

# Relative Information Completeness

Wenfei Fan

University of Edinburgh and Harbin Institute of Technology

and

Floris Geerts

University of Edinburgh

---

The paper investigates the question of whether a partially closed database has complete information to answer a query. In practice an enterprise often maintains master data  $D_m$ , a closed-world database. We say that a database  $D$  is *partially closed* if it satisfies a set  $V$  of containment constraints of the form  $q(D) \subseteq p(D_m)$ , where  $q$  is a query in a language  $\mathcal{L}_C$  and  $p$  is a projection query. The part of  $D$  not constrained by  $(D_m, V)$  is open, from which some tuples may be missing. The database  $D$  is said to be *complete* for a query  $Q$  relative to  $(D_m, V)$  if for all partially closed extensions  $D'$  of  $D$ ,  $Q(D') = Q(D)$ , *i.e.*, adding tuples to  $D$  either violates some constraints in  $V$  or does not change the answer to  $Q$ .

We first show that the proposed model can also capture the consistency of data, in addition to its relative completeness. Indeed, integrity constraints studied for data consistency can be expressed as containment constraints. We then study two problems. One is to decide, given  $D_m$ ,  $V$ , a query  $Q$  in a language  $\mathcal{L}_Q$  and a partially closed database  $D$ , whether  $D$  is complete for  $Q$  relative to  $(D_m, V)$ . The other is to determine, given  $D_m$ ,  $V$  and  $Q$ , whether there exists a partially closed database that is complete for  $Q$  relative to  $(D_m, V)$ . We establish matching lower and upper bounds on these problems for a variety of languages  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ . We also provide characterizations for a database to be relatively complete, and for a query to allow a relatively complete database, when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are conjunctive queries.

Categories and Subject Descriptors: H.1.1 [Information Systems]: Models and Principles—*Value of information*; H.2.1 [Database Management]: Logical Design—*Data models*

General Terms: Design, Languages, Reliability, Theory

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

---

## 1. INTRODUCTION

One of the issues central to data quality concerns incomplete information. Given a database  $D$  and a query  $Q$ , we want to know whether  $Q$  can be answered by using the data in  $D$ . If the information in  $D$  is incomplete, one can hardly expect its answer to  $Q$  to be accurate. Incomplete information introduces serious prob-

---

Authors' addresses: School of Informatics, Laboratory for Foundations of Computer Science, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB, Scotland, UK; email: {wenfei, fgeerts}@inf.ed.ac.uk.

Support: Wenfei Fan and Floris Geerts are supported in part by EPSRC EP/E029213/1.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20YY ACM 0362-5915/20YY/0300-0001 \$5.00

lems to enterprises: it routinely leads to misleading analytical results and biased decisions, and accounts for loss of revenues, credibility and customers. Worse still, records missing from, *e.g.*, financial, medical and administrative databases, may have disastrous consequences [Herzog et al. 2007].

The study of incomplete information is almost as old as the relational model itself. There has been a host of work on *missing values* in tuples, under the Closed World Assumption (CWA). That is, all the tuples representing real-world entities are assumed already in place, but the values of some fields in these tuples are missing. As a result, facts that are not in the database are assumed to be false. To this end, a number of approaches have been proposed, notably representation systems for a set of possible worlds (*e.g.*, *v*-tables, *c*-tables, OR-object databases; see *e.g.*, [Grahne 1991; Imieliński and Lipski 1984]) and disjunctive logic programming (see [van der Meyden 1998] for a comprehensive survey).

Equally important to data quality is how to handle *missing tuples*, under the Open World Assumption (OWA). That is, a database may only be a proper subset of the set of tuples that represent real-world entities. While there has also been work on missing tuples (*e.g.*, [Gottlob and Zicari 1988; Levy 1996; Motro 1989; Vardi 1986]), this issue has received relatively less attention. Under OWA, one can often expect few sensible queries to find complete answers.

In several emerging applications, however, neither CWA nor OWA is quite appropriate. This is evident in, *e.g.*, Master Data Management (MDM [Dreibelbis et al. 2007; Radcliffe and White 2008; Loshin 2008]), one of the fastest growing software markets. An enterprise nowadays typically maintains *master data* (*a.k.a. reference data*), a single repository of high-quality data that provides various applications with a synchronized, consistent view of the core business entities of the enterprise. The master data contains complete information about the enterprise in certain categories, *e.g.*, employees, departments, projects, and equipment.

Master data can be regarded as a closed-world database. Meanwhile a number of other databases may be in use in the enterprise for, *e.g.*, sales, project control and customer support. On one hand, these databases may not be complete, *e.g.*, some sale transactions may be missing. On the other hand, certain parts of the databases are *constrained by* the master data, *e.g.*, employees and projects. In other words, these databases are neither entirely closed-world, nor entirely open-world. It becomes more interesting to decide whether the information available in these databases is complete to answer a query.

**Example 1.1:** Consider a company that maintains  $\text{DCust}(\text{cid}, \text{name}, \text{ac}, \text{phn})$ , a master data relation consisting of all its domestic customers, in which a tuple  $(c, n, a, p)$  specifies the id  $c$ , name  $n$ , area code  $a$  and phone number  $p$  of a customer. In addition, the company also has databases (a)  $\text{Cust}(\text{cid}, \text{name}, \text{cc}, \text{ac}, \text{phn})$  of all customers of the company, domestic (with country code  $\text{cc} = 01$ ) or international; and (b)  $\text{Supt}(\text{eid}, \text{dept}, \text{cid})$ , indicating that employee  $\text{eid}$  in  $\text{dept}$  supports customer  $\text{cid}$ . Neither  $\text{Cust}$  nor  $\text{Supt}$  is part of the master data.

Consider a query  $Q_1$  posed on  $\text{Supt}$  to find all the customers in NJ with  $\text{ac} = 908$  who are supported by the employee with  $\text{eid} = e_0$ . The query may *not* get a complete answer since some tuples may be missing from  $\text{Supt}$ . However, if  $Q_1$  returns all NJ customers with  $\text{ac} = 908$  found in the master relation  $\text{DCust}$ , then we can safely

conclude that query  $Q_1$  can find a complete answer from  $\text{Supt}$ . That is, there is no need to add more tuples to  $\text{Supt}$  in order to answer  $Q_1$ .

Now consider a query  $Q_2$  to find *all* customers supported by the employee  $e_0$ . Note that the international customers of  $\text{Cust}$  are not constrained by master data; in other words, the company may not maintain a complete list of its customers. As a result, we are not able to tell whether any  $\text{Supt}$  tuples in connection with  $e_0$  are missing. Worse still, we do not even know what tuples should be added to  $\text{Supt}$  such that the answer to  $Q_2$  in  $\text{Supt}$  is complete.

Not all is lost. If we know that  $\text{eid} \rightarrow \text{dept}, \text{cid}$  is a functional dependency (FD) on  $\text{Supt}$ , then we can also conclude that the answer to  $Q_2$  in  $\text{Supt}$  is complete as long as it is nonempty. More generally, suppose that there is a constraint which asserts that an employee supports at most  $k$  customers. Then if the answer to  $Q_2$  in  $\text{Supt}$  returns  $k$  customers, we know that the seemingly incomplete relation  $\text{Supt}$  is actually complete for  $Q_2$ . That is, adding more tuples to  $\text{Supt}$  does not change the answer to  $Q_2$  in  $\text{Supt}$ . Even when  $Q_2$  returns  $k'$  tuples, where  $k' < k$ , we know that we need to add at most  $k - k'$  tuples to  $\text{Supt}$  to make it complete for  $Q_2$ .

As another example, consider a master relation  $\text{Manage}_m(\text{eid}_1, \text{eid}_2)$ , which indicates that employee  $\text{eid}_2$  directly reports to  $\text{eid}_1$ . Suppose that  $\text{Manage}(\text{eid}_1, \text{eid}_2)$  is a relation that is not part of master data, but it contains all tuples in  $\text{Manage}_m$ . Consider query  $Q_3$  on  $\text{Manage}$  to find all the people above  $e_0$  in the management hierarchy, *i.e.*, the people to whom  $e_0$  reports, directly or indirectly. Note that if  $Q_3$  is in, *e.g.*, datalog, then we may expect the answer to  $Q_3$  to be complete. In contrast, if  $Q_3$  is a conjunctive query, then the answer to  $Q_3$  is incomplete unless  $\text{Manage}$  contains the transitive closure of  $\text{Manage}_m$ . In the latter case, the seemingly complete  $\text{Manage}$  relation turns out to be “incomplete”. This tells us that the completeness of information is also relative to the query language in use.  $\square$

Several natural questions have to be answered. Given a query  $Q$  posed on a database  $D$  that is partially constrained by master data  $D_m$ , can we find complete information from  $D$  to answer  $Q$ ? Does there exist a database  $D$  at all that is partially constrained by  $D_m$  and has the complete information to answer  $Q$  (*i.e.*, is there a finite  $D$  such that adding tuples to  $D$  will not change the answer to  $Q$  in  $D$ )? These questions are not only of theoretical interest, but are also important in practice. Indeed, the ability to answer these questions not only helps us determine whether a query can find a complete answer from a particular database, but also provides a guidance for what data should be collected in order to answer a query. The increasing demand for MDM highlights the need for a full treatment of the completeness of information *relative to* master data and user queries.

**Relative completeness.** In response to the need, we propose a notion of relative information completeness. To characterize databases  $D$  that are partially constrained by master data  $D_m$ , we specify a set  $V$  of *containment constraints*. A containment constraint is of the form  $q(D) \subseteq p(D_m)$ , where  $q$  is a query in a language  $\mathcal{L}_C$  posed on  $D$ , and  $p$  is a simple projection query on  $D_m$ . Intuitively, the part of  $D$  that is constrained by  $V$  is bounded by  $D_m$ , while the rest is open-world.

We refer to a database  $D$  that satisfies  $V$  as a *partially closed* database *w.r.t.*  $(D_m, V)$ . A database  $D'$  is referred to as a *partially closed extension* of  $D$  if  $D \subseteq D'$

and  $D'$  is partially closed *w.r.t.*  $(D_m, V)$  itself.

For a query  $Q$  in a language  $\mathcal{L}_Q$ , a partially closed database  $D$  is said to be complete *w.r.t.*  $(D_m, V)$  if for all partially closed extensions  $D'$  of  $D$  *w.r.t.*  $(D_m, V)$ ,  $Q(D') = Q(D)$ . That is, there is no need for adding new tuples to  $D$ , since they either violate the containment constraints, or do not change the answer to  $Q$ . In other words,  $D$  already contains complete information necessary for answering  $Q$ .

To simplify the discussion, we focus on missing tuples in this paper. As will be addressed in Section 5, the notion of relatively complete information can be extended to accommodate missing values as well, by capitalizing on representation systems for possible worlds [Grahne 1991; Imieliński and Lipski 1984].

**Completeness and consistency.** Another critical issue to data quality is the consistency of the data. To answer a query using a database  $D$ , one naturally wants the information in  $D$  to be both complete and consistent.

To capture inconsistencies one typically use integrity constraints (*e.g.*, [Arenas et al. 1999; Bravo et al. 2007; Cali et al. 2003; Fan et al. 2008], see [Chomicki 2007; Fan 2008] for recent surveys). That is, conflicts and errors in the data are detected as violations of the constraints. In light of this, one might be tempted to extend the notion of partially closed databases by incorporating integrity constraints.

The good news is that there is no need to overburden the notion with a set of integrity constraints. We show that constraints studied for ensuring data consistency, such as denial constraints [Arenas et al. 1999], conditional functional dependencies [Fan et al. 2008] and conditional inclusion dependencies [Bravo et al. 2007], are expressible as simple containment constraints. As a result, we can assure that only consistent and partially closed databases are considered, by enforcing containment constraints. That is, in a uniform framework we can deal with both relative information completeness and data consistency.

**Main results.** We investigate two important decision problems associated with the relative completeness of information, and establish their complexity bounds. We also provide characterizations for a database to be relatively complete and for a query to allow a relatively complete database, in certain cases when the decision problems are decidable.

*Determining relatively complete databases.* One of the two problems, referred to as the *relatively complete database problem*, is to determine, given a query  $Q$ , master data  $D_m$ , a set  $V$  of containment constraints, and a partially closed database  $D$  *w.r.t.*  $(D_m, V)$ , whether or not  $D$  is complete for  $Q$  relatively to  $(D_m, V)$ . That is to decide, when  $Q$  is posed on  $D$ , whether the answer of  $D$  to  $Q$  is complete.

We parameterize the problem with various  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ , the query languages in which the queries are expressed and in which the containment constraints are defined, respectively. We consider the following  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ , all with equality ‘=’ and inequality ‘≠’:

- conjunctive queries (CQ),
- union of conjunctive queries (UCQ),
- positive existential FO queries ( $\exists\text{FO}^+$ ),
- first-order queries (FO), and
- datalog (FP).

We establish lower and upper bounds for the problem *w.r.t.* all these languages, *all matching*, either  $\Pi_2^P$ -complete or undecidable. The complexity bounds are rather robust: the lower bounds remain intact even when  $D_m$  and  $V$  are predefined and fixed. The problem is already  $\Pi_2^P$ -complete for CQ queries and containment constraints defined as inclusion dependencies (INDs), when  $D_m$  and  $V$  are fixed.

