $\mathcal{I}_{\mathcal{R}}$, 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 $\mathcal{I}'_{\mathcal{R}'}$ properly contains its counterpart I_E in $\mathcal{I}_{\mathcal{R}'}$. Obviously $\mathcal{I}'_{\mathcal{R}'} \models \Theta'$, that is, $\mathcal{I}'_{\mathcal{R}'}$ is indeed a partially closed extension of $\mathcal{I}_{\mathcal{R}'}$. However, $Q(\mathcal{I}'_{\mathcal{R}'})$ is I'_E, which is by no means equal to the answer to Q in $\mathcal{I}_{\mathcal{R}'}$, since the latter is either $\emptyset$ or I_E. This contradicts the assumption that $\mathsf{RCQ}(Q, D_m, V, \Theta')$ is nonempty.

Note that, in these proofs, both master data and the set V of CCs are empty, that is, they are independent of the instance (Θ, φ) 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 relative completeness rather than by integrity constraints. As remarked earlier, CCs are powerful enough to express constraints often used in data cleaning.

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 $\mathcal{T}$, whether $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is nonempty.
—*The extensibility problem* is to determine, given master data D_m, a set V of CCs and a ground instance $\mathcal{I}$, whether $\mathsf{Ext}(\mathcal{I}, D_m, V)$ is nonempty, that is, whether $\mathcal{I}$ 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 following lemma. For a database schema $\mathcal{R}$, we denote by $\mathsf{inst}(\mathcal{R})$ the set of all ground instances of $\mathcal{R}$.

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

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

Here, $\mathcal{L}_Q$ ranges over CQ, UCQ, $\exists$FO$^+$, FO, and FP.

PROOF. We assume, without loss of generality, that all relations R_i in $\mathcal{R}$ 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 $dom(A) = [1, n]$. Define R to be an extension of R' by adding attribute $(A_R : dom(A))$. We define f_D, f_Q, and f_C as follows.

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

$$I_{(01)} = \begin{array}{|c|}\hline X \\ \hline 1 \\ 0 \\ \hline \end{array} \qquad I_{\vee} = \begin{array}{|ccc|}\hline A_1 & A_2 & B \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ \hline \end{array} \qquad I_{\wedge} = \begin{array}{|ccc|}\hline A_1 & A_2 & B \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array} \qquad I_{\neg} = \begin{array}{|cc|}\hline A & \bar{A} \\ \hline 0 & 1 \\ 1 & 0 \\ \hline \end{array}$$

Fig. 2. Ground relations used in the lower-bound proofs of Proposition 3.3.

(3) Similarly, we define f_C such that, for every CC ψ, $f_C(\psi)$ substitutes $R(A_R = i, \vec{x})$ for every occurrence of $R_i(\vec{x})$ in ψ.

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

PROPOSITION 3.3. *The consistency and extensibility problems are both Σ_2^p-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 Σ_2^p-complete.

(1) The consistency problem. We show that, given master data D_m, a set V of CCs and a c-instance $\mathcal{T}$, it is Σ_2^p-complete to decide whether $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is nonempty.

Lower bound. We show that the complement of the consistency problem, that is, the problem to decide whether $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is empty, is Π_2^p-hard. From this, it follows that the consistency problem is Σ_2^p-hard. We verify the Π_2^p-hardness by reduction from the $\forall^*\exists^*$3SAT problem, which is known to be Π_2^p-complete (see Papadimitriou [1994]). The $\forall^*\exists^*$3SAT problem is to determine, given a sentence $\varphi = \forall X \exists Y \psi(X, Y)$, whether φ is true. Here $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, and ψ is an instance of 3SAT, that is, $\psi = C_1 \wedge \cdots \wedge C_r$ and, for each $i \in [1, r]$, clause C_i is of the form $\ell_1^i \vee \ell_2^i \vee \ell_3^i$, where for each $l \in [1, 3]$, ℓ_l^i 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 $\mathcal{R}$, a c-instance $\mathcal{T}$ of $\mathcal{R}$, master data D_m, and a set V of CCs, such that φ is true if and only if $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is empty. We construct $\mathcal{R}$, $\mathcal{T}$, D_m and V as follows.

(a) The database schema $\mathcal{R}$ consists of five relation schemas: $R_{(0,1)}(X)$, $R_{\vee}(A_1, A_2, B)$, $R_{\wedge}(A_1, A_2, B)$, $R_{\neg}(A, \bar{A})$, and $R_X(X_1, \ldots, X_n)$. Intuitively, $R_{(0,1)}(X)$, $R_{\vee}(A_1, A_2, B)$, $R_{\wedge}(A_1, A_2, B)$, and $R_{\neg}(A, \bar{A})$ 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 $\mathcal{T} = (I_{(0,1)}, I_{\vee}, I_{\wedge}, I_{\neg}, T_X)$ in which $I_{(0,1)}$, $I_{\vee}$, $I_{\wedge}$, and $I_{\neg}$ are ground relations, as shown in Figure 2, to encode the Boolean domain, disjunction, conjunction, and negation, respectively, such that ψ can be expressed in CQ in terms of these relations, while $T_X = (\{(x_1, \ldots, x_n)\}, \mathsf{true})$ 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: $R^m_{(0,1)}(X)$, $R^m_{\vee}(A_1, A_2, B)$, $R^m_{\wedge}(A_1, A_2, B)$, $R^m_{\neg}(A, \bar{A})$, and $R^m_{\emptyset}(W)$. Intuitively, $R^m_{(0,1)}$, $R^m_{\vee}$, $R^m_{\wedge}$, and $R^m_{\neg}$ are the same as $R_{(0,1)}(X)$, $R_{\vee}(A_1, A_2, B)$, $R_{\wedge}(A_1, A_2, B)$, and $R_{\neg}(A, \bar{A})$ to store constant relations encoding Boolean values, disjunction, conjunction, and negation of variables, respectively

(denoted by $R^m_{(0,1)} = R_{(0,1)}$, $R^m_\vee = R_\vee$, $R^m_\wedge = R_\wedge$, $R^m_\neg = R_\neg$, respectively). The master data instances consist of $I^m_{(0,1)} = I_{(0,1)}$, $I^m_\vee = I_\vee$, $I^m_\wedge = I_\wedge$, $I^m_\neg = I_\neg$, and $I^m_\emptyset = \emptyset$.

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

—$R_{(0,1)} \subseteq R^m_{(0,1)}$, $R_\vee \subseteq R^m_\vee$, $R_\wedge \subseteq R^m_\wedge$, $R_\neg \subseteq R^m_\neg$; that is, the tables in $\mathcal{T}$ encoding the Boolean values and operations are fixed;
—for each $i \in [1, n]$, $\exists x_1 \ldots x_{i-1} x_{i+1} \ldots x_n R_X(x_1, \ldots, x_n) \subseteq R^m_{(0,1)}$; these ensure that each instance of R_X encodes a valid truth assignment for X;
—$q(w) \subseteq R^m_\emptyset(w)$, with $q(w) = \exists \vec{x}, \vec{y}\,(Q_X(\vec{x}) \wedge Q_Y(\vec{y}) \wedge Q_\psi(\vec{x}, \vec{y}, w) \wedge w = 1)$. Here, $Q_X(\vec{x}) = R_X(x_1, \ldots, x_n)$ picks a truth assignment for X, and $Q_Y(\vec{y})$ is $R_{(0,1)}(y_1) \wedge \cdots \wedge R_{(0,1)}(y_m)$ in CQ, that is, 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 (μ_X, μ_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_ψ can encode ψ in CQ by leveraging relations $I_\neg$, $I_\vee$, and $I_\wedge$. The query $q(w)$ returns $\{(1)\}$ if and only if $\psi(\mu_X, \mu_Y)$ evaluates true, where μ_X and μ_Y are the truth assignment selected by $Q_X(\vec{x})$ and $Q_Y(\vec{y})$, respectively.

Intuitively, for each ground instance $\mathcal{I} = (I_{(0,1)}, I_\vee, I_\wedge, I_\neg, I_X)$ of $\mathcal{T}$, $(\mathcal{I}, D_m) \models V$ if and only if, under the truth assignment μ_X of X variables encoded by I_X, there exists no truth assignment μ_Y of Y variables that makes ψ true.

We next show that φ is false if and only if $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is not empty.

$\boxed{\Rightarrow}$ First, assume that φ is false. Then, there exists a truth assignment μ^0_X of X such that there exists no truth assignment μ_Y that makes $\psi(\mu^0_X, \mu_Y)$ true. Let $\mathcal{I}_\mu = \mu(\mathcal{T}) = (I_{(0,1)}, I_\vee, I_\wedge, I_\neg, \mu(T_X))$ such that $\mu(T_X)$ agrees with μ^0_X. As discussed earlier, $\mathcal{I}_\mu$ is in $\mathsf{Mod}(\mathcal{T}, D_m, V)$. Hence, $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is nonempty.

$\boxed{\Leftarrow}$ Conversely, suppose that φ is true. Then, for all truth assignments μ_X of X, there exists a truth assignment μ_Y such that $\psi(\mu_X, \mu_Y)$ evaluates to true. Hence, for all ground instances $\mathcal{I}_\mu = (I_{(0,1)}, I_\vee, I_\wedge, I_\neg, \mu(T_X))$, where μ sets x_i to 0 or 1 (thus all possible truth assignments of X are considered), $q(\mathcal{I}_\mu)$ returns $\{(1)\}$. This violates the CC $q(w) \subseteq R^m_\emptyset(w)$. Hence, $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is empty.

Upper bound. We next provide a Σ^p_2 algorithm that, given master data D_m, a set V of CCs and a c-instance $\mathcal{T}$ as input, returns "yes" if $\mathsf{Mod}(\mathcal{T}, D_m, V)$ is nonempty.

By Lemma 3.2, we assume without loss of generality that $\mathcal{R}$ consists of a single relation schema and $\mathcal{T}$ is a c-table (T, ξ). To develop the algorithm, we need the following notations.

—We define Adom to be $S \cup \mathsf{New} \cup \mathbf{d}_f$, where (a) S consists of all constants that appear in T, D_m, or V; (b) New is a set of fresh values that are not in S, one for each variable in T or V; and (c) $\mathbf{d}_f$ is the set including all the values in the finite domains of attributes A in the relation schema that have a finite domain. Intuitively, Adom consists of all constants in the active domains of $\mathcal{T}$, D_m, or V and all constants appearing in the domains of attributes with finite domain. We will show that, for consistency checking, it suffices to consider valuations of a c-table drawing values from Adom only.
—A *valuation* μ *of a c-table* (T, ξ) *on* Adom is a valuation of (T, ξ) in which, for each variable x in T, $\mu(x)$ is in Adom and μ 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 Adom. We use $\mu(T)$ to denote the ground instance $\{\mu(t) \mid t \in T, \xi(\mu(t))$ evaluates to $\mathsf{true}\}$.

—We use $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ to denote the set $\{\mu(T) \mid \mu$ is a valuation on Adom, and $(\mu(T), D_m) \models V\}$. We also write $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ as $\mathsf{Mod}_{\mathsf{Adom}}(T)$ 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 μ of (T, ξ) such that $(\mu(T), D_m) \models V$.

(1) Guess a valuation μ of (T, ξ) on Adom.
(2) Check whether $(\mu(T), D_m) \models V$. If so, return "yes"; otherwise, reject the guess.

The algorithm is in Σ_2^p 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; hence, checking whether $(\mu(T), D_m) \not\models V$ can be done in NP as follows.

(1) Guess a constraint $q(\mathcal{R}) \subseteq p(\mathcal{R}_m)$ in V and let (T_q, u_q) be the tableau representing q; guess a valuation μ_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 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 NP since its second step is in PTIME. Hence, the consistency problem is in $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{coNP}}$.

The correctness of the algorithm for consistency checking is ensured by the following property: $\mathsf{Mod}(T, D_m, V)$ is nonempty if and only if $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ is nonempty. Hence, the algorithm returns "yes" only when $\mathsf{Mod}(T, D_m, V)$ is nonempty.

We next verify the property. If $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ is nonempty, then there exists a valuation μ of (T, ξ) on Adom such that $(\mu(T), D_m) \models V$. Then, obviously $\mu(T)$ is in $\mathsf{Mod}(T, D_m, V)$; hence, $\mathsf{Mod}(T, D_m, V)$ is nonempty. Conversely, suppose that there exists an instance I in $\mathsf{Mod}(T, D_m, V)$. Then, there exists a valuation ν of (T, ξ) such that $I = \nu(T)$. We next turn ν into a valuation μ of (T, ξ) on Adom, showing that $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ is nonempty. More precisely, we define μ such that, for every variable x in T, $\mu(x) = \nu(x)$ if $\nu(x) \in$ Adom; otherwise, $\mu(x)$ takes a value in New, such that it preserves the equality on variables, that is, 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 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 CC in V, and (T_q, u_q) be the tableau representing q. Then, for each valuation μ'_q of variables in T_q that draws values from $\mu(\mathcal{T})$, by the definition of $\mu(T)$, there exists a valuation μ_q of the variables such that μ_q agrees with μ'_q on variables that are not assigned a New-value, but take values from $\nu(T)$ outside of Adom, for the remaining variables. Recall that $\nu(\mathcal{T}) \in \mathsf{Mod}(T, D_m, V)$. 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, $(\mu(T), D_m) \models V$; hence, $\mu(T)$ is in $\mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$.

(2) The extensibility problem. We next show that, given D_m, V, and a ground instance $\mathcal{I}$, it is Σ_2^p-complete to decide whether $\mathsf{Ext}(\mathcal{I}, D_m, V)$ is nonempty.

Lower bound. We show that it is Π_2^p-hard to decide whether $\mathsf{Ext}(\mathcal{I}, D_m, V)$ is empty. This is again verified by reduction from the $\forall^*\exists^*3$SAT problem. Given an instance $\varphi = \forall X \exists Y \psi(X, Y)$ of $\forall^*\exists^*3$SAT, we use the same $\mathcal{R}$, D_m, and V given in (1), and let $\mathcal{I}_0 = (I_{(0,1)}, I_\vee, I_\wedge, I_\neg, I_X^\emptyset)$, where $I_X^\emptyset$ is empty.

We next show that φ is true if and only if $\mathsf{Ext}(\mathcal{I}_0, D_m, V)$ is empty.

$\boxed{\Rightarrow}$ First, assume that φ is true. Then, for all truth assignments μ_X of X, there exists a truth assignment μ_Y of Y such that $\psi(\mu_X, \mu_Y)$ evaluates to true. Hence, for all extensions $\mathcal{I}'_0 = (I'_{(0,1)}, I'_\vee, I'_\wedge, I'_\neg, I'_X)$ of $\mathcal{I}_0$, as long as $I'_{(0,1)} = I_{(0,1)}$, $I'_\vee = I_\vee$, $I'_\wedge = I_\wedge$,

$I'_\neg = I_\neg$, and I'_X encodes a truth assignment of X, $q(I'_X)$ returns $\{(1)\}$. This, however, violates the CC $q(w) \subseteq R^m_\emptyset(w)$; hence, $\mathsf{Ext}(\mathcal{I}_0, D_m, V)$ is empty.