Determining relatively complete queries. The other problem, referred to as the *relatively complete query problem*, is to determine, given  $Q$ ,  $D_m$  and  $V$ , whether there exists a partially closed database  $D$  that is complete for  $Q$  relative to  $(D_m, V)$ , *i.e.*, adding more tuples to  $D$  would not change answers to  $Q$  in  $D$ . It is to decide for  $Q$  whether it is possible to find a relatively complete database  $D$  at all. If such a  $D$  exists,  $Q$  is said to be a *query relatively complete w.r.t.  $(D_m, V)$* .

We present complexity bounds for the problem when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  range over CQ, UCQ,  $\exists\text{FO}^+$ , FO and FP. The lower and upper bounds are again all matching: conNP-complete, NEXPTIME-complete, or undecidable. In contrast to its counterpart for relative complete databases, fixed  $D_m$  and  $V$  make our lives easier: the problem becomes  $\Sigma_3^P$ -complete as opposed to NEXPTIME-complete in certain cases.

Characterizations. When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ, we present sufficient and necessary conditions for (a) a partially closed database  $D$  to be complete for a query  $Q$  relative to  $(D_m, V)$ , and (b) a query to be relatively complete *w.r.t.  $(D_m, V)$* . As remarked earlier, the characterizations tell us what data should be collected in  $D$  in order to answer a query, and whether a query can find a complete answer at all. The characterizations can be extended to UCQ and  $\exists\text{FO}^+$ .

To the best of our knowledge, this work is among the first efforts to study the completeness of information in emerging applications such as MDM. Our results provide a comprehensive picture of complexity bounds for important problems associated with relatively complete information, and moreover, guidance for how to make a database relatively complete. A variety of techniques are used to prove the results, including a wide range of reductions and constructive proofs with algorithms.

**Related work.** Several approaches have been proposed to represent or query databases with missing tuples. In [Vardi 1986], a complete and consistent extension of an incomplete database  $D$  is defined to be a database  $D_c$  such that  $D \subseteq \pi_L(D_c)$  and  $D_c \models \Sigma$ , where  $\pi$  is the projection operator,  $L$  is the set of attributes in  $D$ , and  $\Sigma$  is a set of integrity constraints. Complexity bounds for computing the set of complete and consistent extensions of  $D$  *w.r.t.  $\Sigma$*  are established there. A notion of *open null* is introduced in [Gottlob and Zicari 1988] to model locally controlled open-world databases: parts of a database  $D$ , values or tuples, can be marked with open null and are assumed open-world, while the rest is closed. Relational operators are extended to tables with open null. In contrast to [Gottlob and Zicari 1988], this work aims to model databases partially constrained by master data  $D_m$  and consistency specifications, both via containment constraints. In addition, we study decision problems that are not considered in [Gottlob and Zicari 1988].

Partially complete databases  $D$  have also been studied in [Motro 1989], which assumes a virtual database  $D_c$  with “complete information”, and assumes that part of  $D$  is known as a view of  $D_c$ . It investigates the query answer completeness problem, the problem for determining whether a query posed on  $D_c$  can be answered

by an equivalent query on  $D$ . In this setting, the problem can be reduced to query answering using views. Along the same lines, [Levy 1996] assumes that  $D$  contains some CQ views of  $D_c$ . It reduces the query answer completeness problem to the independence problem for deciding the independence of queries from updates [Levy and Sagiv 1993]. As opposed to [Levy 1996; Motro 1989], we assume neither  $D_c$  with complete information nor that an incomplete database  $D$  contains some views of  $D_c$ . Instead, we consider  $D_m$  as an “upper bound” of certain information in  $D$ . Moreover, the decision problems studied here can be reduced to neither the query answering problem nor the independence problem (see below).

There has also been work on modeling negative information via logic programming (see [van der Meyden 1998]), which considers neither partially complete databases nor the decision problems studied in this work.

We now clarify the difference between our decision problems and the independence problem (*e.g.*, [Elkan 1990; Levy and Sagiv 1993]). The latter is to determine whether a query  $Q$  is independent of updates generated by another query  $Q^u$ , such that *for all* databases  $D$ ,  $Q(D) = Q(D \oplus \Delta)$ , where  $\Delta$  denotes updates generated by  $Q^u$ . In contrast, we consider relatively complete queries  $Q$ , such that *there exists* a database  $D$  complete for  $Q$  relative to master data  $D_m$  and containment constraints  $V$ , where  $D$  and  $D_m$  satisfy  $V$ . We want to decide (a) whether for a query  $Q$  *there exists* a relatively complete database  $D$ , and (b) whether a given  $D$  that satisfies  $V$  is a witness for  $Q$  to be relatively complete. Due to the difference between the problems, results for the independence problem do not straightforwardly carry over to ours, and vice versa.

One may think of an incomplete database as a “view” of a database with complete information. There has been a large body of work on answering queries using views (*e.g.*, [Abiteboul and Duschka 1998; Calvanese et al. 2007; Li 2003; Segoufin and Vianu 2005]), to determine certain answers [Abiteboul and Duschka 1998], compute complete answers from views with limited access patterns [Deutsch et al. 2007; Li 2003], or to decide whether views determine queries [Segoufin and Vianu 2005] or are lossless [Calvanese et al. 2007]. This work differs from that line of research in that one may not find a view definable in a query language to characterize a relatively complete database  $D$  in terms of the database with complete information. Indeed,  $D$  is only *partially* constrained by master data  $D_m$ , while  $D_m$  itself may not contain the complete information that  $D$  intends to represent.

There has also been recent work on consistent query answering (*e.g.*, [Arenas et al. 1999; Cali et al. 2003; Chomicki 2007]). That is to decide whether a tuple is in the answer to a query in every repair of a database  $D$ , where a repair is a database that satisfies a given set of integrity constraints and moreover, minimally differs from the original  $D$  *w.r.t.* some repair model. Master data  $D_m$  is not considered there, and we do not consider repairs in this paper. Note that most containment constraints are *not* expressible as integrity constraints studied for consistency.

This paper is an extension of [Fan and Geerts 2009].

**Organization.** In Section 2 we define relatively complete databases and queries, state the decision problems, and show that integrity constraints for capturing inconsistencies can be expressed as containment constraints. We provide complexity bounds and characterizations for determining relatively complete databases in Sec-

tion 3, and for deciding relatively complete queries in Section 4. Section 5 summarizes the main results of the paper and identifies open problems. We refer some proofs to the electronic appendix.

## 2. RELATIVELY COMPLETE DATABASES AND QUERIES

We first present the notion of relative completeness of data, and then show that the consistency of the data can be characterized in the uniform framework. Finally, we demonstrate the benefits of master data and the usage of relative completeness in assessing the quality of data and the quality of query answers.

### 2.1 Relative Completeness

We start with specifications of databases and master data.

**Databases and master data.** A database is specified by a relational schema  $\mathcal{R}$ , which consists of a collection of relation schemas  $(R_1, \dots, R_n)$ . Each schema  $R_i$  is defined over a fixed set of attributes. For each attribute  $A$ , its domain is specified in  $R$ , denoted by  $dom(A)$ . To simplify the discussion we consider two domains: a countably infinite set  $\mathbf{d}$  and a finite set  $\mathbf{d}_f$  with at least two elements. We assume that  $dom(A)$  is either infinite ( $\mathbf{d}$ ) or finite ( $\mathbf{d}_f$ ).

We say that an instance  $D = (I_1, \dots, I_n)$  of  $\mathcal{R}$  is *contained in* another instance  $D' = (I'_1, \dots, I'_n)$  of  $\mathcal{R}$ , denoted by  $D \subseteq D'$ , if  $I_j \subseteq I'_j$  for all  $j \in [1, n]$ . If  $D \subseteq D'$  then we also say that  $D'$  is an *extension* of  $D$ .

Master data (*a.k.a.* reference data) is a closed-world database  $D_m$ , specified by a relational schema  $\mathcal{R}_m$ . As remarked earlier, an enterprise typically maintains master data that is assumed consistent and complete about certain information of the enterprise [Dreibelbis et al. 2007; Radcliffe and White 2008; Loshin 2008]. We do not impose any restriction on the relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ .

**Containment constraints.** Let  $\mathcal{L}_C$  be a query language. A *containment constraint* (CC)  $\phi_v$  in  $\mathcal{L}_C$  is of the form  $q_v(\mathcal{R}) \subseteq p(\mathcal{R}_m)$ , where  $q_v$  is a query in  $\mathcal{L}_C$  defined over schema  $\mathcal{R}$ , and  $p$  is a projection query over schema  $\mathcal{R}_m$ . That is,  $p$  is query of the form  $\exists \bar{x} R_i^m(\bar{x}, \bar{y})$  for some relation  $R_i^m$  in  $\mathcal{R}_m$ .

An instance  $D$  of  $\mathcal{R}$  and master data instance  $D_m$  of  $\mathcal{R}_m$  *satisfy*  $\phi_v$ , denoted by  $(D, D_m) \models \phi_v$ , if  $q_v(D) \subseteq p(D_m)$ .

We say that  $D$  and  $D_m$  *satisfy* a set  $V$  of CCs, denoted by  $(D, D_m) \models V$ , if for each  $\phi_v \in V$ ,  $(D, D_m) \models \phi_v$ .

Intuitively,  $\phi_v$  assures that  $D_m$  is an “upper bound” of the information extracted by  $q_v(D)$ . In other words, CWA is asserted for  $D_m$  that constrains the part of data identified by  $q_v(D)$  from  $D$ . That is, while this part of  $D$  can be extended, the expansion cannot go beyond the information already in  $D_m$ . On the other hand, OWA is assumed for the part of  $D$  that is not constrained by  $\phi_v$ .

We write  $q_v(\mathcal{R}) \subseteq p(\mathcal{R}_m)$  as  $q_v \subseteq p$  when  $\mathcal{R}$  and  $\mathcal{R}_m$  are clear from the context. We write  $q_v \subseteq p$  as  $q_v \subseteq \emptyset$  if  $p$  is a projection on an empty master relation.

**Example 2.1:** Recall Cust, Supt and DCust from Example 1.1. We can write a CC  $\phi_0 = q(\text{Cust}, \text{Supt}) \subseteq \pi_{\text{cid}}(\text{DCust})$  in the language of conjunctive queries, where

$$q(e) = \exists n, cc, a, p, e, d (\text{Cust}(e, n, cc, a, p) \wedge \text{Supt}(e, d, e) \wedge cc = 01),$$

asserting that all domestic customers are constrained by master relation DCust.

Here  $\pi_{\text{cid}}(\text{DCust})$  denotes the projection of DCust on the cid attribute.

Another CC  $\phi_1$  in the language of conjunctive queries is  $q \subseteq \emptyset$ , where

$$q(e) = \exists c_1, d_1, \dots, c_{k+1}, d_{k+1} \left( \bigwedge_{i \in [1, k+1]} \text{Supt}(e, d_i, c_i) \wedge \bigwedge_{i, j \in [1, k+1], i \neq j} (c_i \neq c_j) \right).$$

It asserts that each employee supports at most  $k$  customers.  $\square$

A database  $D$  is called a *partially closed* database *w.r.t.*  $(D_m, V)$  if  $(D, D_m) \models V$ .

Observe that a CC  $q_v(\mathcal{R}) \subseteq p(\mathcal{R}_m)$  is an *inclusion dependency* (IND) when  $q_v$  is also a projection query. In the sequel we simply refer to such CCs as INDs.

**Relative completeness.** Let  $\mathcal{L}_Q$  be a query language, not necessarily the same as  $\mathcal{L}_C$ . Let  $Q$  be a query in  $\mathcal{L}_Q$ .

Consider a partially closed database  $D$  *w.r.t.* master data  $D_m$  and a set  $V$  of CCs. We say that  $D$  is *complete for query  $Q$  relative to  $(D_m, V)$*  if for all instances  $D'$  of  $\mathcal{R}$ , if  $D \subseteq D'$  and  $(D', D_m) \models V$ , then  $Q(D) = Q(D')$ .

That is,  $D$  is complete for  $Q$  relative to  $(D_m, V)$  if (a)  $D$  is partially closed *w.r.t.*  $(D_m, V)$ , and (b) for each partially closed extension  $D'$  of  $D$ ,  $Q(D) = Q(D')$ . In other words, no matter how  $D$  is expanded by including new tuples, as long as the extension does not violate  $V$ , the answer to query  $Q$  remains unchanged. Intuitively,  $D$  has complete information for answering  $Q$ . The notion is *relative to* the master data  $D_m$  and CCs in  $V$ : the extensions of  $D$  should not violate the CCs in  $V$ , *i.e.*,  $(D', D_m) \models V$ . That is, CWA for  $D_m$  is observed.

Given  $D_m, V$  and a query  $Q$  in  $\mathcal{L}_Q$ , we define *the set of complete databases for  $Q$  w.r.t.  $(D_m, V)$* , denoted by  $\text{RCQ}(Q, D_m, V)$ , to be the set of all complete databases for  $Q$  relative to  $(D_m, V)$ .

When  $\text{RCQ}(Q, D_m, V)$  is nonempty,  $Q$  is called a *relatively complete query w.r.t.  $(D_m, V)$* . Intuitively,  $Q$  is relatively complete if it is possible to find a database  $D$  such that the answer to  $Q$  in  $D$  is complete.

**Example 2.2:** As described in Example 1.1, relations Cust and Supt are complete for query  $Q_1$  *w.r.t.* DCust and the CC  $\phi_0$  of Example 2.1, provided that the query result contains all domestic customers in DCust. In this case, the database consisting of the Cust and Supt relations is in  $\text{RCQ}(Q_1, \text{DCust}, \{\phi_0\})$ , and hence,  $Q_1$  is relatively complete *w.r.t.*  $(\text{DCust}, \{\phi_0\})$ .

When the CC  $\phi_1$  of Example 2.1 is in place, Supt is already complete for query  $Q_2$  of Example 1.1 *w.r.t.*  $(\emptyset, \{\phi_1\})$ , as soon as the query result is nonempty. Hence,  $Q_2$  is complete *w.r.t.*  $(\emptyset, \{\phi_1\})$ .

On the other hand, relation Manage of Example 1.1 is not complete for the CQ query  $Q_3$ . However,  $Q_3$  is relatively complete *w.r.t.*  $\text{Manage}_m$ : one can make Manage complete for  $Q_3$  by including the transitive closure of  $\text{Manage}_m$ .  $\square$

**Decision problems.** We study two decision problems. One is the *relatively complete database problem* for  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ , denoted by  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  and stated as:

PROBLEM:  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$   
 INPUT: A query  $Q \in \mathcal{L}_Q$ , master data  $D_m$ , a set  $V$  of CCs in  $\mathcal{L}_C$ , and a partially closed database  $D$  *w.r.t.*  $(D_m, V)$ .  
 QUESTION: Is  $D$  in  $\text{RCQ}(Q, D_m, V)$ ? That is, is  $D$  complete for  $Q$  relative to  $(D_m, V)$ ?

The other one is the *relatively complete query problem* for  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ , denoted by  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  and stated as follows:

PROBLEM:	$\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$
INPUT:	A query $Q \in \mathcal{L}_Q$ , master data $D_m$ , and a set $V$ of CCs in $\mathcal{L}_C$ .
QUESTION:	Is $\text{RCQ}(Q, D_m, V)$ nonempty? That is, does there exist a database that is complete for $Q$ relative to $(D_m, V)$ ?

Intuitively, RCDP is to decide whether a particular database has complete information to answer a query, and RCQP is to decide whether there exists a database at all that is relatively complete for a query.

**Query languages.** We consider  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  ranging over:

- (a) conjunctive queries (CQ), built up from atomic formulas with constants and variables, *i.e.*, relation atoms in database schema  $\mathcal{R}$ , equality (=) and inequality ( $\neq$ ), by closing under conjunction  $\wedge$  and existential quantification  $\exists$ ;
- (b) union of conjunctive queries (UCQ) of the form  $Q_1 \cup \dots \cup Q_k$ , where for each  $i \in [1, k]$ ,  $Q_i$  is in CQ;
- (c) positive existential FO queries ( $\exists\text{FO}^+$ ), built from atomic formulas by closing under  $\wedge$ , *disjunction*  $\vee$  and  $\exists$ ;
- (d) first-order logic queries (FO) built from atomic formulas using  $\wedge$ ,  $\vee$ , *negation*  $\neg$ ,  $\exists$  and universal quantification  $\forall$ ; and
- (f) datalog queries (FP), defined as a collection of rules  $p(\bar{x}) \leftarrow p_1(\bar{x}_1), \dots, p_n(\bar{x}_n)$ , where each  $p_i$  is either an atomic formula (a relation atom in  $\mathcal{R}$ , =,  $\neq$ ) or an IDB predicate. That is, FP is an extension of  $\exists\text{FO}^+$  with an inflational fixpoint operator. See *e.g.*, [Abiteboul et al. 1995] about the details of these languages.

## 2.2 Relative Completeness and Consistency

Real life data often contains errors and conflicts (see *e.g.*, [Batini and Scannapieco 2006]). To capture inconsistencies in the data, it is typical to use integrity constraints. That is, a set  $\Sigma$  of integrity constraints is imposed on a database  $D$  such that errors in  $D$  are detected as violations of one or more constraints in  $\Sigma$ .

Several classes of integrity constraints have been proposed for capturing inconsistencies in relational data (see, *e.g.*, [Chomicki 2007; Fan 2008] for recent surveys). Below we review three classes recently studied for the consistency of data.

- (a) Denial constraints [Arenas et al. 1999; Chomicki 2007] are universally quantified FO sentences of the form:

$$\varphi_d = \forall \bar{x}_1 \dots \bar{x}_k \neg (R_1(\bar{x}_1) \wedge \dots \wedge R_k(\bar{x}_k) \wedge \varphi(\bar{x}_1, \dots, \bar{x}_k)),$$

where  $R_i$  is a relation atom for  $i \in [1, k]$ , and  $\varphi$  is a conjunction of built-in predicates = and  $\neq$ .

- (b) Conditional functional dependencies (CFDs) [Fan et al. 2008] are an extension

of functional dependencies (FDs) of the form:

$$\begin{aligned} \varphi_{\text{cfd}} = \forall \bar{x}_1 \bar{x}_2 \bar{y}_1 \bar{y}_2 \bar{z}_1 \bar{z}_2 ( & R(\bar{x}_1, \bar{z}_1, \bar{y}_1) \wedge R(\bar{x}_2, \bar{z}_2, \bar{y}_2) \wedge \phi(\bar{x}_1) \wedge \phi(\bar{x}_2) \wedge \bar{x}_1 = \bar{x}_2 \\ & \rightarrow \bar{y}_1 = \bar{y}_2 \wedge \psi(\bar{y}_1) \wedge \psi(\bar{y}_2)) \end{aligned}$$

where  $R$  is a relation atom, and  $\phi(\bar{x})$  is a conjunction of the form  $x_{i_1} = c_1 \wedge \dots \wedge x_{i_k} = c_k$ ; here  $\{x_{i_j} \mid j \in [1, k]\}$  is a subset of  $\bar{x}$ , and  $c_j$  is a constant. The expression  $\psi(\bar{y})$  is defined similarly.

Intuitively, a CFD extends a traditional FD  $X \rightarrow Y$  by incorporating patterns of semantically related constants, where  $X$  and  $Y$  are the attributes denoted by  $\bar{x}$  and  $\bar{y}$  in  $R(\bar{x}, \bar{z}, \bar{y})$  above, respectively. That is, for each  $R$  tuple  $t_1$  and  $t_2$ , if  $t_1[X] = t_2[X]$  and in addition,  $t_1[X]$  and  $t_2[X]$  have a certain constant pattern specified by  $\phi(\bar{x})$ , then  $t_1[Y] = t_2[Y]$  and moreover,  $t_1[Y]$  and  $t_2[Y]$  have the constant pattern specified by  $\psi(\bar{y})$ . Note that in the absence of  $\phi(\bar{x})$  and  $\psi(\bar{y})$ , the CFD is a traditional FD.

As an example, suppose that each employee in the BU department supports at most one customer at a time. This can be expressed as a CFD on the `Supt` relation of Example 1.1: `dept = "BU", eid  $\rightarrow$  cid`. This asserts that `eid` is a key of those `Supt` tuples in connection with employees in BU, rather than a key of entire relation.

(c) Conditional inclusion dependencies (CINDs) [Bravo et al. 2007] are an extension of inclusion dependencies (INDs) of the form:

$$\varphi_{\text{cind}} = \forall \bar{x} \bar{y}_1 \bar{z}_1 (R_1(\bar{x}, \bar{y}_1, \bar{z}_1) \wedge \phi(\bar{y}_1) \rightarrow \exists \bar{y}_2 \bar{z}_2 (R_2(\bar{x}, \bar{y}_2, \bar{z}_2) \wedge \psi(\bar{y}_2)))$$

where  $R_1, R_2$  are relation atoms,  $\phi(\bar{z})$  and  $\psi(\bar{z})$  are defined as above. Similarly to CFDs, a CIND extends an IND  $R_1[X] \subseteq R_2[Y]$  by incorporating constant patterns specified by  $\phi(\bar{y}_1)$  and  $\psi(\bar{y}_2)$ , for constraining  $R_1$  tuples and  $R_2$  tuples, respectively. Traditional INDs are a special case of CINDs, in the absence of  $\phi(\bar{y}_1)$  and  $\psi(\bar{y}_2)$ .

We remark that integrity constraints are posed on databases  $D$  regardless of master data  $D_m$ . In contrast, containment constraints are defined on  $(D, D_m)$ .

From the proposition below it follows that by using containment constraints we can enforce both the relative completeness and the consistency of the data.

**Proposition 2.1:** (a) Denial constraints and (b) CFDs can be expressed as containment constraints (CCs) in CQ. (c) CINDs can be expressed as CCs in FO. In all three cases only an empty master data relation is required.  $\square$

PROOF. Consider arbitrary master data  $D_m$  that contains an empty relation  $\emptyset$ .

(a) A denial constraint  $\varphi_d$  of the form given above can be expressed as a single CC  $q(\bar{x}_1, \dots, \bar{x}_k) \subseteq \emptyset$  in CQ, where  $q$  is  $R_1(\bar{x}_1) \wedge \dots \wedge R_k(\bar{x}_k) \wedge \varphi(\bar{x}_1, \dots, \bar{x}_k)$ . Obviously for each database  $D$ ,  $D$  satisfies  $\varphi_d$  iff  $D$  and  $D_m$  satisfy the CC.

(b) A CFD  $\varphi_{\text{cfd}}$  is equivalent to two sets of CCs in CQ. For each pair  $(y_1, y_2)$  of variables in  $(\bar{y}_1, \bar{y}_2)$ , the first set contains  $q(\bar{x}_1, \bar{z}_1, \bar{y}_1, \bar{x}_2, \bar{z}_2, \bar{y}_2) \subseteq \emptyset$ , where  $q$  is

$$R(\bar{x}_1, \bar{z}_1, \bar{y}_1) \wedge R(\bar{x}_2, \bar{z}_2, \bar{y}_2) \wedge \phi(\bar{x}_1) \wedge \phi(\bar{x}_2) \wedge \bar{x}_1 = \bar{x}_2 \wedge y_1 \neq y_2.$$

This CC assures that the CFD is not violated by two distinct tuples.

The second set contains a CC of the form  $q'(\bar{x}, \bar{z}, \bar{y}) \subseteq \emptyset$  for each variable  $y$  in  $\bar{y}_1$  (resp.  $\bar{y}_2$ ) such that  $y = c$  is in  $\psi_1(\bar{y}_1)$  (resp.  $\psi_2(\bar{y}_2)$ ), where  $q'$  is

$$R(\bar{x}, \bar{z}, \bar{y}) \wedge \phi(\bar{x}) \wedge y \neq c.$$

These CCs ensure that  $\varphi_{\text{cfd}}$  is not violated by a single tuple that does not observe the constant patterns. It is easy to verify that for each database  $D$ ,  $D$  satisfies  $\varphi_{\text{cfd}}$  iff  $D$  and  $D_m$  satisfy these two sets of CCs.

(c) Given a CIND  $\varphi_{\text{cind}}$  of the form given above, we define a single CC  $q(\bar{x}, \bar{y}_1, \bar{z}_1) \subseteq \emptyset$  in FO, where  $q$  is  $R_1(\bar{x}, \bar{y}_1, \bar{z}_1) \wedge \phi(\bar{y}_1) \wedge \forall \bar{y}_2 \bar{z}_2 (\neg R_2(\bar{x}, \bar{y}_2, \bar{z}_2) \vee \neg \psi(\bar{y}_2))$ . Then for each database  $D$ ,  $D$  satisfies  $\varphi_{\text{cind}}$  iff  $D$  and  $D_m$  satisfy this CC.  $\square$

### 2.3 Relative Completeness Paradigms

Before we study the complexity and characterizations for deciding relatively complete databases, we first demonstrate how relative completeness and their associated decision problems (RCDP, RCQP) can help in assessing the quality of data and the quality of query answers. We illustrate this by using the application described in Example 1.1, which is a simplified instance of Customer Relationship Management (CRM), a typical usage scenario of MDM [Loshin 2008]. We should remark that relative completeness also finds similar applications in Enterprise Resource Planning (ERP), Supply Chain Management (SCM) and practical scenarios studied in, *e.g.*, [Levy 1996; Grahne 1991; Imieliński and Lipski 1984; Motro 1989].

Consider (a) a master relation  $D_m$  of the schema DCust, which maintains a complete list of domestic customers of a company, and (b) a database  $D$  with two “transitional” relations Cust and Supt that contain information about customers of the company (domestic or international) and about employees of the company for customer support, respectively (see Example 1.1). Consider a set  $V$  consisting of the CC  $\phi_0$  given in Example 2.1, assuring that  $D_m$  imposes an upper bound on domestic customers in the relations Cust and Supt.