$\boxed{\Leftarrow}$ Conversely, assume that φ is false. Let $\mathcal{I}'_0 = (I_{(0,1)}, I_\vee, I_\wedge, I_\neg, I^0_X)$, where I^0_X is an instance of R_X consisting of a single truth assignment μ^0_X of X such that $\exists Y \psi(\mu^0_X, Y)$ is false. Then, along the same lines as the previous proof, one can easily verify that $\mathcal{I}'_0$ is in $\mathsf{Ext}(\mathcal{I}_0, D_m, V)$.

Upper bound. We now develop a Σ^p_2 algorithm that takes master data D_m, a set V of CCs, and a ground instance I as input, and returns "yes" if $\mathsf{Ext}(I, D_m, V)$ is *not empty*.

To present the algorithm, we assume without loss of generality that I is an instance of a relation schema R, by Lemma 3.2. Recall the definition of Adom given in the upper bound proof for the consistency problem, except that $\mathcal{T}$ is replaced with the ground instance I. The algorithm works as follows.

(1) Guess a single tuple t of R with values from 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.

Following the same argument as the upper bound proof for the consistency problem, one can verify that the algorithm is in Σ^p_2. To show that the algorithm is correct, observe the following. (a) There exists an extension I' in $\mathsf{Ext}(I, D_m, V)$ if and only if there exists a single tuple t such that $I \cup \{t\}$ is in $\mathsf{Ext}(I, D_m, V)$. This can be easily verified based on the monotonicity of CQ queries that define the CCs in V. (b) There exists a tuple t such that $I \cup \{t\}$ is in $\mathsf{Ext}(I, D_m, V)$ if and only if there exists a tuple t' with values in Adom such that $I \cup \{t'\}$ is in $\mathsf{Ext}(I, D_m, V)$. This can be verified along the same lines as the upper bound proof for the consistency problem given what has been presented. Putting these together, we conclude that the algorithm returns "yes" if $\mathsf{Ext}(I, D_m, V)$ is nonempty. □

4. STRONG RELATIVE INFORMATION COMPLETENESS

We next study RCDP, RCQP, and MINP for strongly relatively complete databases, that is, 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 $\mathsf{RCQ}^s(Q, D_m, V)$ denotes the set of instances that are strongly complete for Q with regard to (D_m, V).

We establish complexity bounds on these problems for c-instances. For ground instances, we give complexity results for $\mathsf{MINP}^s(\mathcal{L}_Q)$ not considered in Fan and Geerts [2010b], and for the cases of $\mathsf{RCQP}^s(\mathcal{L}_Q)$ 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, $\mathsf{RCDP}^s(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ is FO or FP, and it is Π^p_2-complete when $\mathcal{L}_Q$ ranges over CQ, UCQ, and $\exists$FO$^+$ [Fan and Geerts 2009]. The following result tells us that the presence of missing values does not complicate 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. This is not the case, however: the complexity bounds remain intact when D_m and V are fixed.

THEOREM 4.1. *For c-instances,* $\mathsf{RCDP}^s(\mathcal{L}_Q)$ *is*

—*undecidable when* $\mathcal{L}_Q$ *is either FO or FP, and*
—Π_2^p*-complete when* $\mathcal{L}_Q$ *is CQ, UCQ, or* $\exists FO^+$.

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

PROOF. We first show that $\mathsf{RCDP}^s(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ is FO or FP. We then show that the problem becomes Π_2^p-complete when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists$FO$^+$.

(1) When $\mathcal{L}_Q$ 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 $\mathcal{L}_Q$ is CQ, UCQ, or $\exists$FO$^+$. We next show that $\mathsf{RCDP}^s(\mathcal{L}_Q)$ is Π_2^p-complete when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists$FO$^+$.

Lower bound. It is known that RCDP^s(CQ) for ground instances is Π_2^p-hard with fixed master data and CCs by reduction from the $\forall^*\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^s(CQ), RCDP^s(UCQ), and $\mathsf{RCDP}^s(\exists\text{FO}^+)$ are all in Π_2^p for c-instances. We first provide a Π_2^p algorithm for testing strongly complete c-instances for CQ queries. Later, we show how the algorithm can be extended to UCQ and $\exists$FO$^+$. We first present the algorithm for RCDP^s(CQ).

RCDP^s(CQ). To show that RCDP^s(CQ) is in Π_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 Π_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 $\mathcal{T} = (T, \xi)$. By Lemma 3.2, we assume, without loss of generality, that Q is defined over a relation schema R, and $\mathcal{T} = (T, \xi)$ 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 (e.g., see 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 $Q, \mathcal{T}, D_m$ or V, new distinct values in New that are not in $Q, \mathcal{T}, D_m$, and V, one for each variable in $Q, \mathcal{T}$, or V, and a set of constants $\mathbf{d}_f$ 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 $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_Q, D_m, V)$, $Q(I) = Q(I')$ [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, ξ)*,* (T, ξ) *is in* $\mathsf{RCQ}^s(Q, D_m, V)$ *if and only if, for each* $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$*,* 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 $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if it is bounded by (D_m, V). (2) The c-instance (T, ξ) is in $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if, for each $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$, I is in $\mathsf{RCQ}^s(Q, D_m, V)$. (1) We first show that I is in $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if , for each $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_Q)$, $Q(I) = Q(I')$. First, suppose that I is complete for Q. Then, for every $I' \in \mathsf{Ext}(I)$, $Q(I) = Q(I')$. In particular, for every $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_Q)$, $Q(I) = Q(I')$ since I' is also in $\mathsf{Ext}(I)$. Conversely, suppose that I is not complete for Q. Then, there must exist an $I' \in \mathsf{Ext}(I)$ such that $Q(I) \neq Q(I')$. More specifically, since Q is monotonic, there must exist a valuation ν of T_Q that draws values from I', such that $Q(\nu(T_Q)) \not\subseteq Q(I)$. Define a valuation ν' such that, for every variable x in T_Q, $\nu'(x)$ is a distinct value in New if $\nu(x)$ is not in Adom, and $\nu'(x) = \nu(x)$ otherwise. Observe the following. (a) The valuation ν' draws values from Adom. (b) $Q(\nu'(T_Q)) \not\subseteq Q(I)$, by the choice of the values in New and the assumption that $Q(\nu(T_Q)) \not\subseteq Q(I)$. (c) $(I \cup \nu'(T_Q), D_m) \models V$. This follows from the assumption that $I' \in \mathsf{Ext}(I)$, $I \cup \nu(T_Q) \subseteq I'$, 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 New, $(I \cup \nu'(T_Q), D_m) \models V$. Putting (a), (b), and (c) together, we have that $I \cup \nu'(T_Q)$ is in $\mathsf{Mod}_{\mathsf{Adom}}(I \cup T_Q)$ but $Q(I) \neq Q(I \cup \nu'(T_Q))$.

(2) We next show that (T, ξ) is in $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if, for each $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$, I is in $\mathsf{Mod}(T, D_m, V)$. First, assume that (T, ξ) is in $\mathsf{RCQ}^s(Q, D_m, V)$. Then, by the definition of strongly complete c-instances, all ground instances in $\mathsf{Mod}(T, D_m, V)$ are complete for Q relative to (D_m, V), including those in $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$. Conversely, assume that (T, ξ) is not in $\mathsf{RCQ}^s(Q, D_m, V)$. Then, there exists a valuation μ of T such that $I_1 = \mu(T)$, $I_1 \in \mathsf{Mod}(T, D_m, V)$, and there exists $I_2 \in \mathsf{Ext}(I_1)$ such that $Q(I_1) \subsetneq Q(I_2)$. 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 $I'_1 = \mu'(T)$, $I'_1 \in \mathsf{Mod}(T, D_m, V)$, $I'_2 = I'_1 \cup \nu'(T_Q)$, $I'_2 \in \mathsf{Ext}(I'_1)$, but $Q(I'_1) \subsetneq Q(I'_2)$. The valuations μ' and ν' are constructed by leveraging the choice of the values in New. That is, there exists an $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ 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 $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ and a tuple s such that $s \notin Q(I)$ but $s \in Q(I')$ for some $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_Q)$; otherwise, it returns "no." 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 $\mu(T) \in \mathsf{Mod}(T, D_m, V)$; if so, continue; otherwise, reject the current guess; this test can be done in coNP;
   (b) whether $(\mu(T) \cup \nu(T_Q)) \in \mathsf{Ext}(\mu(T), D_m, V)$; if so, continue; otherwise, reject the current guess; this test can be done in coNP;
   (c) whether $s \notin Q(\mu(T))$; if so, continue; otherwise, reject the current guess; this test can be done in coNP;
   (d) whether $s \in Q(\mu(T) \cup \nu(T_Q))$; if so, return "yes"; otherwise, reject the current guess; this test can be done in NP

The complexity of the algorithm is thus in Σ_2^p; hence, $\mathsf{RCDP}^s(\mathrm{CQ})$ 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. The counterexample consists of $I = \mu(T)$ and $I' = I \cup \nu(T_Q)$, 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." 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 $I = \mu(T)$, $I' = I \cup \nu(T_Q)$, $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$, and $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup \nu(T_Q), D_m, V)$, but there exists a tuple $s \in Q(I')$ and $s \notin Q(I)$. Such valuations μ and ν can 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: $Q_1 \cup \cdots \cup Q_k$, where Q_i is a query in CQ for each $i \in [1, k]$. 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 $i \in [1, k]$. 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 $i \in [1, k]$ and each $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_i)$, $Q(I) = Q(I')$.

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 $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if, for each $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$, 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), that is, for each $i \in [1, k]$ and each $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup T_i)$, $Q(I) = Q(I')$, where (T_i, u_i) is the tableau representation of Q_i; and (2) the c-instance (T, ξ) is in $\mathsf{RCQ}^s(Q, D_m, V)$ if and only if, for each $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$, 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. □

With this characterization, we extend the Σ_2^p algorithm given earlier to UCQ queries. More specifically, the algorithm presented earlier needs only a minor modification: In Step 2, we additionally guess one of the component queries Q_i in Q and a valuation ν_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 $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ and $I' \in \mathsf{Mod}_{\mathsf{Adom}}(I \cup \nu_i(T_i), D_m, V)$ for which $Q(I) \subsetneq Q(I')$. That is, the algorithm verifies whether there is an $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ that is not bounded by (D_m, V).

This modification does not affect the complexity of the algorithm; thus, it remains in Σ_2^p. 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 RCDP^s(CQ), based on Lemma 4.2. This shows that it is in Π_2^p time to determine whether a c-instance is strongly complete for a query in UCQ with regard to (D_m, V).

RCDP($\exists FO^+$). We show that the Π_2^p algorithm given earlier 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 Π_2^p. 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 Π_2^p algorithm for checking $\mathsf{RCDP}^s(\exists\text{FO}^+)$.

This completes the proof of Theorem 4.1. □

Theorem 4.1 is verified for c-instances $\mathcal{T}$ for which $\mathsf{Mod}(\mathcal{T})$ is nonempty. We next show that the complexity bounds remain unchanged without assuming that $\mathsf{Mod}(\mathcal{T})$ is nonempty. This obviously holds when $\mathcal{L}_Q$ is FO or FP, for which RCDP^s is undecidable. For CQ, UCQ, and $\exists$FO$^+$, we need an extra step to check whether $\mathsf{Mod}(\mathcal{T})$ is empty; if so, the algorithm terminates with "yes"; otherwise, it checks RCDP^s in Π_2^p. By

Proposition 3.3, the initial step is also in Π_2^p. Hence, RCDP^s remains in Π_2^p for CQ, UCQ, and $\exists\mathrm{FO}^+$. In other words, the assumption has no impact on the complexity of RCDP^s.

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 $\mathsf{RCQP}^s(\mathcal{L}_Q)$, we do not have to worry about missing values. $\mathsf{RCQP}^s(\mathcal{L}_Q)$ for c-instances and its counterpart for ground instances coincide.

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

PROOF. First, assume that there exists a c-instance $\mathcal{T} \in \mathsf{RCQ}^s(Q, D_m, V)$, with $|\mathcal{T}| \leq K$. Then, by the definition of strongly complete c-instances, $\mathsf{Mod}(\mathcal{T}, D_m, V) \neq \emptyset$; moreover, for each instance $\mathcal{I} \in \mathsf{Mod}(\mathcal{T}, D_m, V)$, $\mathcal{I}$ is complete for Q relative to (D_m, V). Furthermore, $|\mathcal{I}| \leq K$ since $\mathcal{I} = \mu(\mathcal{T})$ for a valuation of $\mathcal{T}$. Conversely, assume that there exists an instance $\mathcal{I}$ in $\mathsf{RCQ}^s(Q, D_m, V)$ with $|\mathcal{I}| \leq K$. Then, $\mathcal{I}$ is also a c-instance itself, which is complete for Q relative to (D_m, V). □

Consequently, one needs to consider only $\mathsf{RCQP}^s(\mathcal{L}_Q)$ for ground instances. Nevertheless, for ground instances, the complexity bounds on $\mathsf{RCQP}^s(\mathcal{L}_Q)$ were left open in Fan and Geerts [2009] when $\mathcal{L}_Q$ is FO or FP, and when CCs are expressed in CQ. $\mathsf{RCQP}(\mathcal{L}_Q)$ was shown undecidable in Fan and Geerts [2009] by using CCs expressed as fixed FO or FP queries. We settle these cases here.

THEOREM 4.5. *For c-instances, $\mathsf{RCQP}^s(\mathcal{L}_Q)$ is*

—undecidable when $\mathcal{L}_Q$ is FO or FP; and
—NEXPTIME-complete when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists$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 $\mathcal{R}$. □

(1) When $\mathcal{L}_Q$ is FO. We show that $\mathsf{RCQP}^s(\mathrm{FO})$ is undecidable by reduction from the satisfiability problem for FO, which is undecidable (see Trakhtenbrot [1950] and 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 $\mathsf{RCQ}^s(Q, D_m, V)$ is empty.

We now define D_m, V, and Q. The reduction does not rely on master data, that is, V and D_m are empty. To define Q, by Lemma 3.2, assume without loss of generality 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':

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

We show that q is satisfiable if and only if $\mathsf{RCQ}^s(Q, D_m, V)$ is empty.

$\boxed{\Rightarrow}$ First, suppose that q is satisfiable, and let I be an instance of R such that $q(I) \neq \emptyset$. For each instance I_1 of R', define $I_2 = I_1 \cup (\{A = b\} \times I) \cup \{t\}$, where b is a distinct constant not appearing in I_1, and t is a tuple not in I_1 and with $t[A] \neq b$. Here, t ensures that $I_1 \neq I_2$ even when I is empty. Then, I_2 is a partially closed extension of I_1 but $Q(I_1) \neq Q(I_2)$. $Q(I_2) = I_2$, but $Q(I_1) \neq I_2$ no matter whether $Q(I_1) = I_1$ or $Q(I_1) = \emptyset$. Thus, I_1 is not in $\mathsf{RCQ}^s(Q, D_m, V)$; hence, $\mathsf{RCQ}^s(Q, D_m, V)$ is empty.