*(1) Assessing the completeness of the data in a database.* Let us first consider a query  $Q_0$  posed on the database  $D$ , which is to find all the customers of the company based in NJ with ac = 908. In the absence of master data, one cannot decide whether  $Q_0(D)$  returns the complete list of customers we want to find. Indeed, as observed in [Loshin 2008], it is typical to find information missing from a transitional database in an enterprise; hence, one could only assume the OWA for  $D$ , and there is not much we can do about it. In contrast, provided the availability of the master data  $D_m$ , we can determine whether  $D$  has complete information to answer  $Q_0$ . More specifically, we can invoke a static analysis procedure for RCDP and decide whether  $D$  is in  $\text{RCQ}(Q_0, D_m, V)$ . If the procedure returns an affirmative answer, we know that we can trust the query answer  $Q_0(D)$ .

*(2) Guidance for what data should be collected in a database.* Suppose that the decision procedure for RCDP returns a negative answer, *i.e.*,  $D$  is not complete for  $Q_0$ . The next question is whether  $D$  can be expanded at all to be complete for  $Q_0$ ? To this end we capitalize on a decision procedure for RCQP, to determine whether  $\text{RCQ}(Q_0, D_m, V)$  is empty, *i.e.*, whether there exists a complete database for  $Q_0$  relative to the master data available. As will be seen in Section 4, the characterizations of relatively complete queries assure that there indeed exists a database complete for  $Q_0$  relative to  $(D_m, V)$ . This suggests that we should expand  $D$  to make it complete for  $Q_0$ . Furthermore, the characterizations to be given in Sec-

tion 3 tell us how to extend  $D$ : we can make  $D$  complete for  $Q_0$  by including the information about domestic customers that is in  $D_m$  but is missing from  $D$ .

(3) *A guideline for how master data should be expanded.* Now consider a query  $Q'_0$  to find all the customers of the company, domestic or international. In this case the RCDP analysis shows that  $D$  is not complete for  $Q'_0$  in the presence of  $D_m$ . Worse still, the RCQP analysis tells us that there exists *no* database complete for  $Q_0$  relative to the current master data  $D_m$ . This suggests that to find a complete answer for  $Q'_0$ , we need to expand the master data. As pointed out in [Loshin 2008], a practical challenge for MDM is to identify what data should be maintained as master data. The study of RCQP provides a guidance on how to identify master data such that one can find complete answer for queries commonly used in practice.

Similar RCDP and RCQP analyses can be carried out for the queries  $Q_1$  and  $Q_2$  of Example 1.1, to decide whether the database is complete for these queries and if not, whether we have to expand the database or the master data. We remark that the traditional OWA does not allow these analyses, in the absence of master data.

### 3. DECIDING RELATIVELY COMPLETE DATABASES

In this section we study  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ , the relatively complete database problem. Given a query  $Q$  in  $\mathcal{L}_Q$ , master data  $D_m$ , a set  $V$  of containment constraints (CCs) in  $\mathcal{L}_C$ , and a partially closed database  $D$  *w.r.t.*  $(D_m, V)$ , it is to determine whether  $D \in \text{RCQ}(Q, D_m, V)$ , *i.e.*, whether  $D$  has complete information to answer  $Q$ .

We first show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable when  $\mathcal{L}_Q$  or  $\mathcal{L}_C$  is either FO or FP. We then focus on  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  supports neither negation nor recursion. To provide insight into what makes a database relatively complete, we present characterizations of databases in  $\text{RCQ}(Q, D_m, V)$  when CCs and queries are in CQ. The characterizations readily extend to the settings where CCs are INDS or CQ, and where queries are in CQ or UCQ. We then establish matching lower and upper bounds for  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  range over CQ, UCQ and  $\exists\text{FO}^+$ , as well as for a special case where  $\mathcal{L}_C$  is the class of INDS.

#### 3.1 The Undecidability of RCDP for FO and FP

We start with negative results: when either  $\mathcal{L}_Q$  or  $\mathcal{L}_C$  is FO or FP, it is infeasible to determine whether a database  $D$  is relatively complete for a query  $Q$  *w.r.t.*  $(D_m, V)$ . This tells us that both  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  may impact the complexity of  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ . Worse still, the undecidability remains intact even when  $D_m$  and  $V$  are predefined.

**Theorem 3.1:**  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable when

- (1)  $\mathcal{L}_Q$  is FO and  $\mathcal{L}_C$  is CQ,
- (2)  $\mathcal{L}_C$  is FO and  $\mathcal{L}_Q$  is CQ,
- (3)  $\mathcal{L}_Q$  is FP and  $\mathcal{L}_C$  is CQ, or
- (4)  $\mathcal{L}_C$  is FP and  $\mathcal{L}_Q$  consists of a fixed query in FP.

If  $\mathcal{L}_Q$  is FO or FP, the problem remains undecidable for fixed master data and fixed containment constraints.  $\square$

**PROOF.** We first show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable for cases (1) and (2). Both proofs are by reduction from the satisfiability problem for FO queries, which

is to determine, given an FO query  $Q$  over a relational schema  $\mathcal{R}$ , whether there exists a nonempty database instance  $D$  of  $\mathcal{R}$  such that  $Q(D) \neq \emptyset$ . This problem is known to be undecidable (cf. [Abiteboul et al. 1995]).

Unless specified otherwise, we use  $\mathcal{R} = (R_1, \dots, R_n)$  and  $\mathcal{R}_m = (R_1^m, \dots, R_k^m)$  to denote the schemas of the database and master data, respectively. We write  $D = (I_1, \dots, I_n)$  for instances of  $\mathcal{R}$  and  $D_m = (I_1^m, \dots, I_k^m)$  for instances of  $\mathcal{R}_m$ .

(1) When  $\mathcal{L}_Q$  is FO and  $\mathcal{L}_C$  is CQ. Given an FO query  $Q$  over a relational schema  $\mathcal{R} = (R_1, \dots, R_n)$ , we define the schema of the input database and the master data to be  $\mathcal{R}$  and  $\mathcal{R}_m = (R_1^m)$ , respectively, where  $R_1^m$  is a unary relation. Let  $D = (I_1 = \emptyset, \dots, I_n = \emptyset)$ ,  $D_m = (I_1^m = \emptyset)$  and furthermore,  $V = \emptyset$ , *i.e.*, no CCs are specified. Assume *w.l.o.g.* that  $Q(D) = \emptyset$  since otherwise one could consider the query  $Q \setminus Q(D)$  instead. We define  $Q'$  to be the FO query derived from  $Q$  such that  $Q'(D) = \{()\}$  when  $Q(D) \neq \emptyset$ , and  $Q'(D) = \emptyset$  otherwise.

We show that  $D$  is complete for  $Q'$  relative to  $D_m$  and  $V$  iff  $Q$  is unsatisfiable. Indeed,  $Q'$  is complete iff  $Q'(D) = \emptyset = Q'(D')$  for each partially closed extension  $D'$  of  $D$  (recall that  $D$  is empty and  $V = \emptyset$ ). However, the latter holds iff  $Q(D') = \emptyset$  for all nonempty  $D'$ . Hence deciding relative completeness is undecidable when  $\mathcal{L}_Q$  is FO, even in the absence of containment constraints and for fixed  $D$  and  $D_m$ .

(2) When  $\mathcal{L}_C$  is FO and  $\mathcal{L}_Q$  is CQ. Given an FO query  $q$  over a relational schema  $\mathcal{R} = (R_1, \dots, R_n)$ , we define the same  $\mathcal{R}_m$ ,  $D$  and  $D_m$  as above. Let  $q'$  be the boolean query derived from  $q$  and defined as follows:  $q'(D) = \{()\}$  if  $q(D) \neq \emptyset$  or  $D = (\emptyset, \dots, \emptyset)$ , and  $q'(D) = \emptyset$  otherwise. We define  $V$  to be a singleton set consisting of CC:  $\{()\} \setminus q' \subseteq R_1^m$ . Clearly, for each instance  $D'$  of  $\mathcal{R}$  we have that  $(D', D_m) \models V$  iff either  $q(D') \neq \emptyset$  or  $D'$  is empty. Finally, we define a query  $Q$  over  $\mathcal{R}$  such that  $Q(D) = \{()\}$  if  $D$  is nonempty and  $Q(D) = \emptyset$  otherwise.

We show that  $D$  is complete for  $Q$  relative to  $D_m$  and  $V$  iff  $q$  is unsatisfiable. First assume that  $q$  is satisfiable, *i.e.*, there exists a nonempty instance  $D'$  of  $\mathcal{R}$  such that  $q(D') \neq \emptyset$ . Then  $(D', D_m) \models V$  but  $Q(D) \neq Q(D')$ . Hence  $D$  is not in  $\text{RCQ}(Q, D_m, V)$ . Conversely, if  $q$  is unsatisfiable, then for all nonempty instances  $D'$ ,  $(D', D_m) \not\models V$ . As a result,  $D$  is in  $\text{RCQ}(Q, D_m, V)$ . Hence deciding relative completeness is undecidable when  $\mathcal{L}_C$  is FO and the query language  $\mathcal{L}_Q$  can test for non-emptiness. Observe that the undecidability holds for a fixed  $D$  and  $D_m$ .

We next show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable for cases (3) and (4). Both proofs are by reduction from the emptiness problem for deterministic finite 2-head automata, which is known to be undecidable [Spielmann 2000].

(3) When  $\mathcal{L}_Q$  is FP and  $\mathcal{L}_C$  is CQ. Our reduction 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,  $\text{E+TC}$ , is undecidable over a schema having at least two non-nullary relation schemas, one of them being a function symbol. Although  $\text{E+TC}$  allows the negation of atomic expression as opposed to FP, the undecidability proof only uses a very restricted form of negation, which we can simulate using  $\neq$  and containment constraints. Recall from Section 2.1 that CQ and FP allow for equality ( $=$ ) and inequality ( $\neq$ ).

For the readers' convenience we present the necessary definitions taken from [Spiel-

mann 2000]. A *deterministic finite 2-head automaton* (or 2-head DFA for short) is a quintuple  $\mathcal{A} = (Q, \Sigma, \Delta, q_0, q_{acc})$  consisting of a finite set of states  $Q$ , an input alphabet  $\Sigma = \{0, 1\}$ , an initial state  $q_0$ , an accepting state  $q_{acc}$ , and a transition function  $\Delta : Q \times \Sigma_\varepsilon \times \Sigma_\varepsilon \rightarrow Q \times \{0, +1\} \times \{0, +1\}$ , where  $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ .

A *configuration of  $\mathcal{A}$*  is a triple  $(q, w_1, w_2) \in Q \times \Sigma^* \times \Sigma^*$ , representing that  $\mathcal{A}$  is in state  $q$ , 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  $(q_0, w, w)$ ; the successor configuration is defined as usual. The 2-head DFA  $\mathcal{A}$  *accepts*  $w$  if it can reach a configuration  $(q_{acc}, w_1, w_2)$  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 DFA's* is to determine, given a 2-head DFA  $\mathcal{A}$ , whether  $L(\mathcal{A})$  is empty or not.

Given a 2-head DFA  $\mathcal{A} = (Q, \Sigma, \delta, q_0, q_{acc})$ , we define schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , a database instance  $D$  of  $\mathcal{R}$  and master data instance  $D_m$  of  $\mathcal{R}_m$ , containment constraints  $V$  expressed in  $\mathcal{L}_C$  and a query  $Q \in \mathcal{L}_Q$ . We show that  $D$  is complete for  $Q$  relative to  $D_m$  and  $V$  iff  $L(\mathcal{A})$  is nonempty.

(a) The relational schema  $\mathcal{R}$  consists of two unary relations  $P(A)$ ,  $\bar{P}(A)$  and a binary relation  $F(A_1, A_2)$ . Intuitively, an instance  $I = (I_P, I_{\bar{P}}, I_F)$  of  $\mathcal{R}$  is to represent a string  $w \in \Sigma^*$  such that elements in  $I_P$  represent the positions in  $w$  where an 1 occurs; similarly,  $I_{\bar{P}}$  records those positions in  $w$  that are 0. The instance  $I_F$  encodes a successor relation over these positions. The relation schema  $\mathcal{R}_m$  consists of a single unary relation  $R_1^m$ .

(b) We define fixed instances:  $D = (I_1 = \emptyset, I_2 = \emptyset, I_3 = \emptyset)$  and  $D_m = (I_1^m = \emptyset)$ .

(c) We use containment constraints to assure that we only consider *well-formed* instances of  $P$ ,  $\bar{P}$  and  $F$ . That is, (i) instances  $I_P$  and  $I_{\bar{P}}$  of  $P$  and  $\bar{P}$  are disjoint; and each instance  $I_F$  of  $F$  must (ii) be a function, and (iii) contain a unique tuple of the form  $(k, k)$  for some constant  $k$  indicating the *final* position. We additionally require that each instance  $I_F$  of  $F$  contains a tuple of the form  $(0, i)$ , where 0 represents the *initial* position and  $i$  is some constant. The latter requirement will be assured by the query  $Q$  to be defined shortly.