$\boxed{\Leftarrow}$ Conversely, if q is not satisfiable, then $Q(I) = \emptyset$ for every instance I of R'. Thus, $\mathsf{RCQ}^s(Q, D_m, V)$ is not empty, since every instance of R' is relatively complete for Q.

(2) When $\mathcal{L}_Q$ is FP. To show that $\mathsf{RCQP}^s(\text{FP})$ is undecidable, we first prove the undecidability of the following problem, from which we will give a reduction to $\mathsf{RCQP}^s(\text{FP})$.

The *satisfiability problem for FP in the presence of FDs* is to determine, given an FP query p defined on schema $\mathcal{R}$ and a set Θ of FDs defined on $\mathcal{R}$, whether there exists an instance $\mathcal{I}$ of $\mathcal{R}$ such that $\mathcal{I} \models \Theta$ and $p(\mathcal{I})$ is nonempty. The undecidability of this problem was claimed in Levy et al. [1993]. We now 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 nonnullary relation schemas, one 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) is a quintuple $\mathcal{A} = (S, \Sigma, \Delta, s_0, s_{\text{acc}})$, consisting of a finite set of states S, an input alphabet $\Sigma = \{0, 1\}$, an initial state s_0, an accepting state s_{acc}, and a transition function $\Delta : S \times \Sigma_\epsilon \times \Sigma_\epsilon \to S \times \{0, +1\} \times \{0, +1\}$, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

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

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

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

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

(a) The database schema $\mathcal{R}$ 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 Σ^*. More specifically, an instance $\mathcal{I} = (I_P, I_S)$ of $\mathcal{R}$ 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 $\pi_{A_1,A_2}(\sigma_{A_1 \neq A_2}(I_S)) \cup \pi_{A_1,A_2}(\sigma_{A_1=A_2 \wedge W=1}(I_S))$, 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 Π_P and $\pi_{A_1,A_2}(\sigma_{A_1 \neq A_2}(S)) \cup \pi_{A_1,A_2}(\sigma_{A_1=A_2 \wedge W=1}(S))$ by FP query Π_S.

(b) We will use three FDs to ensure that we only consider those instances of P and S that represent a word in Σ^*, called *well-formed* instances of P and S. An instance $\mathcal{I} = (I_P, I_S)$ is well formed if (i) $\sigma_{V=1}(I_P)$ and $\sigma_{V=0}(I_P)$ are disjoint (i.e., a string can have only one letter at each position); and $\pi_{A_1,A_2}(\sigma_{A_1 \neq A_2}(I_S)) \cup \pi_{A_1,A_2}(\sigma_{A_1=A_2 \wedge W=1}(I_S))$ 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 ensure this, we require the presence of a tuple $(1, k, k)$ in I_S. We also 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 ensured by FP-queries Π_{ini} and Π_{fin}, respectively, to be defined shortly.

More specifically, the conditions (i) through (iii) will be enforced by the following set Θ of FDs:

—$A \to V$, enforcing that, for every instance $\mathcal{I}' = (I'_P, I'_S)$ of $\mathcal{R}$ such that $\mathcal{I}' \models \Theta$, condition (i) is satisfied for I'_P.
—$A_1 \to A_2$, ensuring that $\pi_{A_1,A_2}(I'_S)$ encodes a function; hence, condition (ii) is satisfied.
—$W \to A_1, A_2$, ensuring that there can be at most one tuple with its W-attribute set to 1 in I'_S. As a result, $\pi_{A_1,A_2}(\sigma_{A_1=A_2 \wedge W=1}(I'_S))$ contains at most one tuple, and condition (iii) is satisfied.

In summary, any instance $\mathcal{I}' = (I'_P, I'_S)$ of $\mathcal{R}$ that satisfies Θ 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 $\mathcal{I}'$.

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

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

Intuitively, $\alpha_1(y)$ is to represent the position of y based on the value of in_1; similarly for $\alpha_2(z)$ and in_2; and $\beta_1(y, y')$ is to decide whether y and y' are consecutive positions or not. More specifically,

—$\alpha_1(y) = \exists y'(\Pi_S(y, y') \wedge y \neq y' \wedge \Pi_P(1, y))$ if $\mathsf{in}_1 = 1$;
—$\alpha_1(y) = \exists y'(\Pi_S(y, y') \wedge y \neq y' \wedge \Pi_P(0, y))$ if $\mathsf{in}_1 = 0$; and
—$\alpha_1(y) = \Pi_S(y, y)$ if $\mathsf{in}_i = \epsilon$;

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

—$\beta_1(y, y') = \Pi_S(y, y')$ if $\mathsf{move}_i = +1$ and
—$\beta_1(y, y') = (y = y')$ if $\mathsf{move}_i = 0$;

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 $\mathcal{A}$ makes a move (with head 1) and $y = y'$ otherwise; similarly for $\beta_1(z, z')$. Then, following Spielmann [2000], one can show that $\Phi = \exists y_1 y_2[\mathsf{TC}_{x,y,z;x',y',z'} \bigvee_{\delta \in \Delta} \varphi_\delta](s_0, 0, 0, s_{acc}, y_1, y_2)$ is satisfiable if and only if $L(\mathcal{A}) \neq \emptyset$.

Clearly, we can compute Φ using a query Π_Φ in FP. Recall that we still need to ensure the existence of an initial and a final position in well-formed instances of I_S. The final FP-query Π is therefore defined as $\Pi_\Phi \wedge \Pi_{ini} \wedge \Pi_{fin}$, where $\Pi_{ini} = \exists w\, x\, \Pi_S(w, 0, x)$ and $\Pi_{fin} = \exists x\, \Pi_S(1, x, x)$.

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

In light of Lemma 4.6, we show that RCQP^s(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 Θ 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 Θ if and only if $\mathsf{RCQ}^s(Q, D_m, V)$ is empty.

Suppose that p and Θ are defined over a database schema $\mathcal{R}$. By Lemma 3.2, we assume without loss of generality 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 $X \to A$ in Θ, $(G, X) \to A$ is an FD defined over R'. Denote by Θ' the set of all such FDs over R' deduced from FDs in Θ. For each FD $(G, X) \to A$ in Θ', we express it as a CC: $p_v(R') \subseteq \emptyset$, where

$$p_v(g) = \exists \vec{x}, y, y', \vec{z}_1, \vec{z}_2\, (R'(g, \vec{x}, y, \vec{z}_1) \wedge R'(g, \vec{x}, y', \vec{z}_2) \wedge y \neq y'),$$

$\vec{x}, y$ and $\vec{z}_1$ correspond to attributes X, A, and $R' \setminus (X \cup \{A\})$, respectively; similarly for $\vec{x}, y'$ and $\vec{z}_2$. That is, $p_v(R')$ extracts tuples that violate the FD $(G, X) \to A$. We define V to be the set of all CCs constructed from FDs in Θ' as delineated earlier. Intuitively, we group tuples of R' by the attribute G, such that FDs in Θ are imposed on each group individually. By Lemma 4.6, Θ is fixed; 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 nonempty,
—there exists an instance I' of R' such that there exists $g \in dom(G)$, $I'_g \models \Theta$ and $p(I'_g)$ is nonempty, where I'_g is the subset of I' consisting of tuples t with $t[G] = g$.

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

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

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

$\boxed{\Leftarrow}$ Conversely, assume that p is satisfiable. Then, there exists an instance I of R such that $I \models \Theta$ and $p(I)$ is nonempty. We next show that $\mathsf{RCQ}^s(Q, D_m, V)$ is empty. That is, we need to prove that, for each instance (I', E) of $\mathcal{R}'$, $(I', E) \notin \mathsf{RCQ}^s(Q, D_m, V)$ when $((I', E), D_m) \models V$. We construct an extension I'' of I' such that, for each tuple t in I, (g, t) is in I'', for a constant $g \in dom(G)$ 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 nonempty. 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 $\mathsf{RCQ}^s(Q, D_m, V)$.

(3) When $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$. It is known [Fan and Geerts 2010b] that $\mathsf{RCQP}^s(\exists\mathrm{FO}^+)$ is in NEXPTIME and that $\mathsf{RCQP}^s(\mathrm{CQ})$ is NEXPTIME-hard when D_m and V are fixed.

From this and Lemma 4.4, it follows that $\mathsf{RCQP}(\mathcal{L}_Q)$ is NEXPTIME-complete when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$.

Note that, in these proofs, 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 does not contain excessive data. The following lemma 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 $I' \subsetneq I$, $(I', D_m) \models V$; and (b) if I is in $\mathsf{RCQ}^s(Q, D_m, V)$, then I is not minimal if and only if there exists a tuple $t \in I$ such that $I\backslash\{t\} \in \mathsf{RCQ}^s(Q, D_m, V)$.*

PROOF. Consider query Q, master data D_m, and a set V of CCs. By Lemma 3.2, we assume without loss of generality 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 $I' \subseteq I$, $(I', D_m) \models V$. We show that, for every $\phi \in V$, $(I', D_m) \models \phi$. Let ϕ be $q(R) \subseteq p(D_m)$, where q is a CQ query. Since $I' \subseteq I$, $q(I') \subseteq q(I)$ because CQ queries are monotonic. By $(I, D_m) \models V$, $q(I) \subseteq p(D_m)$; hence, $q(I') \subseteq p(D_m)$, that is, $(I', D_m) \models \phi$.

(b) Suppose that I is in $\mathsf{RCQ}^s(Q, D_m, V)$. Then, I is not minimal if and only if there exists a tuple $t \in I$ such that $I\backslash\{t\} \in \mathsf{RCQ}^s(Q, D_m, V)$. To see this, first assume that $I\backslash\{t\} \in \mathsf{RCQ}^s(Q, D_m, V)$. Then, obviously, I is not minimal by the definition of minimal instances. Conversely, suppose that I is not minimal, that is, there exists $I_1 \subsetneq I$ such that for each $I_2 \in \mathsf{Ext}(I_1, D_m, V)$, $Q(I_1) = Q(I_2)$. Note that $I \in \mathsf{Ext}(I_1, D_m, V)$. Then, there must exist $I_2 = I\backslash\{t\}$ for some $t \in I$ such that $I_1 \subseteq I_2$ and $Q(I_1) = Q(I_2)$. By (a), $(I_2, D_m) \models V$. In addition, for all $I' \in \mathsf{Ext}(I_2, D_m, V)$, I' is also in $\mathsf{Ext}(I_1, D_m, V)$; hence, $Q(I') = Q(I_1) = Q(I_2)$. Thus, I_2 is in $\mathsf{RCQ}^s(Q, D_m, V)$. □

Capitalizing on this lemma, next, we provide complexity bounds on $\mathsf{MINP}^s(\mathcal{L}_Q)$. Here, the presence of missing values again makes the problem a little harder: $\mathsf{MINP}^s(\mathrm{CQ})$ is D_2^p-complete for ground instances, but it is Π_3^p-complete for c-instances. Here, D_2^p is the class of languages recognized by oracle machines that make a call to a Σ_2^p oracle and a call to a Π_2^p oracle. That is, L is in D_2^p if there exist languages $L_1 \in \Sigma_2^p$ and $L_2 \in \Pi_2^p$ such that $L = L_1 \cap L_2$ [Wooldridge and Dunne 2004].

THEOREM 4.8. *When $\mathcal{L}_Q$ is FO or FP, $\mathsf{MINP}^s(\mathcal{L}_Q)$ is undecidable both for ground instances and for c-instances. When $\mathcal{L}_Q$ is CQ, UCQ, or $\exists$FO$^+$, $\mathsf{MINP}^s(\mathcal{L}_Q)$ is*

—*Π_3^p-complete for c-instances, and*
—*D_2^p-complete for ground instances.*

The complexity is unchanged when D_m and V are fixed.

PROOF. We first show that $\mathsf{MINP}^s(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ is either FO or FP. We then verify that, when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$, $\mathsf{MINP}^s(\mathcal{L}_Q)$ is Π_3^p-complete for c-instances but is D_2^p-complete for ground instances. In the proofs for undecidability and lower bounds to be given later, we use fixed D_m and V.

(1) When $\mathcal{L}_Q$ is FO or FP. To show that $\mathsf{MINP}^s(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ 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 D_m, and a set of V of CCs, whether a special instance $\mathcal{I}_\emptyset$ is in $\mathsf{RCQ}^s(Q, D_m, V)$, where $\mathcal{I}_\emptyset$ is the empty instance of the schema over which Q is defined. If $\mathcal{I}_\emptyset$ is in $\mathsf{RCQ}^s(Q, D_m, V)$, then it is a *minimal* instance complete for Q relative to (D_m, V).

$\mathsf{MINP}^s(FO)$. We verify the undecidability of $\mathsf{MINP}^s(\mathrm{FO})$ by reduction from the satisfiability of FO queries. Given an FO query q, we assume without loss of generality by Lemma 3.2 that q is defined over a single relation R. Consider the FO query Q defined in the proof of Theorem 4.5 (1), and let D_m and V both be empty. It has been shown there that if q is not satisfiable, then $I_\emptyset$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. Conversely, if q is satisfiable, then $\mathsf{RCQ}^s(Q, D_m, V)$ is empty; hence, $I_\emptyset$ is not in $\mathsf{RCQ}^s(Q, D_m, V)$. Thus, q is satisfiable if and only if $I_\emptyset$ is minimal in $\mathsf{RCQ}^s(Q, D_m, V)$.

$\mathsf{MINP}^s(FP)$. Along the same lines as the proof for $\mathsf{MINP}^s(\mathrm{FO})$, it suffices to show that, given Q in FP, D_m, and V, it is undecidable to determine whether the special instance $I_\emptyset$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. This has already been verified by the proof of Theorem 4.5 (2). It has been shown that deciding nonemptiness of $\mathsf{RCQ}^s(Q, D_m, V)$ is undecidable. In particular, $\mathsf{RCQ}^s(Q, D_m, V) \neq \emptyset$ if and only if $I_\emptyset \in \mathsf{RCQ}^s(Q, D_m, V)$.

(2) When $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$. We prove that $\mathsf{MINP}^s(\mathcal{L}_Q)$ is Π_3^p-complete for c-instances and D_2^p-complete for ground instances.

(*2.1*) *For c-instances*. To show that $\mathsf{MINP}^s(\mathcal{L}_Q)$ is Π_3^p-complete for c-instances when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$, it suffices to verify that $\mathsf{MINP}^s(\mathrm{CQ})$ is Π_3^p-hard and that $\mathsf{MINP}^s(\exists\mathrm{FO}^+)$ is in Π_3^p.

Lower bound. We show that $\mathsf{MINP}^s(\mathrm{CQ})$ is Π_3^p-hard by reduction from the complement of the $\exists^*\forall^*\exists^*3\textsc{sat}$ problem, which is known to be Σ_3^p-complete (see Papadimitriou [1994]). The $\exists^*\forall^*\exists^*3\textsc{sat}$ problem is to determine, given a sentence $\varphi = \exists X \forall Y \exists Z \psi$, whether or not φ is true. Here, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $Z = \{z_1, \ldots, z_k\}$, and ψ is an instance of 3SAT.

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

(a) The database schema $\mathcal{R}$ consists of six relation schemas: $R_{(0,1)}(A)$, $R_\neg(A, \bar{A})$, $R_\vee(A_1, A_2, B)$, $R_\wedge(A_1, A_2, B)$, $R_X(\mathsf{id}, X)$, and $R_s(W)$, where $R_{(0,1)}$, $R_\neg$, $R_\vee$, and $R_\wedge$ are the same as their counterparts in the proof of Proposition 3.3. The relation $R_X(\mathsf{id}, X)$ 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 $\mathcal{T} = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, T_X, I_s)$, in which $I_{(0,1)}$, $I_\neg$, $I_\vee$, $I_\wedge$ are ground relations, as shown in Figure 2, while $T_X = (\{(1, x_1), (2, x_2), \ldots, (n, x_n)\}, \mathsf{true})$ 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 $\mathcal{T}$ to encode basic Boolean operations, truth assignments for variables of X, and possible truth values of ψ in the reduction.

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

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

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

The last two CCs 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 φ is true if and only if (i) there exists a ground instance $\mathcal{I}$ of $\mathcal{T}$ that encodes a truth assignment μ_X of X variables by an instance of T_X in $\mathcal{I}$, and (ii) $Q(\mathcal{I})$ returns tuples representing all truth assignments μ_Y of Y variables when ψ is true under (μ_X, μ_Y, μ_Z); here, μ_Z is a truth assignment of Z variables, which is encoded by a tuple returned by a subquery of Q on $\mathcal{I}$ (if it exists). More specifically, query Q is defined as follows.

$$Q(\vec{y}) = \exists \vec{x}, \vec{z}\,(Q_X(\vec{x}) \wedge Q_Y(\vec{y}) \wedge Q_Z(\vec{z}) \wedge Q_\psi(\vec{x}, \vec{y}, \vec{z}, w) \wedge R_s(w) \wedge Q_{\mathsf{all}}),$$

where Q_X is a CQ query $Q_X(x_1, \ldots, x_n) = \bigwedge_{i\in[1,n]} R_X(i, x_i)$, that is, it selects from R_X the truth assignments for X. The query $Q_Y(\vec{y}) = R_{(0,1)}(y_1) \wedge \cdots \wedge R_{(0,1)}(y_m)$ constructs all possible truth assignments of variables in Y; similarly for $Q_Z(\vec{z})$ and Z. Given a truth assignment (μ_X, μ_Y, μ_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_ψ can be defined in CQ by leveraging relations $I_\neg$, $I_\vee$, and $I_\wedge$. The query Q_{all} is to ensure that all the tuples in $I_{(0,1)}$, $I_\neg$, $I_\vee$, $I_\wedge$, and tuple (1) in I_s are in place. More specifically, it is defined as

$$Q_{\mathsf{all}} = Q_{(0,1)} \wedge Q_\neg \wedge Q_\vee \wedge Q_\wedge \wedge Q_s,$$

where $Q_\vee = R_\vee(0,0,0) \wedge R_\vee(0,1,1) \wedge R_\vee(1,0,1) \wedge R_\vee(1,1,1)$ asserts that the removal of any of the four tuples in $I_\vee$ makes $Q(\vec{y})$ empty; similarly for $Q_{(0,1)}$, $Q_\neg$, and $Q_\wedge$. 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 μ_Y of Y such that, for the truth assignment μ_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 φ is false if and only if $\mathcal{T}$ is a minimal c-instance in $\mathsf{RCQ}^s(Q, D_m, V)$, that is, for each ground instance $\mathcal{I}$ of $\mathcal{T}$, $\mathcal{I}$ is minimal in $\mathsf{RCQ}^s(Q, D_m, V)$. The argument is based on the following observations: for every ground instance $\mathcal{I} = (I_{(0,1)}, I_\wedge, I_\vee, I_\neg, I_X, I_s)$ of $\mathcal{T}$, (i) $\mathcal{I}$ has no extensions by the definition of V, that is, V enforces an upper limit on the potential valuations of $\mathcal{T}$; (ii) removing any tuple from I_X, $I_{(0,1)}$, $I_\neg$, $I_\vee$, or $I_\wedge$, or removing tuple (1) from I_s would make Q empty, by the definitions of Q and Q_{all}; that is, Q_{all} imposes a lower limit on which tuples must exist.

$\boxed{\Rightarrow}$ First, assume that φ is false. Then, for each truth assignment μ_X of X, there exists a truth assignment μ_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 $\mathcal{I} = (I_{(0,1)}, I_\wedge, I_\vee, I_\neg, I_X, I_s) \in \mathsf{Mod}(\mathcal{T})$ of $\mathcal{T}$, $Q(\mathcal{I})$ returns all truth assignments of Y since I_s consists of (0) and (1). As argued in (i) earlier, there exists no extension to $\mathcal{I}$; thus, $\mathcal{I} \in \mathsf{RCQ}^s(Q, D_m, V)$. We next show that $\mathcal{I}$ is minimal. As argued in (ii) earlier, we only need to consider the instance $\mathcal{I}' = (I_{(0,1)}, I_\wedge, I_\vee, I_\neg, I_X, I'_s)$, where $I'_s = \{(1)\}$. Obviously, $Q(\mathcal{I}')$ does not contain all truth assignments of Y since φ is false, making it different from $Q(\mathcal{I})$. Hence, $\mathcal{T}$ is a minimal c-instance.

$\boxed{\Leftarrow}$ Conversely, suppose that φ is true. Then, there exists a truth assignment μ^0_X of X such that, for every truth assignment μ_Y of Y, $\exists Z\psi(\mu^0_X, \mu_Y, Z)$ is true. We show that there exists a ground instance in $\mathsf{Mod}(T, D_m, V)$ that is not minimal. Let I^0_X be a ground instance of T_X that agrees with μ^0_X, and $\mathcal{I}^0 = (I_{(0,1)}, I_\wedge, I_\vee, I_\neg, I^0_X, I_s)$. As before, $Q(\mathcal{I}^0)$ consists of all truth assignments of Y. However, $\mathcal{I}^0$ is not minimal. Consider $\mathcal{I}' = (I_{(0,1)}, I_\wedge, I_\vee, I_\neg, I^0_X, I'_s)$ with $I'_s = \{(1)\}$. Then, since φ is true and I^0_X encodes μ^0_X, $Q(\mathcal{I}')$ consists again of all truth assignments of Y. Hence, $\mathcal{I}^0$ is not minimal; thus, $\mathcal{T}$ is not a minimal c-instance in $\mathsf{RCQ}^s(Q, D_m, V)$.

Upper bound. We next show that $\mathsf{MINP}^s(\exists\mathrm{FO}^+)$ is in Π_3^p for c-instances. The proof makes use of Lemma 4.7 and the Σ_2^p algorithm [Fan and Geerts 2009] for checking whether a ground instance is not in $\mathsf{RCQ}^s(Q, D_m, V)$.

We give an Σ_3^p algorithm for the complement of $\mathsf{MINP}^s(\exists\mathrm{FO}^+)$: given a c-instance $\mathcal{T}$, a query Q in $\exists\mathrm{FO}^+$, master data D_m, and a set V of CCs, the algorithm returns "yes" if and only if $\mathcal{T}$ is *not* a minimal c-instance in $\mathsf{RCQ}^s(Q, D_m, V)$. By Lemma 3.2, we assume without loss of generality that $\mathcal{T}$ consists of a single c-table T. Assume that T 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.

(1) Guess a valuation μ of (T, ξ) with Adom and guess an index $i \in [1, k]$. Let $I = \mu(T)$.
(2) Test the following:
   (a) whether $I \notin \mathsf{Mod}(T, D_m, V)$; if not, continue; otherwise, reject the current guess;
   (b) whether I is not a complete ground instance; if so, return "yes";
   (c) whether $\mu(\tau_i) \in I$ and $I \backslash \mu(\tau_i)$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. If so, return "yes"; otherwise, reject the current guess.

The algorithm is in Σ_3^p. It invokes an NP oracle to check whether an instance is in $\mathsf{Mod}(T, D_m, V)$ in Step (2)(a), invokes an Σ_2^p oracle to check whether $I \notin \mathsf{RCQ}^s(Q, D_m, V)$ in Step (2)(b), and a Π_2^p oracle to check whether $I \backslash \mu(\tau_i) \in \mathsf{RCQ}^s(Q, D_m, B)$ (see the proof of Theorem 4.1). Hence, the algorithm is in $\mathsf{NP}^{\Sigma_2^p}$, that is, in Σ_3^p.

The algorithm returns "yes" when a counterexample is found that dispels T as a minimal c-instance in $\mathsf{RCQ}^s(Q, D_m, V)$. Conversely, suppose that T is not a minimal c-instance in $\mathsf{RCQ}^s(Q, D_m, V)$. 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 $\mathsf{RCDP}^s(\exists\mathrm{FO}^+)$, one can show that there must be a ground instance $I \in \mathsf{Mod}_{\mathsf{Adom}}(T, D_m, V)$ such that I is not a minimal ground instance in $\mathsf{RCQ}^s(Q, D_m, V)$, that is, either I is not in $\mathsf{RCQ}^s(Q, D_m, V)$ or I is in $\mathsf{RCQ}^s(Q, D_m, V)$, but there exists $I' \subsetneq I$ such that I' is also in $\mathsf{RCQ}^s(Q, D_m, V)$. The former is checked by the Σ_2^p oracle in Step (2)(b). The latter is inspected by Step (2)(c) of the algorithm, which tests whether there exists a tuple $t = \mu(\tau_i)$ in I such that $I \backslash \{t\}$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. 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. When it comes to ground instances, we show that $\mathsf{MINP}^s(\mathcal{L}_Q)$ is D_2^p-complete, when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$. It suffices to prove that, in this setting, $\mathsf{MINP}(\mathrm{CQ})$ is D_2^p-hard and that $\mathsf{MINP}^s(\exists\mathrm{FO}^+)$ is in D_2^p.

Lower bound. We first show that $\mathsf{MINP}^s(\mathrm{CQ})$ is D_2^p-hard by reduction from the $\exists^*\forall^*3\mathrm{DNF}$-$\forall^*\exists^*3\mathrm{CNF}$ problem, which is D_2^p-complete [Wooldridge and Dunne 2004]. An instance of $\exists^*\forall^*3\mathrm{DNF}$-$\forall^*\exists^*3\mathrm{CNF}$ is a pair of $\forall^*\exists^*3\mathrm{SAT}$ instances $\varphi_1 = \forall X_1 \exists Y_1 \psi_1(X_1, Y_1)$ and $\varphi_2 = \forall X_2 \exists Y_2 \psi_2(X_2, Y_2)$. It is to decide whether φ_1 is true and φ_2 is false. Given (φ_1, φ_2), we define $\mathcal{R}$, $\mathcal{I}$, D_m, V, and Q, such that φ_1 is true and φ_2 is false if and only if $\mathcal{I}$ is a minimal instance in $\mathsf{RCQ}^s(Q, D_m, V)$.

(a) The database schema $\mathcal{R}$ consists of six relation schemas: $R_{(0,1)}(A)$, $R_\neg(A, \bar{A})$, $R_\vee(A_1, A_2, B)$, and $R_\wedge(A_1, A_2, B)$, 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 later.

(b) We construct a ground instance $\mathcal{I} = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I_1, I_2)$, in which $I_{(0,1)}$, $I_\neg$, $I_\vee$, and $I_\wedge$ are ground relations, as shown in Figure 2, $I_1 = \{(1)\}$ and $I_2 = \{(0), (1)\}$.

(c) Master data D_m is specified by six relation schemas: $R^m_{(0,1)} = R_{(0,1)}$, $R^m_\neg = R_\neg$, $R^m_\vee = R_\vee$, $R^m_\wedge = R_\wedge$, $R^m_1 = R_1$, and $R^m_2 = R_2$. The master data instance consists of $I^m_{(0,1)} = I_{(0,1)}$, $I^m_\neg = I_\neg$, $I^m_\vee = I_\vee$, $I^m_\wedge = I_\wedge$, and $I^m_1 = I^m_2 = \{(0), (1)\}$.

(d) The set V consists of the following CCs: $R_{(0,1)} \subseteq R^m_{(0,1)}$, $R_\neg \subseteq R^m_\neg$, $R_\vee \subseteq R^m_\vee$, $R_\wedge \subseteq R^m_\wedge$, $R_1 \subseteq R^m_1$, and $R_2 \subseteq R^m_2$.

(e) The CQ query $Q(\vec{x}_1, \vec{x}_2)$ is defined as $Q_1(\vec{x}_1) \wedge Q_2(\vec{x}_2) \wedge Q_{\mathsf{all}}$, where $\vec{x}_1$ and $\vec{x}_2$ correspond to the X-variables in φ_1 and φ_2, respectively. The query Q_{all} is used to ensure that (i) all tuples in I_{01}, $I_\neg$, $I_\vee$, and $I_\wedge$ are present; and (ii) that I_1 and I_2 contain (1). Otherwise, Q_{all} returns false (empty set). The queries $Q_i(\vec{x}_i)$ for $i = 1, 2$ are defined as

$$Q_i(\vec{x}_i) = \exists \vec{y}_i, w_i \, (Q_{X_i}(\vec{x}_i) \wedge Q_{Y_i}(\vec{y}) \wedge Q_{\psi_i}(\vec{x}_i, \vec{y}_i, w_i) \wedge R_i(w_i)),$$

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_{01}; and Q_{ψ_i} is a CQ query encoding ψ_i by leveraging relations $I_\neg$, $I_\vee$, and $I_\wedge$. More specifically, for given truth assignments μ_{X_i} and μ_{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 $I_i = \{(1)\}$, then $Q_i(\vec{x}_i)$ returns all truth assignments μ_{X_i} for X_i for which there exists a truth assignment μ_{Y_i} of Y_i that satisfies ψ_i; and (b) if $I_i = \{(0), (1)\}$, then $Q_i(\vec{x}_i)$ returns all possible truth assignments of X_i.

We next verify that φ_1 is true and φ_2 is false if and only if $\mathcal{I}$ is a minimal instance in $\mathsf{RCQ}(Q, D_m, V)$.