More specifically, the set  $V$  of CCs consists of the following:

- $V_1 : \exists x (P(x) \wedge \bar{P}(x)) \subseteq \emptyset$ , enforcing that for each instance  $D' = (I'_1, I'_2, I'_3)$  of  $\mathcal{R}$  such that  $(D', D_m) \models V$ ,  $I'_1 \cap I'_2 = \emptyset$ ;
- $V_2 : \exists x, y, z (F(x, y) \wedge F(x, z) \wedge y \neq z) \subseteq \emptyset$ , ensuring that the relation  $I'_3$  encodes a function; and finally,
- $V_3 : \exists x, y (F(x, x) \wedge F(y, y) \wedge x \neq y) \subseteq \emptyset$ , asserting that the relation  $I'_3$  contains at most one tuple of the form  $(k, k)$ .

In short, for each instance  $D' = (I'_1, I'_2, I'_3)$  of  $\mathcal{R}$  that satisfies  $V$ ,  $D'$  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  $I'_3$  of  $F$  in  $D'$ .

(d) Before we define the query  $Q$ , we show, following [Spielmann 2000], how the non-emptiness of  $L(\mathcal{A})$  can be expressed in terms of an E+TC-formula over  $R$ . Consider a transition  $\delta \in \Delta$  of the form  $\delta = (q, in_1, in_2) \rightarrow (q', move_1, move_2)$ . This can be encoded by means of the conjunctive query

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

Here  $\alpha_i(x) = \exists y(F(x, y) \wedge x \neq y \wedge P(x))$  if  $\text{in}_i = 1$ ;  $\alpha_i(x) = \exists y(F(x, y) \wedge x \neq y \wedge \bar{P}(x))$  if  $\text{in}_i = 0$ ; and  $\alpha_i(x) = F(x, x)$  if  $\text{in}_i = \varepsilon$ . Moreover,  $\beta_i(x, y) = F(x, y)$  if  $\text{move}_i = +1$  and  $\beta_i(x, y) = (x = y)$  if  $\text{move}_i = 0$ . Intuitively,  $\alpha_i(x)$  enforces  $x$  to be a position in the string coded by  $P$  or  $\bar{P}$  that has a successor, unless  $x$  is the final position where  $\alpha_i(x)$  demands  $F(x, x)$ . Moreover,  $\beta_i(x, y)$  ensures that  $x$  and  $y$  are consecutive positions when  $\mathcal{A}$  makes a move (with head  $i$ ) and  $x = y$  otherwise. Then  $\Phi = \exists y_1 \exists y_2 [\text{TC}_{x,y,z;x',y',z'} \bigvee_{\delta \in \Delta} \varphi_\delta](q_0, 0, 0, q_{acc}, y_1, y_2)$  is satisfiable iff  $L(\mathcal{A}) \neq \emptyset$ .

Clearly, we can compute  $\Phi$  using a query  $Q'$  in FP (*i.e.*, DATALOG). Recall that we still need to assure the existence of an initial and a final position in the instance of  $F$  in  $D'$ . We therefore define boolean query  $Q = Q' \wedge Q_{ini} \wedge Q_{fin}$ , where  $Q_{ini} = \exists x F(0, x)$  and  $Q_{fin} = \exists x F(x, x)$ .

This concludes the construction of  $\mathcal{R}$ ,  $\mathcal{R}_m$ ,  $D$ ,  $D_m$ ,  $V$  and  $Q$ .

We now show that  $D$  is complete for  $Q$  relative to  $D_m$  and  $V$  iff  $L(\mathcal{A}) = \emptyset$ . Because  $D$  is empty, we have that  $Q(D) = \emptyset$ .

Suppose that  $L(\mathcal{A}) = \emptyset$ . Then there does not exist any well-formed instance that makes  $Q$  true. Hence,  $D$  is complete as required. Conversely, suppose that  $Q(D') = \emptyset$  for all instances  $D' = (I'_1, I'_2, I'_3)$  of  $\mathcal{R}$  such that  $(D', D_m) \models V$ . Then either  $Q_{ini} \wedge Q_{fin}$  is not satisfied in  $D'$ , or  $Q'$  is not satisfied in  $D'$ . In other words, there does not exist any well-formed instance  $D'$  of  $\mathcal{R}$  that makes  $Q'$  true. That is,  $L(\mathcal{A}) = \emptyset$  as desired.

Observe that the CCs of  $V$  are expressed in CQ and are predefined, *i.e.*, they are independent of the 2-head DFA. Similarly, both  $D$  and  $D_m$  are fixed.

(4) When  $\mathcal{L}_C$  is FP and  $\mathcal{L}_Q$  is a fixed query in FP. We show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable by reduction from the emptiness problem of deterministic finite 2-head DFA. The proof is referred to the Appendix.

We remark that when  $\mathcal{L}_Q$  is FO or FP, it remains undecidable if  $D_m$  and  $V$  are fixed, as verified in the proofs above.  $\square$

### 3.2 Characterizations of Relatively Complete Databases for CQ

In light of the undecidability results, in the rest of the section we focus on query languages that support neither negation nor recursion, namely, CQ, UCQ and  $\exists\text{FO}^+$ . To understand what it takes to make a database  $D$  complete for a query  $Q$  relative to master data  $D_m$  and a set  $V$  of CCs, we identify sufficient and necessary conditions for  $D$  to be included in  $\text{RCQ}(Q, D_m, V)$ . These conditions provide a guidance for what data should be collected by  $D$  in order to accurately answer query  $Q$ .

To simplify the discussion we assume in the rest of the section that  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are the same language, unless explicitly stated otherwise. In practice, if users are allowed to define CCs in a query language, there is no reason for not allowing them to issue queries in the same language.

Nevertheless, we also consider a special case where CCs are INDs, *i.e.*, CCs of the form  $q_v \subseteq p$  when both  $q_v$  and  $p$  are projection queries on  $D$  and  $D_m$ , respectively. We consider  $V$  consisting of INDs, as commonly found in practice, while the user queries are expressed in CQ, UCQ or  $\exists\text{FO}^+$ .

We first present characterizations when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ. We then adjust the conditions to characterize relatively complete databases when  $\mathcal{L}_Q$  is CQ and  $\mathcal{L}_C$  is the class of INDs, and when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ. It should be remarked that along the same lines conditions can be developed when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ .

**When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ.** We start with notations to express the conditions.

*Tableau queries and valuations.* To simplify the discussion, we consider CQ queries over a single relation. It does not lose generality due to the lemma below. For a relational schema  $\mathcal{R}$ , we denote by  $\text{inst}(\mathcal{R})$  the set of all database instances of  $\mathcal{R}$ .

**Lemma 3.2:** For each relational schema  $\mathcal{R} = (R_1, \dots, R_n)$ , there exist a single relation schema  $R$ , a linear-time computable function  $f_D$  from  $\text{inst}(\mathcal{R})$  to the set  $\text{inst}(R)$  of instances of  $R$ , and a linear-time function  $f_Q: \text{CQ} \rightarrow \text{CQ}$  such that for each instance  $D$  of  $\mathcal{R}$  and each CQ query  $Q$  over  $\mathcal{R}$ ,  $Q(D) = f_Q(Q)(f_D(D))$ .  $\square$

PROOF. We assume *w.l.o.g.* that all  $R_i$ 's in  $\mathcal{R}$  have the same set of attributes, since one can make the  $R_i$ 's uniform by renaming attributes and adding dummy attributes. We denote this uniform set of attributes by  $R'$ . Consider a distinct attribute  $A_R$  that takes values from  $\text{dom}(A) = [1, n]$ . Define  $R$  to be the schema consisting of the attributes in  $R'$  augmented with the attribute  $(A_R : \text{dom}(A))$ . Define  $f_D$  such that for each instance  $D = (I_1, \dots, I_n)$  of  $\mathcal{R}$ ,  $f_D(D) = \bigcup_{j \in [1, n]} I_j \times (A_R = j)$ . Furthermore, define  $f_Q$  such that for each CQ query  $Q$  defined on  $\mathcal{R}$ ,  $f_Q(Q)$  replaces every occurrence of  $R_j$  with a Project-Select expression  $\pi_{R'}(\sigma_{A_R=j}(R))$ . Then it is easy to verify that  $Q(D) = f_Q(Q)(f_D(D))$ . Furthermore,  $f_D$  and  $f_Q$  can be constructed in linear time.  $\square$

In light of this, we represent a CQ query  $Q$  as a tableau query  $(T_Q, u_Q)$ , where  $T_Q$  denotes formulas in  $Q$  and  $u_Q$  is the output summary (see *e.g.*, [Abiteboul et al. 1995] for details). For each variable  $x$  in  $Q$ , we use  $\text{eq}(x)$  to denote the set of variables  $y$  in  $Q$  such that  $x = y$  is induced from equality in  $Q$ . In  $T_Q$ , we represent atomic formula  $x = y$  by assigning the same distinct variable to all variables in  $\text{eq}(x)$ , and  $x = 'c'$  by substituting constant  $'c'$  for each occurrence of  $y$  in  $\text{eq}(x)$ . This is well defined when  $Q$  is satisfiable, *i.e.*, when there exists a database  $D$  such that  $Q(D)$  is nonempty. Note that the size of  $T_Q$  and the number of variables in  $T_Q$  are bounded by the size of  $Q$ .

Consider a CQ query  $Q$ , master database  $D_m$ , a set  $V$  of CCs in CQ, and a partially closed database  $D$  *w.r.t.*  $(D_m, V)$ . Assume *w.l.o.g.* that  $Q$  is satisfiable, since otherwise  $D$  is trivially complete for  $Q$  *w.r.t.* any  $(D_m, V)$  as long as  $(D, D_m) \models V$ .

We denote by **Adom** the set consisting of (a) all constants that appear in  $D, D_m, Q$  or  $V$ , and (b) a set **New** of distinct values not in  $D, D_m, Q$  and  $V$ , one for each variable that is in either  $T_Q$  or in the tableau representations of the queries in  $V$ ; when there are more variables with finite domain than values in  $\mathbf{d}_f$  (recall  $\mathbf{d}_f$  from Section 2),  $\mathbf{d}_f \subseteq \text{Adom}$ .

For each variable  $y$  in  $T_Q$ , we define its active domain, denoted by  $\text{adom}(y)$ . If  $y$  appears in some column  $A$  in  $T_Q$  such that  $\text{dom}(A)$  is finite  $\mathbf{d}_f$ , then  $\text{adom}(y)$  is  $\mathbf{d}_f \cap \text{Adom}$ . Otherwise  $\text{adom}(y)$  is **Adom**.

A valuation  $\mu$  for variables in  $T_Q$  is said to be *valid* if (a) for each variable  $y$  in  $T_Q$ ,  $\mu(y)$  is a value from  $\text{adom}(y)$ , and (b)  $Q(\mu(T_Q))$  is nonempty, *i.e.*,  $\mu$  observes

inequality formulas  $x \neq y$  and  $x \neq 'b'$  specified in  $Q$ .

*Characterizations.* To illustrate the conditions for  $D$  to be in  $\text{RCQ}(Q, D_m, V)$ , let us first examine some examples of relatively (in)complete databases.

**Example 3.1:** Recall the query  $Q_2$  from Example 1.1 and CC  $\phi_1$  from Example 2.1, both defined over schema **Supt**. The query is to find all customers supported by the employee  $e_0$ , *i.e.*, all **Supt** tuples  $t$  with  $t[\text{eid}] = 'e_0'$ , and can be expressed in CQ. The CC ensures that no employee supports more than  $k$  customers. Consider an instance  $D_1$  of **Supt** such that  $Q_2(D_1)$  returns  $k$  distinct tuples. Then adding any new tuple to  $D_1$  violates  $\phi_1$  if the new tuple is in connection with  $e_0$ , and it does not change the answer to  $Q_2$  in  $D_1$  otherwise. That is, there already exist  $k$  “witnesses” for the completeness of  $Q_2$  for  $D_1$  relative to  $D_m$  and  $\phi_1$ , which “block” further additions of tuples in connection with  $e_0$ . Thus  $D_1$  is in  $\text{RCQ}(Q_2, D_m, \{\phi_1\})$ .

As another example, recall from Example 1.1 the FD  $\text{eid} \rightarrow \text{dept}, \text{cid}$  on **Supt**. By Proposition 2.1, we can express the FD as two CCs in CQ, denoted by  $\Sigma_2$ , using  $D_m$  that has an empty relation. Consider an instance  $D_2$  of **Supt** in which there exists no tuple  $t$  such that  $t[\text{eid}] = 'e_0'$ . Then  $D_2$  is not complete for  $Q_2$  relative to  $(D_m, \Sigma_2)$ . Indeed,  $Q_2(D_2) = \emptyset$ , but one can add a tuple  $t$  with  $t[\text{eid}] = 'e_0'$  that leads to a nonempty answer to  $Q_2$  in the updated  $D_2$ .  $\square$

These examples tell us that there are intriguing interactions among  $Q$ ,  $V$  and the data already in  $D$ . While it is hard to characterize the interactions syntactically, we provide sufficient and necessary conditions for  $D$  to be in  $\text{RCQ}(Q, D_m, V)$ . These conditions are expressed in terms of a notion of bounded databases given as follows.

A database  $D$  is said to be *bounded by*  $(D_m, V)$  for  $Q$  if for each valid valuation  $\mu$  for variables in  $T_Q$ , either  $(D \cup \mu(T_Q), D_m) \not\models V$  or  $\mu(u_Q) \in Q(D)$ .

More specifically,  $D$  is bounded if for each valid valuation  $\mu$ ,

(C1) when  $Q(D) = \emptyset$ , then  $(D \cup \mu(T_Q), D_m) \not\models V$ ;

(C2) when  $Q(D) \neq \emptyset$ , if  $(D \cup \mu(T_Q), D_m) \models V$ , then  $\mu(u_Q) \in Q(D)$ .

The proposition below tells us that bounded databases characterize relatively complete databases:  $D$  is bounded iff for each set  $\Delta$  of tuples, if  $Q(D) \neq Q(D \cup \Delta)$ , then  $(D \cup \Delta, D_m) \not\models V$ . In other words, adding tuples to  $D$  either violates  $V$  or does not change  $Q(D)$ . Furthermore, the notion of bounded databases reveals the small model property for checking relative completeness: while there may exist infinitely many  $\Delta$ 's, it suffices to inspect  $\Delta$  constructed with values in **Adom** only.

In Section 3.3, we shall develop algorithms (in  $\Pi_2^P$ ) for checking the conditions C1 and C2 (and for checking analogous conditions when  $\mathcal{L}_Q$  is UCQ and  $\exists\text{FO}^+$ ). By the propositions below, these algorithms effectively decide RCDP.

**Proposition 3.3:** For each query  $Q$  in CQ, master data  $D_m$ , each set  $V$  of CCs in CQ, and each partially closed  $D$  *w.r.t.*  $(D_m, V)$ ,  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff  $D$  is bounded by  $(D_m, V)$  for  $Q$ , *i.e.*,  $D$  satisfies the conditions C1 and C2.  $\square$

**PROOF.** We show that C1 and C2 are sufficient and necessary conditions for the relative completeness. We first consider C1, followed by C2.

We show that C1 is a sufficient and necessary condition for the relative completeness of  $D$  when  $Q(D) = \emptyset$ . Let  $(T_Q, u_Q)$  be the tableau representation of  $Q$ . Suppose that  $D$  is complete. Assume by contradiction that there exists a valid

valuation  $\mu$  of  $T_Q$  such that  $(D \cup \mu(T_Q), D_m) \models V$ . Let  $D' = D \cup \mu(T_Q)$ . Then  $(D', D_m) \models V$  and moreover,  $\mu(T_Q)$  is nonempty by the definition of valid valuations. Hence,  $Q(\mu(T_Q)) \neq \emptyset$  and  $Q(D') \neq \emptyset$  by the monotonicity of CQ queries. Therefore,  $Q(D) \neq Q(D')$ , which contradicts the assumption that  $D$  is complete.

Conversely, suppose  $D$  is not complete. Then there exists an extension  $D'$  of  $D$  such that  $(D', D_m) \models V$  and  $Q(D')$  is nonempty. Then there exists a valuation  $\mu'$  for variables in  $T_Q$  that draws values from  $D'$  such that  $(D \cup \mu'(T_Q), D_m) \models V$ , by the monotonicity of CQ queries, and moreover,  $\mu'(u_Q)$  is nonempty. Define a valuation  $\mu$  such that for each variable  $x$  in  $T_Q$ ,  $\mu(x)$  is a distinct value in **New** if  $\mu'(x)$  is not in **Adom**, and  $\mu(x) = \mu'(x)$  otherwise. Then  $(D \cup \mu(T_Q), D_m) \models V$  and  $\mu(u_Q)$  is nonempty, by the choice of the values in **New**. That is,  $\mu$  is a valid valuation violating condition C1.

Similarly, we show that the condition C2 is a sufficient and necessary condition for the relative completeness of  $D$  when  $Q(D) \neq \emptyset$ . Suppose that  $D$  is complete. Then for each extension  $D'$  of  $D$ , if  $(D', D_m) \models V$  then  $Q(D) = Q(D')$ . As a result, for each valid valuation  $\mu$  for variables in  $T_Q$ , if  $(D \cup \mu(T_Q), D_m) \models V$  then  $Q(D)$  must include  $\mu(u_Q)$  by the definition of relatively complete databases.

Conversely, suppose that  $D$  is not complete. We show that the condition C2 does not hold. Since  $D$  is not complete, there exists an extension  $D'$  of  $D$  such that  $(D', D_m) \models V$  and  $Q(D) \neq Q(D')$ . Then there must exist a valuation  $\mu'$  for variables in  $T_Q$  that draws values from  $D'$  such that  $(D \cup \mu'(T_Q), D_m) \models V$  by the monotonicity of CQ queries, and moreover,  $\mu'(u_Q) \notin Q(D)$ . Define a valuation  $\mu$  such that for each variable  $x$  in  $T_Q$ ,  $\mu(x)$  is a distinct value in **New** if  $\mu'(x)$  is not in **Adom**, and  $\mu(x) = \mu'(x)$  otherwise. Then  $(D \cup \mu(T_Q), D_m) \models V$  and  $\mu(u_Q) \notin Q(D)$ , by the choice of the values in **New**. Thus the condition C2 does not hold since  $\mu$  is a valid valuation,  $(D \cup \mu(T_Q), D_m) \models V$ , but  $\mu(u_Q) \notin Q(D)$ .  $\square$

**When  $\mathcal{L}_C$  is the class of INDs.** If  $V$  is a set of INDs, the notion of bounded databases is simpler: a database  $D$  is said to be *bounded by*  $(D_m, V)$  for  $Q$  if

(C3) for each valid valuation  $\mu$  of  $T_Q$ , either  $(\mu(T_Q), D_m) \not\models V$  or  $\mu(u_Q) \in Q(D)$ .

The result of Proposition 3.3 holds for the revised notion of bounded databases.

**Corollary 3.4:** For each query  $Q$  in CQ, master data  $D_m$ , each set  $V$  of INDs, and each partially closed  $D$  w.r.t.  $(D_m, V)$ ,  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff  $D$  is bounded by  $(D_m, V)$  for  $Q$ , i.e., the condition C3 holds.  $\square$

**PROOF.** We show that when  $Q(D) \neq \emptyset$ ,  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff for each valid valuation  $\mu$  of  $T_Q$ , if  $(\mu(T_Q), D_m) \models V$ , then  $\mu(u_Q) \in Q(D)$ . The proof for Proposition 3.3 tells us that  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff for each valid valuation  $\mu$  of  $T_Q$ , if  $(D \cup \mu(T_Q), D_m) \models V$ , then  $\mu(u_Q) \in Q(D)$ . Since  $V$  is a set of INDs,  $(D \cup \mu(T_Q), D_m) \models V$  iff  $(D, D_m) \models V$  and  $(\mu(T_Q), D_m) \models V$ . Since  $D$  is partially closed w.r.t.  $(D_m, V)$ ,  $(D, D_m) \models V$ . Thus  $(D \cup \mu(T_Q), D_m) \models V$  iff  $(\mu(T_Q), D_m) \models V$ . From this and Proposition 3.3, Corollary 3.4 follows.

The proof for  $Q(D) = \emptyset$  is similar as indicated in the proof of Proposition 3.3.  $\square$

**When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ.** Consider a query in UCQ:  $Q = Q_1 \cup \dots \cup Q_k$ , where  $Q_i$  is in CQ for each  $i \in [1, k]$ . We represent  $Q_i$  as a tableau query  $(T_i, u_i)$ . Then

a valuation  $\mu$  for  $Q$  is  $(\mu_1, \dots, \mu_k)$  such that for each  $i \in [1, k]$ ,  $\mu_i$  is a valuation for variables in  $T_i$  and moreover, for each variable  $y$  in  $T_i$ ,  $\mu_i(y) \in \text{adom}(y)$ . The valuation is *valid* if there exists some  $j \in [1, k]$  such that  $Q_j(\mu_i(T_j))$  is nonempty.

A database  $D$  is said to be *bounded by*  $(D_m, V)$  for  $Q$  if

(C4) for each valid valuation  $\mu = (\mu_1, \dots, \mu_k)$  for  $Q$ , either  $(D \cup \Delta, D_m) \not\models V$ , or for each  $i \in [1, k]$ ,  $\mu_i(u_i) \in Q(D)$ , where  $\Delta$  denotes  $\mu_1(T_1) \cup \dots \cup \mu_k(T_k)$ .

Using this revised notion, the result of Proposition 3.3 also remains intact when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ.

**Corollary 3.5:** For each query  $Q$  in UCQ, master data  $D_m$ , each set  $V$  of CCs in UCQ, and each partially closed database  $D$  w.r.t.  $(D_m, Q)$ ,  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff  $D$  is bounded by  $(D_m, V)$  for  $Q$ , i.e., the condition C4 holds.  $\square$

PROOF. We show that when  $Q(D) \neq \emptyset$ ,  $D$  is in  $\text{RCQ}(Q, D_m, V)$  iff the condition C4 holds, i.e., for each valid valuation  $\mu = (\mu_1, \dots, \mu_k)$  for  $Q$ , if  $(D \cup \Delta, D_m) \models V$ , then for each  $i \in [1, k]$ ,  $\mu_i(u_i) \in Q(D)$ , where  $\Delta$  denotes  $\mu_1(T_1) \cup \dots \cup \mu_k(T_k)$ . The proof for  $Q(D) = \emptyset$  is similar.

First, suppose that  $D$  is complete. Then for each extension  $D'$  of  $D$ , if  $(D', D_m) \models V$  then  $Q(D) = Q(D')$ . Hence for each valid valuation  $\mu$  for  $Q$ , if  $(D \cup \Delta, D_m) \models V$  then there exists no  $i \in [1, k]$  such that  $\mu_i(u_i) \notin Q(D)$ . Thus the condition C4 holds.

Conversely, suppose that  $D$  is not complete. We show that the condition C4 does not hold. Since  $D$  is not complete, there exists an extension  $D'$  of  $D$  such that  $(D', D_m) \models V$  and  $Q(D) \neq Q(D')$ . Then there exists a valuation  $\mu' = (\mu'_1, \dots, \mu'_k)$  for variables in  $(T_1, \dots, T_n)$  that draws values from  $D'$  such that  $(D \cup \Delta', D_m) \models V$  and moreover, there exists  $i \in [1, k]$  such that  $\mu'_i(u_i) \notin Q(D)$ , where  $\Delta'$  is  $\mu'_1(T_1) \cup \dots \cup \mu'_k(T_k)$ . Define a valuation  $\mu = (\mu_1, \dots, \mu_k)$  such that each  $\mu_j$  is defined in the same way as in the proof for Proposition 3.3. Then the argument given there suffices to show the following: (a)  $(D \cup \Delta, D_m) \models V$ , where  $\Delta = \mu_1(T_1) \cup \dots \cup \mu_k(T_k)$ , and (b)  $\mu_i(u_i) \notin Q(D)$ . That is,  $\mu$  is a valid valuation,  $(D \cup \Delta, D_m) \models V$ , but there exists  $i \in [1, k]$  such that  $\mu_i(u_i) \notin Q(D)$ . Hence the condition C4 does not hold.

Along the same lines as the proof of Proposition 3.3 one can readily verify the result for  $Q(D) = \emptyset$ .  $\square$

### 3.3 The Complexity of RCDP for CQ, UCQ and $\exists\text{FO}^+$

Capitalizing on the characterizations given above, we next provide complexity bounds on  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  for CQ, UCQ and  $\exists\text{FO}^+$ . We show that the absence of negation and recursion makes our lives easier:  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is in the polynomial hierarchy for these query languages. Furthermore, the complexity bounds are rather robust: the problem is  $\Pi_2^P$ -complete when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ , and it remains  $\Pi_2^P$ -complete when  $\mathcal{L}_Q$  is CQ and  $\mathcal{L}_C$  is the class of INDs.

**Theorem 3.6:**  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\Pi_2^P$ -complete when

- (1)  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ, UCQ or  $\exists\text{FO}^+$ ,
- (2)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ,
- (3)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ, or
- (4)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ .  $\square$

PROOF. It suffices to show the following complexity bounds. (1) Lower bound:  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\Pi_2^p$ -hard when  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ. (2) Upper bound:  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is in  $\Pi_2^p$  when  $\mathcal{L}_C$  and  $\mathcal{L}_Q$  are both  $\exists\text{FO}^+$ . For if these hold, then the complexity bounds remain the same for cases (1)–(4).