$\boxed{\Rightarrow}$ Suppose that φ_1 is true and φ_2 is false. Observe that $Q(\mathcal{I}) = F_{X_1} \times F_{X_2}$, where F_{X_i} consists of all possible truth assignments for X_i. $Q_1(\mathcal{I}) = F_{X_1}$ because φ_1 is true, whereas $Q_2(\mathcal{I}) = F_{X_2}$ because I_2 consists of both (0) and (1). Furthermore, observe that $\mathcal{I}$ is also complete. The only possible extension of $\mathcal{I}$ is $\mathcal{I}' = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I'_1, I_2)$, with $I'_1 = \{(0), (1)\}$. Clearly, $Q(\mathcal{I}') = Q(\mathcal{I})$ since $Q(\mathcal{I})$ already generates the largest possible query result. That is, $\mathcal{I} \in \mathsf{RCQ}^s(Q, D_m, V)$. We next show that $\mathcal{I}$ is also minimal. For this, it suffices to observe that, among all possible subsets of $\mathcal{I}$, the instance $\mathcal{I}'' = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I_1, I''_2)$ with $I''_2 = \{(1)\}$ is the only one that can possibly lead to $Q(\mathcal{I}'') \neq \emptyset$. All other subinstances of $\mathcal{I}$ make Q_{all} return empty; thus, these cannot be complete because $Q(\mathcal{I}) \neq \emptyset$. It remains to show that $Q(\mathcal{I}'') \subsetneq Q(\mathcal{I})$, thus $\mathcal{I}''$ is not in $\mathsf{RCQ}^s(Q, D_m, V)$. To see this, recall that φ_2 is false. This implies that there exists a truth assignment $\mu^0_{X_2}$ of X_2 for which all truth assignments μ_{Y_2} of Y_2 make ψ false. Since $I''_2 = \{(1)\}$, $Q_2(\mathcal{I}'')$ will not return $\mu^0_{X_2}$. On the other hand, $\mu^0_{X_2} \in Q_2(\mathcal{I})$. Since $Q_1(\mathcal{I}) = Q_1(\mathcal{I}'')$, we can conclude that $Q(\mathcal{I}'') \subsetneq Q(\mathcal{I})$.

$\boxed{\Leftarrow}$ Suppose that φ_1 is false or φ_2 is true. We distinguish between the following two cases: (i) φ_1 is false and (ii) both φ_1 and φ_2 are true. For case (i), we immediately have that $\mathcal{I}$ is not in $\mathsf{RCQ}^s(Q, D_m, V)$, thus cannot be minimal. Consider the unique extension $\mathcal{I}'$ of $\mathcal{I}$ described earlier. Clearly, $Q(\mathcal{I}') = F_{X_1} \times F_{X_2}$. However, since φ_1 is false and I_1 contains only (1), $Q_1(\mathcal{I})$ will not include at least one truth assignment of X_1. Hence, $Q(\mathcal{I}) \subsetneq Q(\mathcal{I}')$. For case (ii), since φ_1 and φ_2 are both true, $Q(\mathcal{I}) = F_{X_1} \times F_{X_2}$; thus, $Q(\mathcal{I}') = Q(\mathcal{I})$ since no more result tuples can be added. That is, $\mathcal{I}$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. We show that $\mathcal{I}$ is not minimal. Consider the subinstance $\mathcal{I}''$ described earlier with $I''_2 = \{(1)\}$. We claim that $\mathcal{I}''$ is in $\mathsf{RCQ}^s(Q, D_m, V)$. To see this, observe that the only extensions of $\mathcal{I}''$ are $\mathcal{I}$, $\mathcal{I}'$, and $\mathcal{I}''' = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I'_1, I''_2)$. Since φ_1 is true, adding (0) to I_1 (as done in the extensions $\mathcal{I}'$ and $\mathcal{I}'''$) does not affect $Q_1(\mathcal{I}'')$. Similarly, adding (0) to I''_2 (as done in the extensions $\mathcal{I}$ and $\mathcal{I}'$) does not affect $Q_2(\mathcal{I}'')$ since φ_2 is true. Hence,

$Q(\mathcal{I}'') = Q(\mathcal{I}) = Q(\mathcal{I}') = Q(\mathcal{I}''')$ and $\mathcal{I}''$ is indeed in $\mathsf{RCQ}^s(Q, D_m, V)$; this shows that $\mathcal{I}$ is not minimal.

Upper bound. We show that, for ground instances, $\mathsf{MINP}(\exists\mathrm{FO}^+)$ is in D_2^p. By Lemma 4.7, the set of yes-instances to $\mathsf{MINP}(\exists\mathrm{FO}^+)$ is $L_1 \cap L_2$, where

—$L_1 = \{(I, Q, D_m, V) \mid I \in \mathsf{RCQ}^s(Q, D_m, V)\}$; and
—$L_2 = \{(I, Q, D_m, V) \mid \text{for all } t \in I,\ I\backslash\{t\} \notin \mathsf{RCQ}(Q, D_m, V)\}$.

It now suffices to show that $L_1 \in \Pi_2^p$ and $L_2 \in \Sigma_2^p$. Clearly, $L_1 \in \Pi_2^p$ follows from Theorem 4.1. To show that $L_2 \in \Sigma_2^p$, we modify the algorithm for the complement problem of $\mathsf{RCDP}(\exists\mathrm{FO}^+)$ 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 $I_i = I\backslash\{t_i\}$ for $i \in [1, k]$. We then apply the Σ_2^p-algorithm for each (I_i, Q, D_m, V) "in parallel," that is, the algorithm guesses k tuples s_i (by means of a valuation of the query Q) such that $s_i \in Q(I_i')\backslash Q(I_i)$ 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_is are in $\mathsf{RCQ}^s(Q, D_m, V)$. In other words, $(I, Q, D_m, V) \in L_2$. □

5. WEAK RELATIVE INFORMATION COMPLETENESS

We next investigate RCDP, RCQP, and MINP for weakly complete databases, denoted by RCDP^w, RCQP^w, and MINP^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 $\mathsf{RCQ}^w(Q, D_m, V)$ 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 the one hand, the certain-answer semantics simplifies the analysis of some problems; for example, 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; for example, MINP^w becomes Π_4^p-complete for UCQ, as opposed to Π_3^p for MINP^s. In addition, some problems even have different bounds for CQ and UCQ; for example, MINP^w is coDP-complete for CQ, while it is Π_4^p-complete for UCQ.

5.1. The Relatively Complete Database Problem in the Weak Model

As opposed to Theorem 4.1, RCDP^w is decidable for FP. In addition, RCDP^w for c-instances and RCDP^w for ground instances are both Π_3^p-complete when $\mathcal{L}_Q$ is CQ, UCQ or $\exists\mathrm{FO}^+$, while their counterparts in the strong model are Π_2^p-complete (Theorem 4.1).

THEOREM 5.1. *For c-instances and for ground instances,* $\mathsf{RCDP}^w(\mathcal{L}_Q)$ *is*

—*undecidable when* $\mathcal{L}_Q$ *is FO,*
—*co*NEXPTIME*-complete when* $\mathcal{L}_Q$ *is FP, and*
—Π_3^p*-complete when* $\mathcal{L}_Q$ *is CQ, UCQ or* $\exists$*FO*$^+$.

PROOF. We show that, in the weak model, $\mathsf{RCDP}^w(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ is FO, coNEXPTIME-complete when $\mathcal{L}_Q$ is FP, and it becomes Π_3^p-complete when $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$. The lower bounds hold even when only ground instances are considered, and when D_m and V are fixed.

(1) When $\mathcal{L}_Q$ 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 without loss of generality 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, $Q(I) = \{()\}$ 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 $\mathsf{RCQ}^w(Q, D_m, V)$.

$\boxed{\Rightarrow}$ First, assume that q is not satisfiable, that is, for all instances I of R, $q(I) = \emptyset$. Then, for all $I \in \mathsf{Ext}(\emptyset)$, $Q(I) = \{()\}$; hence, $\bigcap_{I \in \mathsf{Ext}(\emptyset)} Q(I) = \{()\} = Q(\emptyset)$. Thus, $\emptyset$ is in $\mathsf{RCQ}(Q, D_m, V)$.

$\boxed{\Leftarrow}$ Conversely, assume that q is satisfiable, that is, there exists an instance I_0 of R such that $q(I_0)$ is not empty. Then, by the definition of Q, $Q(I_0) = \emptyset = \bigcap_{I \in \mathsf{Ext}(\emptyset)} Q(I) \neq Q(\emptyset) = \{()\}$, since $q(\emptyset) = \emptyset$. Hence, $\emptyset$ is not in $\mathsf{RCQ}(Q, D_m, V)$.

We remark that, in this proof, 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.

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

Lower bound. We show that, for ground instances, $\mathsf{RCDP}^w(\mathrm{FP})$ is coNEXPTIME-hard by reduction from the SUCCINCT-TAUT problem, which is coNEXPTIME-complete (see 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 $a_i \in \{\wedge, \vee, \neg, \mathsf{in}\}$ is the type of the gate g_i, g_j and g_k for $j, k < i$ are the inputs of the gate (unless g_i is an in-gate, in which case $j = k = 0$, or unless g_i is a $\neg$-gate, in which case $j = k$). Suppose that C has n input gates; then, C defines the Boolean function $f_C : \{0,1\}^n \to \{0,1\}$, 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 $\bar{w} \in \{0,1\}^n$, $f_C(\bar{w}) = 1$, that is, whether C is a tautology.

Given an instance of the latter problem, we define database schemas $\mathcal{R}$ and $\mathcal{R}_m$, a ground instance $\mathcal{I}$ of $\mathcal{R}$, a set V of CCs, master data D_m of $\mathcal{R}_m$, and an FP query Q. We show that C is a tautology if and only if $\mathcal{I}$ is a minimal instance in $\mathsf{RCQ}^w(Q, D_m, V)$.

(a) The database schema $\mathcal{R}$ consists of a single relation $R(A_0, A_1, \ldots, A_{30})$, where, for a tuple t of R, $t[A_1, A_2]$ is to encode $I_{(0,1)}$; $t[A_3, \ldots, A_{14}]$ encodes $I_\vee$; $t[A_{15}, \ldots, A_{26}]$ encodes $I_\wedge$, and $t[A_{27}, \ldots, A_{30}]$ encodes $I_\neg$.

(b) A ground instance $\mathcal{I}$ of $\mathcal{R}$ consists of a single tuple t that is formed by juxtaposing instances $I_{(0,1)}$, $I_\vee$, $I_\wedge$, and $I_\neg$ in $t[A_1, \ldots, A_{30}]$, with $t[A_0] = 1$. Here, $I_{(0,1)}$, $I_\vee$, $I_\wedge$, 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) $t[A_1, \ldots, A_{30}]$ encodes the instances mentioned earlier; and (ii) $t[A_0]$ only takes values from $\{0, 1\}$.

(d) The query Q in FP uses an $(n+1)$-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:

$$
\begin{array}{lll}
 & I(x) & \leftarrow R(A_0, x, A_2, \ldots, A_{30});\\
 & I(x) & \leftarrow R(A_0, A_1, x, A_3, \ldots, A_{30});\\
 & R_X(\vec{x}) & \leftarrow I(x_1), \ldots, I(x_n);\\
\text{If } a_i = \mathsf{in}, \text{ then} & G_i(B, \vec{x}) & \leftarrow R_X(\vec{x}), B = x_i;\\
\text{If } a_i = \vee, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), G_k(B_2, \vec{x}), R(A_0, A_1, A_2, B_1, B_2, B, A_6, \ldots, A_{30});\\
 & \vdots & \vdots\\
\text{If } a_i = \vee, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), G_k(B_2, \vec{x}), R(A_0, \ldots, A_{11}, B_1, B_2, B, A_{15}, \ldots, A_{30});\\
\text{If } a_i = \wedge, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), G_k(B_2, \vec{x}), R(A_0, \ldots, A_{14}, B_1, B_2, B, A_{18}, \ldots, A_{30});\\
 & \vdots & \vdots\\
\text{If } a_i = \wedge, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), G_k(B_2, \vec{x}), R(A_0, \ldots, A_{23}, B_1, B_2, B, A_{27}, \ldots, A_{30});\\
\text{If } a_i = \neg, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), R(A_0, \ldots, A_{26}, B_1, B, A_{29}, A_{30});\\
\text{If } a_i = \neg, \text{ then} & G_i(B, \vec{x}) & \leftarrow G_j(B_1, \vec{x}), R(A_0, \ldots, A_{28}, B_1, B);
\end{array}
$$

and, finally, two more rules:

$$
\begin{array}{ll}
G(\vec{x}) & \leftarrow G_M(B, \vec{x}), R(0, A_1, \ldots, A_{30}),\\
G(\vec{x}) & \leftarrow G_M(B, \vec{x}), B = 1,
\end{array}
$$

where G_M is the IDB corresponding to the output gate g_M. Intuitively, $Q(\mathcal{I})$ 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 $\mathcal{I} \in \mathsf{RCQ}^w(Q, D_m, V)$.

$\boxed{\Rightarrow}$ First, assume that C is a tautology. We show that $\mathcal{I}$ is weakly complete for Q relative to (D_m, V). Since for all $\bar{w} \in \{0, 1\}^n$, $f_C(\bar{w}) = 1$, $Q(\mathcal{I})$ will return all $\bar{w} \in \{0, 1\}^n$. Observe that the only extension $\mathcal{I}'$ of $\mathcal{I}$ is $\{t, t'\}$, where t is in $\mathcal{I}$ and t' is the same as t except that $t'[A_0] = 0$ while $t[A_0] = 1$. Obviously, we have that $Q(\mathcal{I}')$ returns all $\bar{w} \in \{0, 1\}^n$ as well. Hence, $\mathcal{I}$ is weakly complete.

$\boxed{\Leftarrow}$ Conversely, suppose that C is not a tautology but $\mathcal{I}$ is weakly complete. Note again that the ground instance $\mathcal{I}'$ mentioned earlier 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 $Q(\mathcal{I})$ contains only those $\bar{w} \in \{0, 1\}^n$ for which $f_C(\bar{w}) = 1$ (recall that $t[A_0] = 1$). Hence, $\mathcal{I}$ cannot be weakly complete.

Upper bound. We show that RCDP^w(FP) is in coNEXPTIME by providing an NEXPTIME algorithm that decides the complement problem. That is, given a c-instance $\mathcal{T}$, master data D_m, a set V of CCs, and an FP query Q, the algorithm returns "yes" if $\mathcal{T}$ 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 following lemma. By Lemma 3.2, we assume without loss of generality that $\mathcal{R}$ consists of a single relation schema R and $\mathcal{T}$ is a c-table (T, ξ). Recall the notion of Adom given in the proof of Theorem 4.1.

LEMMA 5.2. *For every Q in FP, master data D_m, set V of CCs, and any c-instance $\mathcal{T} = (T, \xi)$, let $\mathcal{T}' = (T \cup \{(x_1, \ldots, x_n), \xi\})$, that is, $\mathcal{T}$ extended with a single tuple consisting of (new) variables only. Then, (T, ξ) is not in $\mathsf{RCQ}^w(Q, D_m, V)$ if and only if*

there exist a tuple t and an instance $\mathcal{I} \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T})$ such that $t \in \bigcap_{\mathcal{I}' \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T}')} Q(\mathcal{I}')$ and $t \notin Q(\mathcal{I})$.

Proof. First, assume that $(\mathcal{T}, \xi)$ is not in $\mathsf{RCQ}^w(Q, D_m, V)$. Then, $\bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T})} Q(\mathcal{I}) \neq \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$. By the monotonicity of FP, we have that $\bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T})} Q(\mathcal{I}) \subseteq \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$. Thus, there must exist a tuple $t' = (a_1, \ldots, a_n)$ such that $t' \in \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$ and $t' \notin \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T})} Q(\mathcal{I})$. In other words, there exists an $\mathcal{I} \in \mathsf{Mod}(\mathcal{T})$ such that $t' \in \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$ and $t' \notin Q(\mathcal{I})$. Note that $\bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}') \subseteq \bigcap_{\mathcal{I}' \in \mathsf{Mod}(\mathcal{T}')} Q(\mathcal{I}')$, and then $t' \in \bigcap_{\mathcal{I}' \in \mathsf{Mod}(\mathcal{T}')} Q(\mathcal{I}')$. 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 $\mathcal{T}$, 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 $\bigcap_{\mathcal{I}' \in \mathsf{Mod}(\mathcal{T}')} Q(\mathcal{I}')$ and then in $\bigcap_{\mathcal{I}' \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T}')} Q(\mathcal{I}')$, but t is not in $Q(\mathcal{I})$.

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

Capitalizing on the characterization, we next present the NEXPTIME algorithm.

(1) Guess a tuple t of the (output) schema of Q with values from Adom.
(2) Check whether $t \notin Q(\mu(\mathcal{T}))$ for some valuation μ of $\mathcal{T}$ taking values from Adom; if so, continue; otherwise, reject the guess. Since the valuations range over a finite domain, each $\mu(\mathcal{T})$ is of polynomial size, and evaluating FP queries takes EXPTIME, the total cost of this process is EXPTIME.
(3) For $\mathcal{T}' = \mathcal{T} \cup \{(x_1, \ldots, x_n)\}$, we test whether $t \in Q(\mu'(\mathcal{T}'))$ for each valuation μ' of $\mathcal{T}'$ such that $(\mu'(\mathcal{T}'), D_m) \models V$, $(\mu'(\mathcal{T}), D_m) \models V$ and $\mu'(\mathcal{T}) \subsetneq \mu'(\mathcal{T}')$. If successful, the algorithm returns "yes." Otherwise, the current guess is rejected. For the same reason as outlined earlier, this process takes EXPTIME.

The overall complexity of the algorithm is NEXPTIME; hence, $\mathsf{RCDP}^w(\mathrm{FP})$ is in coNEXPTIME. Obviously, the algorithm is correct, by Lemma 5.2. The algorithm returns "yes" if and only if there exists a tuple t and instance $\mathcal{I} \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T})$ such that $t \in \bigcap_{\mathcal{I}' \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T}')} Q(\mathcal{I}')$ and $t \notin Q(\mathcal{I})$.

(3) When $\mathcal{L}_Q$ is CQ, UCQ, or $\exists\mathrm{FO}^+$. It suffices to show that $\mathsf{RCDP}^w(\mathrm{CQ})$ is Π_3^p-hard for ground instances and $\mathsf{RCDP}^w(\exists\mathrm{FO}^+)$ is in Π_3^p for c-instances.

Lower bound. We show that, for ground instances, $\mathsf{RCDP}^w(\mathrm{CQ})$ is Π_3^p-hard by reduction from the complement of the $\exists^*\forall^*\exists^*3\mathrm{SAT}$-problem. Given a formula $\varphi = \exists X \forall Y \exists Z \psi$, $\exists^*\forall^*\exists^*3\mathrm{SAT}$ is to determine whether φ is true. Here, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, and $Z = \{z_1, \ldots, z_l\}$, which are sets of variables; and $\psi = C_1 \wedge \cdots \wedge C_r$ is an instance of 3SAT. It is known that $\exists^*\forall^*\exists^*3\mathrm{SAT}$ is Σ_3^p-complete (see Papadimitriou [1994]).

Given an instance $\varphi = \exists X \forall Y \exists Z \psi$ of $\exists^*\forall^*\exists^*3\mathrm{SAT}$, we define database schemas $\mathcal{R}$ and $\mathcal{R}_m$, a ground instance $\mathcal{I}$, a set V of CCs, master data D_m, and a CQ query Q. We show that φ is true if and only if $\mathcal{I}$ is not weakly complete for Q relative to (D_m, V).

(a) The database schema $\mathcal{R}$ consists of five relation schemas: $R_{(0,1)}(A)$, $R_\neg(A, \bar{A})$, $R_\vee(A_1, A_2, B)$, and $R_\wedge(A_1, A_2, B)$, 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 $\mathcal{I}$ is given by $(I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I_Y)$. Here, $I_{(0,1)}$, $I_\neg$, $I_\vee$, and $I_\wedge$ 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 $R^m_{(0,1)} = R_{(0,1)}$, $R^m_\neg = R_\neg$, $R^m_\vee = R_\vee$, $R^m_\wedge = R_\wedge$, and $R_\emptyset(W, W')$. The master data instance D_m consists of $I^m_{(0,1)} = I_{(0,1)}$, $I^m_\neg = I_\neg$, $I^m_\vee = I_\vee$, $I^m_\wedge = I_\wedge$, and $I^m_\emptyset = \emptyset$.

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

—$R_{(0,1)} \subseteq R^m_{(0,1)}$, $R_\vee \subseteq R^m_\vee$, $R_\wedge \subseteq R^m_\wedge$, $R_\neg \subseteq R^m_\neg$;
—$\phi_i : q_i(y_i) \subseteq R_{(0,1)}$, 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
—$\phi'_i : q'_i(y_i, y'_i) \subseteq R_\emptyset$, where, for $i \in [1, m]$, $q'_i(y_i, y'_i) = \exists y_1 y'_1 \ldots y_{i-1} y'_{i-1} y_{i+1} y'_{i+1} \ldots y_m y'_m$ $(R_Y(y_1, \ldots, y_m) \wedge R_Y(y'_1, \ldots, y'_m) \wedge y_i \neq y'_i)$.

It is easy to see that, for each extension I'_Y of I_Y, $I'_Y \models \bigwedge_{i\in[1,m]}(\phi_i \wedge \phi'_i)$ 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:

$$Q(\vec{x}) = \exists \vec{y}, \vec{z}\, Q_X(\vec{x}) \wedge R_Y(\vec{y}) \wedge Q_Z(\vec{z}) \wedge Q_\psi(\vec{x}, \vec{y}, \vec{z}, w) \wedge w = 1,$$

where $\vec{x} = (x_1, \ldots, x_n)$, $\vec{y} = (y_1, \ldots, y_m)$, and $\vec{z} = (z_1, \ldots, z_l)$. The subqueries $Q_X(\vec{x}) = \bigwedge_{i=1}^m R_{01}(x_i)$ and $Q_Z(\vec{z}) = \bigwedge_{i=1}^l R_{01}(z_i)$ generate all valid truth assignments of X variables and Z variables, respectively, by means of Cartesian products of R_{01}. Given truth assignments (μ_X, μ_Y, μ_Z) of (X, Y, Z), query Q_ψ is to encode the truth value of $\psi(\mu_X, \mu_Y, \mu_Z)$ as the value of w. Obviously, Q_ψ can be expressed in CQ in terms of $R_\vee$, $R_\wedge$, and $R_\neg$. Intuitively, query Q returns all truth assignments μ_X of X if there exists truth assignments μ_Y and μ_Z of Y and Z, respectively, that make ψ true.

We next show that the reduction is correct, that is, φ is true if and only if $\mathcal{I}$ is not weakly complete for Q relative to (D_m, V).

$\boxed{\Rightarrow}$ First, assume that φ is true. Then, there exists a truth assignment μ^0_X of X such that, for each truth assignment μ_Y of Y, there exists a truth assignment μ_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 $Q(\mathcal{I}) \neq \bigcap_{\mathcal{I}'\in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$. Observe the following: (i) $Q(\mathcal{I}) = \emptyset$ since I_Y is empty; and (ii) for each instance $\mathcal{I}'$ in $\mathsf{Ext}(\mathcal{I})$, $\mathcal{I}' = (I_{(0,1)}, I_\neg, I_\vee, I_\wedge, I'_Y)$, where I'_Y encodes a valid truth assignment of Y, and $(\vec{x}_0) \in Q(\mathcal{I}')$ represents the truth assignment μ^0_X. Thus, $(\vec{x}_0) \in \bigcap_{\mathcal{I}'\in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$. Hence, $\mathcal{I}$ is not weakly complete for Q relative to (D_m, V).

$\boxed{\Leftarrow}$ Conversely, if φ is false, then there exists no truth assignment μ_X of X such that, for all truth assignments of Y, there exists a truth assignment of Z that makes ψ true. Recall that, for each partially closed extension $\mathcal{I}'$ of $\mathcal{I}$, $\mathcal{I}'$ can only be the form of $(I_{(01)}, I_\neg, I_\vee, I_\wedge, I'_Y)$, where I'_Y encodes a truth assignment of Y. By the definition of Q, if there exists a tuple t in $\bigcap_{\mathcal{I}'\in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$, 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 ψ true. Thus, φ is true, which contradicts the assumption that φ is false. As a result, $\bigcap_{\mathcal{I}'\in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$ must be empty, and $\mathcal{I}$ is weakly complete for Q relative to (D_m, V).

Upper bound. It suffices to show that $\mathsf{RCDP}^w(\exists\mathrm{FO}^+)$ is in Π^p_3. To do this, we use the same algorithm given for $\mathsf{RCDP}^w(\mathrm{FP})$. We show the algorithm is in Σ^p_3; then, $\mathsf{RCDP}^w(\exists\mathrm{FO}^+)$ is in Π^p_3. Step (2) of the algorithm can be done in Σ^p_2 by the following procedure.

(1) Guess a valuation μ of $\mathcal{T}$ by taking values from Adom.
(2) Check whether $\mu(\mathcal{T})$ satisfies ξ. If so continue; otherwise, reject the guess.
(3) Check whether $t \notin Q(\mu(\mathcal{T}))$. If so, return "yes"; otherwise reject the guess.

Obviously, it returns "yes" if and only if there exists a valuation of $\mathcal{T}$ such that $t \notin Q(\mu(\mathcal{T}))$. It is in Σ_2^p since Step (2) is in PTIME and Step (3) is in coNP for CQ.

Moreover, Step (3) of the algorithm is also in Σ_2^p for $\exists$FO$^+$. We give a Π_2^p procedure for the complement of Step (3), that is, it returns "no" if and only if $t \notin \bigcap_{I' \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T}')} Q(I')$.

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

It returns "no" if and only if $t \notin \bigcap_{I' \in \mathsf{Mod}_{\mathsf{Adom}}(\mathcal{T}')} Q(I')$. It is in Π_2^p since Step (2) is in PTIME, Step (3) is in NP, and Steps (4) and (5) are both in coNP, for $\exists$FO$^+$queries. □

5.2. The Relatively Complete Query Problem in the Weak Model

Recall that RCQP^s for c-instances is equivalent to RCQP^s for ground instances, as verified by Lemma 4.4. However, the following example 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 RCQ$^w(Q, D_m, V)$ since $Q(I_1, I_2) \neq \emptyset$ for all (I_1, I_2), while $\bigcap_{\mathcal{I}' \in \mathsf{Ext}(I_1, I_2)} Q(\mathcal{I}') = \emptyset$. In contrast, consider a c-instance $\mathcal{T} = (T_1, T_2)$, where $T_1 = (\{(x)\}, \emptyset)$ and $T_2 = (\{(y)\}, \emptyset)$. Obviously, $\mathcal{T}$ is in RCQ$^w(Q, D_m, V)$ since $Q(\mathcal{T}) = \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T})} Q(\mathcal{I}) = \emptyset = \bigcap_{\mathcal{I} \in \mathsf{Mod}(\mathcal{T}), \mathcal{I}' \in \mathsf{Ext}(\mathcal{I})} Q(\mathcal{I}')$.

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

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 refer to Appendix A for the proof of this Theorem. In a nutshell, we show that RCQP^w(FO) is undecidable for ground instances by reduction from the satisfiability problem for 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 FP query. From this, it follows that RCQP^w(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, that is, to decide whether an instance I is minimal, it does not suffice to inspect $I \backslash \{t\}$ only.

Example 5.5. Consider a CQ query Q defined on a pair of unary relations (R_1, R_2): $Q(x) = \exists y\, z(R_1(y) \wedge R_2(z) \wedge x = a)$. 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 nonempty. Consider an instance $\mathcal{I}_0 = (\{(0)\}, \{(1)\})$, an empty set V of CCs, and any master data D_m. Then, $\mathcal{I}_0$ is in $\mathsf{RCQ}^w(Q, D_m, V)$. Nevertheless, it is not minimal: the empty instance $(\emptyset, \emptyset)$ of (R_1, R_2) is also in $\mathsf{RCQ}^w(Q, D_m, V)$. Indeed, $(\{0\}, \emptyset)$ is a partially closed extension of $(\emptyset, \emptyset)$ and $Q(\{0\}, \emptyset) = \emptyset$. Then, $\bigcap_{I' \in \mathsf{Ext}((\emptyset,\emptyset))} = \emptyset$; thus, $(\emptyset, \emptyset)$ is in $\mathsf{RCQ}^w(Q, D_m, V)$. However, removing one tuple from $\mathcal{I}_0$ does not make it a weakly complete instance, that is, a counterexample to the minimality of $\mathcal{I}_0$ cannot be found by removing only one tuple from $\mathcal{I}_0$.

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 (D_2^p for ground instances vs. Π_3^p for c-instances). (b) It is much easier to check $\mathsf{MINP}^w(\mathrm{CQ})$ than $\mathsf{MINP}^w(\mathrm{UCQ})$ (coDP-complete vs. Π_4^p-complete), whereas $\mathsf{MINP}^s(\mathrm{CQ})$ and $\mathsf{MINP}^s(\mathrm{UCQ})$ have the same complexity. Recall that $\mathrm{coDP} = \mathsf{NP} \cup \mathsf{coNP}$ (see Papadimitriou [1994]).

THEOREM 5.6. *For c-instances and ground instances.* $\mathsf{MINP}^w(\mathcal{L}_Q)$ *is*

—*undecidable when $\mathcal{L}_Q$ is FO,*
—*coNEXPTIME-complete when $\mathcal{L}_Q$ is FP,*
—*Π_4^p-complete when $\mathcal{L}_Q$ is UCQ or $\exists$FO$^+$, and*
—*coDP-complete when $\mathcal{L}_Q$ is CQ.*

PROOF. We show that $\mathsf{MINP}^w(\mathcal{L}_Q)$ is undecidable when $\mathcal{L}_Q$ is FO, coNEXPTIME-complete when $\mathcal{L}_Q$ is FP, and Π_4^p-complete when $\mathcal{L}_Q$ is UCQ or $\exists$FO$^+$. When $\mathcal{L}_Q$ is CQ, the problem is shown to be coDP-complete. All lower bounds remain intact when only ground instances are considered.

(1) When $\mathcal{L}_Q$ is FO. It suffices to show that $\mathsf{MINP}^w(\mathrm{FO})$ 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 $\mathcal{I}_\emptyset$ is in $\mathsf{RCQ}^w(Q, D_m, V)$, where $\mathcal{I}_\emptyset$ is the empty database instance of the schema over which Q is defined. This is because if $\mathcal{I}_\emptyset$ is in $\mathsf{RCQ}^w(Q, D_m, V)$, 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 $\mathcal{L}_Q$ is FP. We show that $\mathsf{MINP}^w(\mathrm{FP})$ is coNEXPTIME-hard for ground instances and provide a coNEXPTIME algorithm for deciding $\mathsf{MINP}^w(\mathrm{FP})$ for c-instances.