*Lower bound.* We show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\Pi_2^p$ -hard when  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ (*i.e.*, case (1)), by reduction from the  $\forall^*\exists^*$ -3SAT-problem. The latter is to determine, given  $\varphi = \forall X \exists Y C_1 \wedge \dots \wedge C_r$ , whether or not  $\varphi$  evaluates to true. Here  $X = \{x_1, \dots, x_n\}$  and  $Y = \{y_1, \dots, y_m\}$ , which are sets of variables; and  $C_1 \wedge \dots \wedge C_r$  is an instance of 3SAT, *i.e.*, each clause  $C_i$  is of the form  $\ell_1^i \vee \ell_2^i \vee \ell_3^i$ , where for  $k \in [1, 3]$  and  $i \in [1, r]$ ,  $\ell_k^i$  is either a variable or the negation of a variable in  $X \cup Y$ . This problem is known to be  $\Pi_2^p$ -complete (cf. [Papadimitriou 1994]). We construct relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , instances  $D$  and  $D_m$  over  $\mathcal{R}$  and  $\mathcal{R}_m$ , respectively, a set  $V$  of INDs as CCs, and a query  $Q$  in CQ such that  $D$  is complete for  $Q$  relative to  $(D_m, V)$  iff  $\varphi$  evaluates to true.

(a) The relational schema  $\mathcal{R}$  consists of six relation schemas:  $R_1(x)$ ,  $R_2(x_1, x_2, x_3)$ ,  $R_3(x_1, x_2, x_3)$ ,  $R_4(x, \bar{x})$ ,  $R_5(x_1, x_2, x_3)$  and  $R_6(x)$ . We say that instances  $I = (I_1, I_2, I_3, I_4, I_5, I_6)$  of  $\mathcal{R}$  are well-formed iff

- $I_1 = I_{01} = \{(0), (1)\}$ , encoding the boolean domain  $\{0, 1\}$ ;
- $I_2 = I_\vee = \{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 1)\}$ , encoding the truth table of disjunction, *i.e.*,  $I_\vee(x, y, z)$  iff  $x \vee y = z$ ;
- $I_3 = I_\wedge = \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 1)\}$ , encoding conjunction;
- $I_4 = I_\neg = \{(0, 1), (1, 0)\}$ , encoding negation, *i.e.*,  $I_\neg(x, y)$  iff  $x = \bar{y}$ ;
- $I_5 = I_c = \{(0, 0, 1), (0, 1, 1), (1, 0, 0), (1, 1, 1)\}$ ; here  $I_c$  is a truth table such that  $I_c(x, y, 1)$  iff  $x = 0$  or  $x = 1$  and  $y = 1$ ; and finally,
- $\{(1)\} \subseteq I_6 \subseteq \{(0), (1)\}$ , which states that  $I_6$  should contain at least (1) and at most (0) and (1).

The schema  $\mathcal{R}_m$  is the same as  $\mathcal{R}$ , *i.e.*,  $\mathcal{R}_m = (R_1^m = R_1, R_2^m = R_2, R_3^m = R_3, R_4^m = R_4, R_5^m = R_5, R_6^m = R_6)$ .

(b) We define  $D = (I_{01}, I_\vee, I_\wedge, I_\neg, I_c, I_6 = \{(1)\})$ , and  $D_m = (I_1^m = I_{01}, I_2^m = I_\vee, I_3^m = I_\wedge, I_4^m = I_\neg, I_5^m = I_c, I_6^m = \{(0), (1)\})$ .

(c) The set  $V$  of CCs consists of the following INDs:  $R_i \subseteq R_i^m$ , for  $i \in [1, 6]$ . For each instance  $D' = (I'_1, I'_2, I'_3, I'_4, I'_5, I'_6)$  of  $\mathcal{R}$  such that  $D \subseteq D'$  and  $(D', D_m) \models V$ , it is easily verified that  $D'$  is necessarily well-formed.

(d) The query  $Q$  is constructed in several steps. Let  $D' = (I'_1, I'_2, I'_3, I'_4, I'_5, I'_6)$  be an instance of  $\mathcal{R}$  such that  $D \subseteq D'$  and  $(D', D_m) \models V$ . First, we encode truth assignments for the  $\forall^*\exists^*$ -3SAT-instance  $\varphi$ . Since  $D'$  is assured to be well-formed by  $V$ ,  $I'_1 = I_{01}$ ; hence one can construct all  $n + m$ -ary binary vectors by means of the Cartesian product  $R_1 \times \dots \times R_1$  ( $n + m$  times). Furthermore, we encode the 3SAT-instance in  $\varphi$ , by leveraging instances of  $R_2$ ,  $R_3$  and  $R_4$ , which correspond to the boolean truth table of disjunction, conjunction and negation, respectively. More specifically, we construct an  $n + m + 1$ -ary relation  $T$ , with attributes  $x_1, \dots, x_n, y_1, \dots, y_m, z$  such that for each tuple  $t$  in that relation,  $t[z]$  is 1 if  $C_1 \wedge \dots \wedge C_r$  evaluates to true for the truth assignment of  $X \cup Y$  given by

$t[x_1, \dots, x_n, y_1, \dots, y_m]$ , and  $t[z]$  is 0 otherwise. Note that  $I_\vee$  and  $I_\neg$  are required since CQ has neither disjunction nor negation. We included  $I_\wedge$  just for convenience.

We next select certain truth assignments for  $X$ , by making use of the relations  $R_6$  and  $R_5$ . We take the product  $R_6 \times T$ , and denote the first attribute in  $R_6 \times T$  by  $z'$ . When evaluated on  $D'$ , the result of this query is  $\{1\} \times T(D')$  when  $I'_6$  only contains (1), and it is  $\{1\} \times T(D') \cup \{0\} \times T(D')$  otherwise. Finally, we use relation  $R_5$  to select certain tuples  $s$  from  $I'_6 \times T(D')$ . More specifically, the query  $Q$  is:

$$Q(\bar{x}) = \pi_{\bar{x}}(R_6(z') \times T(\bar{x}, \bar{y}, z) \times R_5(z', z, 1)).$$

When  $I'_6 = \{(1)\}$ , we only want to retrieve those truth assignments for  $X$  for which there exists a truth assignment for  $Y$  that makes  $C_1 \wedge \dots \wedge C_r$  true. In other words, we select those tuples  $s$  from  $I'_6 \times T(D')$  for which  $I'_5(s[z'], s[z], 1)$  exists. Indeed, in this case  $s[z]$  must be 1 to be selected. When  $I'_6 = \{(0), (1)\}$ , we simply want to find all possible truth assignments for  $X$ . Again, we select those tuples  $s$  from  $I'_6 \times T(D')$  for which  $I'_5(s[z'], s[z], 1)$  exists. In this case  $s[z]$  can be either 0 or 1 to be selected.

This completes the construction of  $\mathcal{R}$ ,  $\mathcal{R}_m$ ,  $D$ ,  $D_m$ ,  $V$  and  $Q$ .

We now show that  $D$  is complete for  $Q$  relative to  $(D_m, V)$  iff  $\varphi$  is true. If  $\varphi$  is true then  $Q(D)$  will return all truth assignments for  $X$ , and so will  $Q(D')$  for each partially closed extension  $D'$ . Conversely, if  $D$  is complete for  $Q$  relative to  $(D_m, V)$ , then in particular we must have that  $Q(D) = Q(D')$  with  $D' = (I'_1 = I_{01}, I'_2 = I_\vee, I'_3 = I_\wedge, I'_4 = I_\neg, I'_5 = I_c, I'_6 = \{(0), (1)\})$ . However, since  $I_6 = \{(1)\}$ ,  $Q(D)$  only returns the truth assignments for  $X$  for which there exists a truth assignment for  $Y$  that makes  $C_1 \wedge \dots \wedge C_r$  true. Since  $I'_6 = \{(0), (1)\}$ ,  $Q(D')$  will return *all* truth assignments for  $X$ . As consequence,  $\varphi$  must be true.

This completes the proof for the  $\Pi_2^p$  lower bound. It should be remarked that in the proof,  $D_m$  and  $V$  are fixed.

*Upper bound.* We next establish the  $\Pi_2^p$  upper bound. To illustrate the main idea, we first provide a  $\Pi_2^p$  algorithm for testing relative completeness for the case when both  $\mathcal{L}_C$  and  $\mathcal{L}_Q$  are UCQ (*i.e.*, case (3)). We then show how the algorithm can be modified for the case when both  $\mathcal{L}_C$  and  $\mathcal{L}_Q$  are  $\exists\text{FO}^+$  (*i.e.*, case (4)).

*When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ.* Let  $V$  be a set of CCs expressed in UCQ,  $D$  a database, and  $D_m$  a master data instance of schema  $\mathcal{R}_m$ . Let  $Q$  be a UCQ query  $Q = Q_1 \cup \dots \cup Q_p$ , where  $Q_i$  is in CQ. By Lemma 3.2, we assume that the tableau representation of  $Q$  consists of  $p$  tableaux  $(T_i, u_i)$ , one for each CQ query  $Q_i$ . Recall the definitions of  $\text{Adom}$ ,  $\text{adom}(y)$  and valid valuations from Section 3.2. We assume that the set  $\text{New}$  now consists of distinct values, one for each variable in  $(T_i, u_i)$  or  $V$ . Similarly, the notion of valid valuations generalizes to a set of tableaux in a straightforward manner.

We now present a  $\Pi_2^p$  algorithm for testing the relative completeness of a database, based on Corollary 3.5. The algorithm is in fact a  $\Sigma_2^p$  algorithm for the complement of our problem: given  $D$ ,  $D_m$ ,  $V$  and  $Q$ , it returns “yes” if there exists a tuple  $s$  such that  $s \notin Q(D)$  but  $s \in Q(D')$  for some  $D' \supseteq D$  such that  $D$  and  $D_m$  satisfy  $V$ , and returns “no” otherwise. More specifically, the algorithm does the following:

- (1) Guess the following:

- (a) a component query  $Q_i$  (from  $i \in [1, p]$ ); and
  - (b) a valuation  $\nu$  for  $T_i$  such that for each variable  $y$ ,  $\nu(y) \in \text{adom}(y)$ .
- (2) If  $\nu$  is invalid then reject the current guess. Note that the validity of a valuation can be verified in PTIME. Otherwise, let  $s = \nu(u_i)$ . Observe that this is well-defined since we assume that  $u_i$  solely consists of variables. Furthermore, assume that  $T_i$  consists of  $\ell$  tuples  $t_j^i$ . Then  $s \in Q(\Delta)$  since  $\mu$  is valid, where  $\Delta = \{\nu(t_j^i) \mid j \in [1, \ell]\}$ .
- (3) We next make two calls to an NP oracle:
- (a) We test whether  $s \in Q(D)$ . If it is, then we reject the current guess. Otherwise we continue. Testing whether a tuple belongs to the query result of a UCQ query on a given instance is known to be in NP [Chandra and Merlin 1977].
  - (b) Let  $D' = D \cup \Delta$ . We check whether  $D'$  and  $D_m$  do not satisfy one of the CCs in  $V$ , a check that can also be done in NP. If  $D'$  indeed violates one of the CCs, then the current guess is again rejected. Otherwise,  $D'$  is a counterexample for the completeness of  $D$  for  $Q$  relative to  $(D_m, V)$ , and the algorithm returns “yes”.

We next verify the correctness of the algorithm. Clearly, the algorithm returns “yes” if a counterexample for the completeness of  $D$  for  $Q$  has been found. Indeed, the counterexample is  $D \cup \Delta$ , where  $\Delta = \nu(T_i)$ , and  $\nu$  and  $T_i$  are the guesses that lead to a successful run of the algorithm. Conversely, we show that if  $D$  is incomplete for  $Q$  relative to  $(D_m, V)$ , then the algorithm returns “yes”. Indeed, by Corollary 3.5, if  $D$  is incomplete, then there exist a *valid* valuation  $\mu = (\mu_1, \dots, \mu_k)$  and  $i \in [1, k]$ , such that  $(D \cup \Delta, D_m) \models V$  and  $\mu_i(u_i) \notin Q(D)$ . Such a valuation  $\mu_i$  is indeed one that can be guessed by the algorithm. That is, the algorithm is able to find a counterexample, as desired.

(4) *When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ .* The  $\Pi_2^p$  algorithm given above can be easily modified for this case. A query  $Q$  in  $\exists\text{FO}^+$  is equivalent to a possibly exponentially long union of CQ queries. Therefore, an unfolding of the query will bring us beyond  $\Pi_2^p$ . However, we can avoid unfolding  $Q$  by (i) first guessing the disjunctions in  $Q$ ; and (ii) running the  $\Pi_2^p$  algorithm described above for the tableau corresponding the CQ query that results from this choice of disjunctions. Along the same lines, testing whether a tuple is in the answer to a query in a database can be conducted in NP when the query is in  $\exists\text{FO}^+$  (step (2) in the algorithm). Putting these together, we have a  $\Pi_2^p$  algorithm for checking  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ .

This completes the proof for the  $\Pi_2^p$  upper bound for cases (1)–(4).  $\square$

In practice, master data  $D_m$  and containment constraints  $V$  are often predefined and fixed, and only databases and user queries vary.

One might be tempted to think that fixed  $D_m$  and  $V$  would lower the complexity bounds. Unfortunately, the next result tells us that the lower bound of Theorem 3.6 remains unchanged when  $D_m$  and  $V$  are fixed, even when  $V$  is a fixed set of INDs.