Lower bound. We show that, for ground instances, $\mathsf{MINP}^w(\mathrm{FP})$ 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 $\mathcal{R}$ and $\mathcal{R}_m$, ground instance $\mathcal{I}$ of $\mathcal{R}$, set V of CCs, master data D_m of $\mathcal{R}_m$, 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 $\mathcal{I}$ is the minimal in $\mathsf{RCQ}^w(Q, D_m, V)$.

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

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

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

(1) Check whether $\mathcal{T} \in \mathsf{RCQ}^w(Q, D_m, V)$. This is a coNEXPTIME process by Theorem 5.1(2). If so, continue; otherwise, return "no."
(2) For each $\Delta \subseteq \mathcal{T}$, $\Delta \neq \emptyset$, check whether $\mathcal{T} \backslash \Delta \in \mathsf{RCQ}^w(Q, D_m, V)$. If such a Δ is found, return "no"; otherwise, return "yes." The enumeration of the subsets Δ and checking whether $\mathcal{T} \backslash \Delta \in \mathsf{RCQ}^w(Q, D_m, V)$ are again in coNEXPTIME.

Hence, MINP^w(FP) is in coNEXPTIME. The correctness of the algorithm is immediate.

(3) When $\mathcal{L}_Q$ is UCQ or $\exists$FO$^+$. We next show that $\mathsf{MINP}^w(\mathcal{L}_Q)$ is Π_4^p-complete when $\mathcal{L}_Q$ is UCQ or $\exists$FO$^+$. It suffices to show that MINP^w(UCQ) is Π_4^p-hard, and $\mathsf{MINP}^w(\exists\text{FO}^+)$ is in Π_4^p. The lower bound holds when considering ground instances only.

Lower bound. We show the Π_4^p-hardness by reduction from the $\forall^*\exists^*\forall^*\exists^*$3SAT problem, which is known to be Π_4^p-complete (see Papadimitriou [1994]). The $\forall^*\exists^*\forall^*\exists^*$3SAT problem is to determine, given a sentence $\varphi = \forall X \exists Y \forall Z \exists W \psi$, whether φ 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 ψ is an instance of 3SAT. Given an instance $\varphi = \forall X \exists Y \forall Z \exists W \psi$ of the $\forall^*\exists^*\forall^*\exists^*$3SAT problem, where $\psi = C_1 \wedge \cdots \wedge C_r$, 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 φ is true if and only if either $\mathcal{I}_0$ is not in $\mathsf{RCQ}^w(Q, D_m, V)$ or $\mathcal{I}_0 \in \mathsf{RCQ}^w(Q, D_m, V)$, but it is not minimal.

(a) The database schema $\mathcal{R}$ consists of the following six relation schemas: $R_{(0,1)}(X)$, $R_\vee(A_1, A_2, B)$, $R_\wedge(A_1, A_2, B)$, $R_\neg(A_1, A_2)$, $R_X(\mathsf{id}, X)$, and $R_Z(Z_1, \ldots, Z_k)$. Instances of $R_X(\mathsf{id}, X)$ are to encode truth assignments of X, and instances of R_Z are singleton sets, each encoding a truth assignment of Z. We let $\mathcal{I}_0 = (I_X, I_{(0,1)}, I_\vee, I_\wedge, I_\neg, I_Z)$, where $I_X = \{(1, 0), (1, 1), \ldots, (n, 0), (n, 1)\}$, $I_Z = \emptyset$, and $I_{(0,1)}$, $I_\vee$, $I_\wedge$, and $I_\neg$ 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 $I'_X \subseteq I_X$; (ii) any instance I'_Z of R_Z consists of a single tuple with values taken from $\{0, 1\}$; and (iii) instances of $R_{(0,1)}$, $R_\vee$, $R_\wedge$, and $R_\neg$ are subsets of $I_{(0,1)}$, $I_\vee$, $I_\wedge$, and $I_\neg$, respectively. Clearly, $(\mathcal{I}, D_m) \models V$.

(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_is 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 φ. As we will see here, $\mathcal{I}_0 \in \mathsf{RCQ}^w(Q, D_m, V)$, but it is only minimal when there does not exist a proper subset I_X^- of I_X such that $\mathcal{I}_0^- = (I_X^-, I_{(0,1)}, I_\vee, I_\wedge, I_\neg, \emptyset) \in \mathsf{RCQ}^w(Q, D_m, V)$. 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, $\mathcal{I}_0^- \in \mathsf{RCQ}^w(Q, D_m, V)$ if and only if the truth assignment μ_X encoded in I_X^- is such that there does not exist a μ_Y of Y such that, for every μ_Z of Z, $\exists W \psi(\mu_X, \mu_Y, \mu_Z, W)$ evaluates to true. In other words, $\mathcal{I}_0^- \in \mathsf{RCQ}^w(Q, D_m, V)$ if and only if φ is false. Consequently, $\mathcal{I}_0$ is minimal if and only if φ is true. In the rest of the lower bound proof, we use

$\mathcal{I}$ and $\mathcal{I}'$ to range over the instances of $\mathcal{R}$, $\mathcal{I}^-$, and $(\mathcal{I}')^-$ to denote subinstances (proper subsets) of $\mathcal{I}$ and $\mathcal{I}'$, and $\mathcal{I}^+$ and $(\mathcal{I}')^+$ to denote extensions 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 $2n + 13$, that is, there are at most $2n$ tuples in instances of R_X, 12 tuples in the instances corresponding to $R_{(0,1)}$, $R_\vee$, $R_\wedge$, and $R_\neg$, and at most one tuple in instances of R_Z.

For $i \in [1, 2n + 12]$, we define $Q_i(\vec{u})$ as a UCQ that returns $\vec{a}_i = (a_i, \ldots, a_i)$ 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 $Q' = Q_1 \cup \cdots \cup Q_{2n+12}$ and any instance $\mathcal{I}$ of $\mathcal{R}$ of size i. Then, $Q'(\mathcal{I}) = \{\vec{a}_1, \ldots, \vec{a}_i\}$. However, for any extension $\mathcal{I}^+$ of $\mathcal{I}$ (i.e., for $\mathcal{I}^+ \in \mathsf{Ext}(\mathcal{I})$), we have that $Q'(\mathcal{I}) \subsetneq Q'(\mathcal{I}^+)$ since the latter surely contains $\{\vec{a}_1, \ldots, \vec{a}_i, \vec{a}_{i+1}\}$. In other words, if we were to use only Q' instead of Q, no strict subinstance $\mathcal{I}^-$ of $\mathcal{I}_0$ can be in $\mathsf{RCQ}^w(Q', D_m, V)$. We will see shortly how the additional query P_1 in Q provides the opportunity for specific subinstances of $\mathcal{I}$, that is, those that correspond to valid truth assignments of X, to be in $\mathsf{RCQ}^w(Q, D_m, V)$. Observe that $Q'(\mathcal{I}_0) = \{\vec{a}_1, \ldots, \vec{a}_{2n+12}\}$. Similarly, $Q'(\mathcal{I}_0^+) = \{\vec{a}_1, \ldots, \vec{a}_{2n+12}\}$ 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 $\mathcal{I}_0 \in \mathsf{RCQ}^w(Q', D_m, V)$.

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

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

where Q_{all} is to ensure that all the tuples in $I_{(0,1)}$, $I_\neg$, $I_\vee$, and $I_\wedge$ are in place. That is, query P_1 puts $\vec{a}_{n+13}$ into the query result on instances $\mathcal{I}'$ of $\mathcal{R}$ for which I'_X contains a truth assignment μ_X of X, that is, when I'_X 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 an effect *only when* $\mathcal{I}'$ has size $n+12$ (when its R_X instance encodes a valid truth assignment for X). $\vec{a}_{n+13}$ is already in the query result for larger instances because of the Q_js described earlier. On the other hand, it cannot affect instances $\mathcal{I}'$ of smaller size. I'_X must contain at least n tuples and all 12 tuples in $I_{(0,1)}$, $I_\vee$, $I_\wedge$, and $I_\neg$ must be present.

It is easily verified that, for $Q'' = Q' \cup P_1$, $\mathcal{I}_0^- \in \mathsf{RCQ}^w(Q'', D_m, V)$ if and only if either (a) $\mathcal{I} = \mathcal{I}_0$ or (b) $\mathcal{I}_0^-$ consists of precisely $n + 12$ tuples and satisfies the condition in P_1. We refer to such subinstances $\mathcal{I}_0^-$ of $\mathcal{I}_0$ 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 $\mathcal{I}_0^-$ is equal to $\{\vec{a}_1, \ldots, \vec{a}_{n+12}, \vec{a}_{n+13}\}$, which equals $Q''(\mathcal{I}_0^-)$.

What remains to show is that no weakly complete candidates can be weakly complete if and only if φ is true. For if this holds, $\mathcal{I}_0$ is minimal in $\mathsf{RCQ}^w(Q, D_m, V)$ if and only if φ 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 $(\mathcal{I}_0^-)^+$ of weakly complete candidates $\mathcal{I}_0^-$, if its R_X instance $(I'_X)^+$ does not encode a truth assignment of X, then $(\mathcal{I}_0^-)^+$ does not affect the certain-answer result. More specifically, P_2 is a disjunction of n queries $P_{2,i}(\vec{y}) = R_X(i, 0) \wedge R_X(i, 1) \wedge R_{(0,1)}(y_1) \wedge \cdots \wedge R_{(0,1)}(y_m)$. That is, whenever P_2 is applied on an instance $(\mathcal{I}_0^-)^+$ 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 $Q'''(\mathcal{I}_0) = \{\vec{a}_1, \ldots, \vec{a}_{2n+12}\} \cup F_Y = Q'''(\mathcal{I}_0^+)$ for any extension of $\mathcal{I}_0^+$ of $\mathcal{I}_0$. Hence, $\mathcal{I}_0 \in \mathsf{RCQ}^w(Q''', D_m, V)$. Similarly, for a weakly complete

candidate $\mathcal{I}_0^-$, $Q'''(\mathcal{I}_0^-) = \{\vec{a}_1, \ldots, \vec{a}_{n+13}\} = \bigcap_{(\mathcal{I}_0^-)^+ \in \mathsf{Ext}(\mathcal{I}_0^-)} Q'''((\mathcal{I}_0^-)^+)$. This follows from the fact that $\mathsf{Ext}(\mathcal{I}_0^-)$ contains at least one instance $\mathcal{I}_1$ of size $n+13$ on which P_2 does not produce F_Y. For example, $\mathcal{I}_1$ could be $\mathcal{I}_0^-$ extended with an additional tuple in its R_Z instance I'_Z. Since $Q'''(\mathcal{I}_1) = Q(\mathcal{I}_0^-) = \{\vec{a}_1, \ldots, \vec{a}_{n+13}\} \subseteq \bigcap_{\mathcal{I}' \in \mathsf{Ext}(\mathcal{I}_0^-)} Q'''(\mathcal{I}')$ and $\mathcal{I}_1 \in \mathsf{Ext}(\mathcal{I}_0^-)$, we may conclude that P_2 alone does not prevent weakly complete candidates to be in $\mathsf{RCQ}^w(Q''', D_m, V)$. The relevance of P_2 comes only in play together with query P_3, which we define next.

More specifically,

$$P_3(\vec{y}) = \exists \vec{x}, \vec{z}, \vec{w}\,(Q_X(\vec{x}) \wedge Q_Y(\vec{y}) \wedge R_Z(\vec{z}) \wedge Q_W(\vec{w}) \wedge Q_\psi(\vec{x}, \vec{y}, \vec{z}, \vec{w}, 1)),$$

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

We now have that $Q = Q''' \cup P_3$. Observe that $Q(\mathcal{I}_0) = Q'''(\mathcal{I}_0)$. Furthermore, since $Q'''(\mathcal{I}_0)$ already contains F_Y, no further tuples can be added to the query result in any extension of $\mathcal{I}_0$. Hence, $\mathcal{I}_0$ remains a weakly complete database for Q, D_m, and V.

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

Upper bound. We show that $\mathsf{MINP}^w(\exists\mathrm{FO}^+)$ is in Π_4^p by providing an Σ_4^p-algorithm that decides the complement problem. That is, the algorithm returns "yes" if, for a given c-instance $\mathcal{T}$, master data D_m, a set V of CCs, and an $\exists\mathrm{FO}^+$query Q, $\mathcal{T}$ is not a minimal c-instance that is weakly complete for Q relative to (D_m, V), and "no" otherwise. The algorithm does the following:

(1) Check whether $\mathcal{T} \notin \mathsf{RCQ}^w(Q, D_m, V)$. If so, then return "yes." Otherwise, continue. By Theorem 5.1(3), this step is in Σ_3^p.
(2) We guess $\Delta \subseteq \mathcal{T}$ and make a call to a Π_3^p-oracle, to check whether $\mathcal{T} \setminus \Delta \in \mathsf{RCQ}^w(Q, D_m, V)$. If not, reject the current guess. Otherwise, return "yes."

Hence, the overall complexity of the algorithm is $\mathsf{NP}^{\Sigma_3^p}$ or Σ_4^p; therefore, $\mathsf{MINP}(\exists\mathrm{FO}^+)$ is in Π_4^p. The correctness of the algorithm is immediate.

(4) When $\mathcal{L}_Q$ is CQ. We show that $\mathsf{MINP}^w(\mathrm{CQ})$ is coDP-complete for ground instances. By Lemma 3.2, we assume without loss of generality that $\mathcal{T}$ is defined over a single relation schema. When $\mathcal{L}_Q$ 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 $\mathcal{T}$, $\mathcal{T}$ is a minimal instance in $\mathsf{RCQ}^w(Q, D_m, V)$ if and only if either $\mathcal{T} = \emptyset$ is in $\mathsf{RCQ}^w(Q, D_m, V)$, or when $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$, $|\mathcal{T}|$ is a singleton set, and $\mathsf{Mod}(\mathcal{T}, D_m, V) \neq \emptyset$.*

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

Suppose that $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$. Then, $\bigcap_{(\{t\}, D_m) \models V} Q(\{t\}) \neq \emptyset$. Once $\bigcap_{(\{t\}, D_m) \models V} Q(\{t\}) = \emptyset$, then $\bigcap_{\mathcal{I}' \in \mathsf{Ext}(\emptyset)} Q(\mathcal{I}') = \emptyset = Q(\emptyset)$; thus, $\emptyset \in \mathsf{RCQ}^w(Q, D_m, V)$, 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 τ be a tuple in $\mathcal{T}$ such that $\mathsf{Mod}(\tau, D_m, V) \neq \emptyset$. First, note that $\mathsf{Mod}(\tau, D_m, V)$ is a subset of $\{t \mid (\{t\}, D_m) \models V\}$ and, for each $t \in \mathsf{Mod}(\tau, D_m, V)$, we have that $Q(\{t\}) = u$. We show that $\{\tau\}$ is in $\mathsf{RCQ}^w(Q, D_m, V)$. We only need to show that the intersection of $Q(\{t, s\})$, where $t \in \mathsf{Mod}(\tau, D_m, V)$ and $(\{t, s\}, D_m) \models V$, is equal to u. In fact, we show a stronger result: $Q(\{t, s\}) = u$ for each t and s as before. Suppose, by contradiction, that there exists a valuation μ' 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 μ'. Observe, however, that also $(\{s\}, D_m) \models V$, 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 $\mathcal{T}$ such that either $\mathcal{T} = \emptyset$ or $|\mathcal{T}| = 1$. Furthermore, for $\mathcal{T}$ with $|\mathcal{T}| \leq 1$, the problem of testing minimality reduces to testing whether $\emptyset \in \mathsf{RCQ}^w(Q, D_m, V)$.

Lower bound. We show that, for ground instances, $\mathsf{MINP}^w(\mathrm{CQ})$ is coDP-hard by reduction from the complement of the SAT-UNSAT problem, which is DP-complete (see Papadimitriou [1994]). An instance of SAT-UNSAT is to determine whether, for a pair of 3SAT-instances (ϕ, ϕ'), ϕ is satisfiable and ϕ' is not satisfiable. Here, $\phi = C_1 \wedge \cdots \wedge C_r$ and $\phi' = C'_1 \wedge \cdots \wedge C'_s$, that is, for each $i \in [1, r]$ (resp. $i \in [1, s]$), clause C_i (resp. C'_i) is of the form $\ell^i_1 \vee \ell^i_2 \vee \ell^i_3$, where, for each $l \in [1, 3]$, ℓ^i_l 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 ϕ is not satisfiable or ϕ' 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 ($I_{(0,1)} = \{(0), (1)\}$, $I_\emptyset = \emptyset$).

(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 μ_X of the variables in C_i that makes C_i false, we add a selection condition of the form $\sigma_{\mu_X}(R) \subseteq I_\emptyset$, ensuring that the projection of R on $X_1, \ldots, X_n$ contains no tuples satisfying the selection condition. For example, for $C = \bar{x}_1 \vee \bar{x}_2 \vee x_3$ and $\mu_X = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$, we have the constraint $\sigma_{X_1=1 \wedge X_2=1 \wedge X_3=0}(R) \subseteq I_\emptyset$; 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 $C'_i = x_3 \vee x_4 \vee \bar{x}_5$ and $\mu_X = \{x_3 \mapsto 1, x_4 \mapsto 0, x_5 \mapsto 1\}$, we have the constraint $\sigma_{Y=1 \wedge X_3=1 \wedge X_4=0 \wedge X_5=1}(R) \subseteq I_\emptyset$; note that there are $r + s$ such CC's. Observe the following: (i) $\{t[Y] \mid (\{t\}, D_m) \models V\}$ is empty if and only if ϕ is not satisfiable; (ii) it is $\{(0)\}$ if ϕ is satisfiable and ϕ' is unsatisfiable; and (iii) it is $\{(0), (1)\}$ if both ϕ and ϕ' 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 $\mathsf{RCQ}^w(Q, D_m, V)$ if and only if ϕ is unsatisfiable or ϕ' is satisfiable. Note that $I = \emptyset$ is in $\mathsf{RCQ}^w(Q, D_m, V)$ if and only if $\bigcap_{I' \neq \emptyset, (I', D_m) \models V} Q(I') = \emptyset$. Since Q is monotonic, the latter is equivalent to $\bigcap_{(\{t\}, D_m) \models V} Q(\{t\}) = \emptyset$, which happens only when either $\{t[y] \mid (\{t\}, D_m) \models V\} = \emptyset$, meaning that ϕ is not satisfiable, or when $\{t[y] \mid (\{t\}, D_m) \models V\} \neq \emptyset$ and there exist two elements in this set. In this case, $Q(\{t_1\}) = \{t_1\} \neq \{t_2\} = Q(\{t_2\})$. Hence, the only case in which $I \notin \mathsf{RCQ}^w(Q, D_m, V)$ is when ϕ is satisfiable and ϕ' is not satisfiable.

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

(1) Check whether $|\mathcal{T}| > 1$. If so, return "no"; otherwise, continue.
(2) Check whether $\emptyset \in \mathsf{RCQ}^w(Q, D_m, V)$ and $\mathcal{T} = \emptyset$. If so, return "yes."
(3) Check whether $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$ and $|\mathcal{T}| = 1$. If so, return "yes"; otherwise, return "no."

It is in NP to check whether $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$. $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$ if and only if $\bigcap_{I' \in \mathsf{Ext}(\emptyset)} Q(I') \neq \emptyset$. One can verify that the latter holds if and only if $\bigcap_{(\{t\}, D_m) \models V} Q(\{t\}) \neq \emptyset$. As discussed before, we have that $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$ 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 coNP algorithm for its complement problem, that is, checking whether $\emptyset \in \mathsf{RCQ}^w(Q, D_m, V)$, 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 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 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 PTIME for one-tuple instances.

It is in coNP to check whether $\emptyset \in \mathsf{RCQ}^w(Q, D_m, V)$; thus, it is in NP to check whether $\emptyset \notin \mathsf{RCQ}^w(Q, D_m, V)$. Thus, the algorithm for deciding $\mathsf{MINP}^w(\mathrm{CQ})$ is in coDP $=$NP $\cup$coNP. □

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

6. VIABLE RELATIVE INFORMATION COMPLETENESS

We next investigate RCDP, RCQP, and MINP for viably complete c-instances, denoted by RCDP^v, RCQP^v, and 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 for 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 $\mathsf{RCQ}^v(Q, D_m, V)$ to represent the set of all viably complete instances.

In contrast to Theorem 4.1, $\mathsf{RCDP}^v(\mathrm{CQ})$ for c-instances is Σ_3^p-complete rather than Π_2^p-complete. Here, $\mathsf{RCDP}^v(\mathrm{FP})$ remains undecidable, as opposed to its counterpart in the weak model (Theorem 5.1).

THEOREM 6.1. *For c-instances,* $\mathsf{RCDP}^v(\mathcal{L}_Q)$ *is*

—*undecidable when* $\mathcal{L}_Q$ *is FO or FP, and*
—Σ_3^p*-complete when* $\mathcal{L}_Q$ *is CQ, UCQ, or* $\exists FO^+$.

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

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

COROLLARY 6.2. *For c-instances,* $\mathsf{RCQP}^v(\mathcal{L}_Q)$ *is*

—*undecidable when* $\mathcal{L}_Q$ *is FO or FP, and*
—NEXPTIME*-complete when* $\mathcal{L}_Q$ *is CQ, UCQ, or* $\exists FO^+$.

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

For c-instances, $\mathsf{MINP}^v(\mathcal{L}_Q)$ becomes Σ_3^p-complete for CQ, UCQ, or $\exists\mathrm{FO}^+$, rather than Π_3^p-complete, as in the strong model. The complexity bound is rather robust: it is the same for CQ, UCQ, and $\exists\mathrm{FO}^+$, as opposed to their counterparts in the weak model.

COROLLARY 6.3. $\mathsf{MINP}^v(\mathcal{L}_Q)$ *is*

—*undecidable for c-instances and for ground instances when* $\mathcal{L}_Q$ *is FO or FP, and*
—Σ_3^p*-complete for c-instances and* D_2^p*-complete for ground instances, when* $\mathcal{L}_Q$ *is CQ, UCQ or* $\exists FO^+$.

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 to 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. We often have a fixed query load, for example, 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, 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. The study of data complexity is nontrivial, however. 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 PTIME for CQ, UCQ, and $\exists$FO$^+$ when ground instances are considered, the 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,*

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

The relatively complete query problem. When we use INDs as CCs, that is, for CCs of the form $q \subseteq p$ when q and p are both projection queries, RCQP^s and RCQP^v become much simpler. The positive results hold even when the set V of CCs is *not* fixed. Moreover, RCDP^w is in constant time for FP for general CCs defined in CQ, by Theorem 5.4.

COROLLARY 7.2. *For fixed queries,*

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

The minimality problem. Similar to RCDP, we get tractable cases of 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,*

—MINP^s *and* MINP^v *are in* PTIME *for CQ, UCQ, and* $\exists$*FO*$^+$*; and*
—MINP^w *is in* 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: RCQP, RCDP, and 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 will 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 concerns the complexity of RCQP for FO in the weak model. We only know that it is undecidable for ground instances; 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

Serge Abiteboul and Oliver M. Duschka. 1998. Complexity of answering queries using materialized views. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*. ACM, New York, NY, 254–263. DOI:http://dx.doi.org/10.1145/275487.275516

Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley, New York, NY.

Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. 1991. On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78, 1, 159–187. DOI:http://dx.doi.org/10.1016/0304-3975(51)90007-2

Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'99)*. ACM, New York, NY, 68–79. DOI:http://dx.doi.org/10.1145/303976.303983

Marcelo Arenas, Jorge Pérez, Juan L. Reutter, and Cristian Riveros. 2009. Composition and inversion of schema mappings. *SIGMOD Record* 38, 3, 17–28. DOI:http://dx.doi.org/10.1145/1815933.1815938

Yang Cao, Ting Deng, Wenfei Fan, and Floris Geerts. 2014. On the data complexity of relative information completeness. *Information Systems* 45, 18–34. DOI:http://dx.doi.org/10.1016/j.is.2014.04.001

Jan Chomicki. 2007. Consistent query answering: Five easy pieces. In *Proceedings of the 11th International Conference on Database Theory (ICDT'07)*. Springer-Verlag, New York, NY, 1–17. DOI:http://dx.doi.org/10.1007/11965893_1

Charles Elkan. 1990. Independence of logic database queries and update. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, New York, NY, 154–160. DOI:http://dx.doi.org/10.1145/298514.298557

Wenfei Fan. 2008. Dependencies revisited for improving data quality. In *Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'08)*. ACM, New York, NY, 159–170. DOI:http://dx.doi.org/10.1145/1376916.1376940

Wenfei Fan and Floris Geerts. 2009. Relative information completeness. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'09)*. ACM, New York, NY, 97–106. DOI:http://dx.doi.org/10.1145/1559795.1559811

Wenfei Fan and Floris Geerts. 2010a. Capturing missing tuples and missing values. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'10)*. ACM, New York, NY, 169–178. DOI:http://dx.doi.org/10.1145/1807085.1807109

Wenfei Fan and Floris Geerts. 2010b. Relative information completeness. *ACM Transactions on Database Systems* 35, 4, Article 27, 44 pages. DOI:http://dx.doi.org/10.1145/1862919.1862924

Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, San Francisco, CA. DOI:http://dx.doi.org/10.2200/S00439ED1V01Y201207DTM030

Georg Gottlob and Roberto Zicari. 1988. Closed world databases opened through null values. In *Proceedings of the 14th International Conference on Very Large Data Bases (VLDB'88)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 50–61. http://dl.acm.org/citation.cfm?id=645915.671794

Gösta Grahne. 1991. *The Problem of Incomplete Information in Relational Databases*. Lecture Notes in Computer Science, Vol. 554. Springer. DOI:http://dx.doi.org/10.1007/3-540-54919-6

Tomasz Imieliński and Witold Lipski, Jr. 1984. Incomplete information in relational databases. *Journal of the ACM* 31, 4, 761–791. DOI:http://dx.doi.org/10.1145/1634.1886

Phokion G. Kolaitis. 2005. Schema mappings, data exchange, and metadata management. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'05)*. ACM, New York, NY, 61–75. DOI:http://dx.doi.org/10.1145/1065167.1065176

Willis Lang, Rimma V. Nehme, Eric Robinson, and Jeffrey F. Naughton. 2014. Partial results in database systems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*. ACM, New York, NY, 1275–1286. DOI:http://dx.doi.org/10.1145/2588555.2612176

Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'02)*. ACM, New York, NY, 233–246. DOI:http://dx.doi.org/10.1145/543613.543644

Alon Levy, Inderpal Singh Mumick, Yehoshua Sagiv, and Oded Shmueli. 1993. Equivalence, query-reachability and satisfiability in datalog extensions. In *Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'93)*. ACM, New York, NY, 109–122. DOI:http://dx.doi.org/10.1145/153850.153860

Alon Y. Levy. 1996. Obtaining complete answers from incomplete databases. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 402–412. http://dl.acm.org/citation.cfm?id=645922.673332

Alon Y. Levy and Yehoshua Sagiv. 1993. Queries independent of updates. In *Proceedings of the 19th International Conference on Very Large Data Bases (VLDB'93)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 171–181. http://dl.acm.org/citation.cfm?id=645919.672674

Leonid Libkin. 2014. Incomplete data: What went wrong, and how to fix it. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'14)*. ACM, New York, NY, 1–13. DOI:http://dx.doi.org/10.1145/2594538.2594561

David Loshin. 2008. *Master Data Management*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Microsoft. 2008. SQL Server 2008 R2 Master Data Services. Retrieved April 3, 2016 from http://www.microsoft.com/sqlserver/2008/en/us/MDS.aspx.

Donald W. Miller Jr., John D. Yeast, and Robin L. Evans. 2005. Missing prenatal records at a birth center: A communication problem quantified. In *AMIA Annual Symposium Proceedings (AMIA'05)*. 535–539.

Amihai Motro. 1989. Integrity = validity + completeness. *ACM Transactions on Database Systems* 14, 4, 480–502. DOI:http://dx.doi.org/10.1145/76902.76904

Dan Olteanu, Christoph Koch, and Lyublena Antova. 2008. World-set decompositions: Expressiveness and efficient algorithms. *Theoretical Computer Science* 403, 2–3, 265–284.

Christos H. Papadimitriou. 1994. *Computational Complexity*. Addison-Wesley, New York, NY.

John Radcliffe and Andrew White. 2008. Key issues for Master Data Management. Gartner. Retrieved April 3, 2016 from https://www.gartner.com/doc/590507/key-issues-master-data-management.

Luc Segoufin and Victor Vianu. 2005. Views and queries: Determinacy and rewriting. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'05)*. ACM, New York, NY, 49–60. DOI:http://dx.doi.org/10.1145/1065167.1065174

Mark Spielmann. 2000. *Abstract State Machines: Verification Problems and Complexity*. Ph.D. Dissertation. RWTH Aachen University, Aachen, Germany.

Boris Trakhtenbrot. 1950. The impossibility of an algorithm for the decidability problem on finite classes. *Doklady Akademii Nauk SSSR* 70, 4, 569–572.

Ron van der Meyden. 1998. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, J. Chomicki and G. Saake (Eds.). Kluwer, 307–356.

Moshe Vardi. 1986. On the integrity of databases with incomplete information. In *Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS'86)*. ACM, New York, NY, 252–266. DOI: http://dx.doi.org/10.1145/6012.15419

Michael Wooldridge and Paul E. Dunne. 2004. On the computational complexity of qualitative coalitional games. *Artificial Intelligence* 158, 1, 27–73. DOI: http://dx.doi.org/10.1016/j.artint.2004.04.002

Received March 2015; revised October 2015; accepted December 2015