**Corollary 3.7:**  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  remains  $\Pi_2^p$ -complete when master data  $D_m$  and the set  $V$  of containment constraints are fixed, and when (a)  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ, UCQ or  $\exists\text{FO}^+$ ; (b)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ; (c)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ; or

(d)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ . □

PROOF. We show that  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  remains  $\Pi_2^p$ -complete for all the cases considered in Theorem 3.6 when  $V$  and  $D_m$  are fixed. Indeed, the upper bound of Theorem 3.6 carries over to fixed  $D_m$  and  $V$ . For the lower bound, it suffices to observe that the  $\Pi_2^p$ -hardness proof of Theorem 3.6 only uses a fixed set  $V$  of INDs and fixed master data  $D_m$ , when  $Q$  is a CQ query. □

We have also seen from Theorem 3.1 that the problem remains undecidable for queries in FO or FP when  $D_m$  and  $V$  are fixed. Putting these together, we can conclude that fixed  $D_m$  and  $V$  do not lower the complexity of  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ . In contrast, as will be seen in the next section, fixed  $D_m$  and  $V$  simplify the analysis of  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$ , the problem for deciding whether a query is relatively complete.

#### 4. DETERMINING RELATIVELY COMPLETE QUERIES

In this section we investigate  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$ , the relatively complete query problem. Given a query  $Q$  in  $\mathcal{L}_Q$ , master data  $D_m$  and a set  $V$  of CCs in  $\mathcal{L}_C$ , we want to decide whether there exists a database  $D$  that is complete for  $Q$  relative to  $(D_m, V)$ , *i.e.*, whether  $\text{RCQ}(Q, D_m, V)$  is nonempty.

We first show the undecidability of  $\text{RCQ}(Q, D_m, V)$  for FO and FP. We then characterize relatively complete queries in CQ or UCQ, when  $\mathcal{L}_C$  ranges from INDs to UCQ. Based on the characterization, we provide matching lower and upper bounds for  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  range over CQ, UCQ and  $\exists\text{FO}^+$ . Compared to  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ , the complexity bounds of  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  are relatively more diverse; moreover, fixed master data and containment constraints simplify the analysis of relatively complete queries, to some extent.

##### 4.1 The Undecidability of RCQP for FO and FP

Recall from Theorem 3.1 that it is undecidable to determine whether a database is in  $\text{RCQ}(Q, D_m, V)$  when either  $\mathcal{L}_Q$  or  $\mathcal{L}_C$  is FO or FP. It is no better for  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$ : in these settings  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is also undecidable. Moreover, the undecidability is rather robust: the problem is already beyond reach in practice when master data and containment constraints are predefined and fixed.

**Theorem 4.1:**  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable when

- (1)  $\mathcal{L}_Q$  is FO and  $\mathcal{L}_C$  consists of fixed queries in FO,
- (2)  $\mathcal{L}_C$  is FO and  $\mathcal{L}_Q$  is CQ,
- (3)  $\mathcal{L}_Q$  is FP and  $\mathcal{L}_C$  consists of fixed queries in FP, or
- (4)  $\mathcal{L}_C$  is FP and  $\mathcal{L}_Q$  is CQ.

When  $\mathcal{L}_Q$  is FO or FP, it remains undecidable for fixed master data and fixed containment constraints. □

PROOF. When  $\mathcal{L}_Q$  is FO, the proof is not as simple as one might have expected. The undecidability for case (1) is verified by reduction from the emptiness problem for 2-head DFA. Along the same lines we also show the undecidability when  $\mathcal{L}_Q$  is FP (case (3)) and when  $\mathcal{L}_C$  is FP (case (4)). The proof is easier for case (2), when  $\mathcal{L}_C$  is FO; it is by reduction from the satisfiability problem for FO queries.

(1) When  $\mathcal{L}_Q$  is FO and  $\mathcal{L}_C$  consists of fixed FO queries. Recall the emptiness problem for 2-head DFA's from the proof of Theorem 3.1. Given a 2-head DFA  $\mathcal{A}$ , we define relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , master data instance  $D_m$  of  $\mathcal{R}_m$ , a set  $V$  of fixed CCs, and an FO query  $Q$  over  $\mathcal{R}$  such that  $\text{RCQ}(Q, D_m, V) \neq \emptyset$  iff  $L(\mathcal{A}) \neq \emptyset$ .

(a) The relational schema  $\mathcal{R}$  consists of five relation schemas. We have seen four of them in the proof Theorem 3.1 (4), namely, two unary relations  $P(A)$  and  $\bar{P}(A)$ , a binary relation  $F(A_1, A_2)$ , and a 6-ary relation  $R_\Delta(x, y, z, x', y', z')$ . Intuitively, an instance  $D = (I_P, I_{\bar{P}}, I_F)$  of  $(P, \bar{P}, F)$  is to represent a string  $w$  such that elements in  $I_P$  denote the positions in  $w$  where a “1” occurs, and  $I_{\bar{P}}$  keeps track of those positions in  $w$  that are “0”. The relation  $I_F$  encodes a successor relation over these positions. We use instances  $I_\Delta$  of  $R_\Delta$  to encode all valid transitions  $\delta \in \Delta$  of the transition function  $\Delta$  of  $\mathcal{A}$ . In addition to these, we use another 6-ary relations  $R_{\Delta^*}(x, y, z, x', y', z')$  whose instances  $I_{\Delta^*}$  are to encode the transitive closure of the instance of  $R_\Delta$ . We define  $\mathcal{R}_m = (R_1^m)$ , where  $R_1^m$  is a unary relation.

(b) We define the master data instance as  $D_m = (I_1^m = \emptyset)$ .

(c) The set  $V$  consists of the following CCs, all expressible as fixed CQ or FO queries:

- $V_1 : \exists x(P(x) \wedge \bar{P}(x)) \subseteq \emptyset$ , ensuring that no 0 and 1 appear in the same positions in the input string;
- $V_2 : \exists x \exists y \exists z (F(x, y) \wedge F(x, z) \wedge y \neq z) \subseteq \emptyset$ , assuring that instances of  $F$  are functions;
- $V_3 : \exists x \exists y (F(x, x) \wedge F(y, y) \wedge x \neq y) \subseteq \emptyset$ , asserting that there exists at most one tuple of the form  $(k, k)$  in an instance of  $F$ ;
- $V_4$  states that the first three attributes in  $R_\Delta$  are a key for the relation;  
Let  $\bar{u} = (x, y, z)$ ,  $\bar{v} = (x', y', z')$  and  $\bar{w} = (x'', y'', z'')$ . Then,
- $V_5 : (R_\Delta(\bar{u}, \bar{v}) \vee \exists \bar{w} (R_\Delta(\bar{u}, \bar{w}) \wedge R_{\Delta^*}(\bar{w}, \bar{v}))) \wedge \neg R_{\Delta^*}(\bar{u}, \bar{v}) \subseteq \emptyset$ ; and
- $V_6 : (R_{\Delta^*}(\bar{u}, \bar{v}) \wedge \neg (R_\Delta(\bar{u}, \bar{v}) \vee \exists \bar{w} (R_\Delta(\bar{u}, \bar{w}) \wedge R_{\Delta^*}(\bar{w}, \bar{v})))) \subseteq \emptyset$ .

Intuitively,  $V_5$  and  $V_6$  enforce  $I_{\Delta^*}$  to be the transitive closure of  $I_\Delta$ .

(d) The FO query  $Q$  expresses the following. Let  $D = (I_P, I_{\bar{P}}, I_F, I_\Delta, I_{\Delta^*})$  be an instance of  $\mathcal{R}$ . Then  $Q(D) = I_\Delta$  or  $Q(D) = \text{true}$  depending on the conditions below:

- (a) If  $D \not\models \exists x F(0, x)$  then  $Q(D) = I_\Delta$ .
- (b) If  $D \not\models \exists x F(x, x)$  then  $Q(D) = I_\Delta$ .
- (c) If there exists a  $\delta \in \Delta$ ,  $\delta = (q, \text{in}_1, \text{in}_2) \rightarrow (q', \text{move}_1, \text{move}_2)$  such that  $D \not\models \varphi_\delta(x, y, z, x', y', z')$ , then  $Q(D) = I_\Delta$ , where  $\varphi_\delta$  is  $(x = q \wedge x' = q' \wedge \alpha_1(y) \wedge \alpha_2(z) \wedge \beta_1(y, y') \wedge \beta_2(z, z'))$ . Here  $\alpha_i(x) = \exists y (F(x, y) \wedge x \neq y \wedge P(x))$  if  $\text{in}_i = 1$ ;  $\alpha_i(x) = \exists y (F(x, y) \wedge x \neq y \wedge \bar{P}(x))$  if  $\text{in}_i = 0$ ; and  $\alpha_i(x) = F(x, x)$  if  $\text{in}_i = \varepsilon$ . Moreover,  $\beta_i(x, y) = F(x, y)$  if  $\text{move}_i = +1$  and  $\beta_i(x, y) = (x = y)$  if  $\text{move}_i = 0$ .
- (d) Finally, if there does not exist a tuple  $(q, 0, 0, q_{acc}, x, y)$  in  $I_{\Delta^*}$ , then  $Q(D) = I_\Delta$ , otherwise  $Q(D) = \text{true}$ .

We now show that  $L(\mathcal{A}) \neq \emptyset$  iff there exists a database complete for  $Q$  relative to  $(D_m, V)$ . First suppose that  $L(\mathcal{A}) \neq \emptyset$ . We show that  $\text{RCQ}(Q, D_m, V)$  is nonempty by constructing an instance  $D$  of  $\mathcal{R}$  that is complete for  $Q$  relative to  $(D_m, V)$ . Let  $I_P, I_{\bar{P}}$  and  $I_F$  be an encoding of an input string that  $\mathcal{A}$  accepts. Furthermore, let

$I_\Delta$  consist of all  $\varphi_\delta$ 's given above, and let  $I_{\Delta^*}$  be the transitive closure of  $I_\Delta$ . We define  $D = (I_P, I_{\bar{P}}, I_F, I_\Delta, I_{\Delta^*})$ . Since the run of  $\mathcal{A}$  on input  $I_P, I_{\bar{P}}$  and  $I_F$  is accepting, there exists a tuple  $(q, 0, 0, q_{acc}, x, y)$  in  $I_{\Delta^*}$ . Hence  $Q(D) = \text{true}$ . Consider any  $D' = (I'_P, I'_{\bar{P}}, I'_F, I'_\Delta, I'_{\Delta^*})$  such that  $D \subseteq D'$  and  $(D', D_m) \models V$ . From this and the definition of  $Q$ , it readily follows that  $Q(D') = \text{true}$  as well. Hence  $D$  is indeed complete for  $Q$  relative to  $(D_m, V)$ .

Conversely, suppose that  $L(\mathcal{A}) = \emptyset$ . We show that there cannot exist a  $D = (I_P, I_{\bar{P}}, I_F, I_\Delta, I_{\Delta^*})$  that is complete for  $Q$  relative to  $(D_m, V)$ . Clearly, the only scenario that can lead to a complete instance is when  $Q(D) = \text{true}$ . However, this happens only when (i)  $I_P, I_{\bar{P}}$  and  $I_F$  correctly encode an input, (ii)  $I_\Delta$  contains all valid transitions in  $\Delta$  of  $\mathcal{A}$ , and (iii) the transitive closure  $I_{\Delta^*}$  of  $I_\Delta$  contains a tuple of the form  $(q, 0, 0, q_{acc}, x, y)$ . Observe that the first three attributes of  $R_\Delta$  are a key for the relation  $R_\Delta$ , and that  $I_{\Delta^*}$  is the transitive closure of  $I_\Delta$ . Taken together, these assure the existence of a successful run of  $\mathcal{A}$ , contradicting our assumption. Indeed, given  $(q, 0, 0, q_{acc}, x, y)$  we know that there exists a unique  $(q_1, x_1, y_1)$  such that  $(q, 0, 0, q_1, x_1, y_1) \in I_\Delta$  and  $(q_1, x_1, y, q_{acc}, x, y) \in I_{\Delta^*}$ . Furthermore, since  $I_\Delta$  contains the valid transitions of  $\Delta$  and by the key constraint,  $(q, 0, 0, q_1, x_1, y_1)$  corresponds to a valid transition in  $\Delta$  as well. We can therefore keep unfolding  $(q_1, x_1, y, q_{acc}, x, y)$ , which leads to a sequence of valid transitions of  $\mathcal{A}$  from  $(q, 0, 0)$  to  $(q_{acc}, x, y)$ . This contradicts the assumption that  $L(\mathcal{A}) = \emptyset$ .

(2) When  $\mathcal{L}_C$  is FO and  $\mathcal{L}_Q$  is CQ. We show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable by reduction from the satisfiability problem for FO. Given an FO query  $q$  over a relational schema  $\mathcal{R} = (R_1, \dots, R_n)$ , we derive a boolean query  $q'$  from  $q$ , defined by  $q'(D) = \{\emptyset\}$  if  $q(D) \neq \emptyset$  or  $D$  is empty, and  $q'(D) = \emptyset$  otherwise. The schema of the input database and master data are defined to be, respectively,  $\mathcal{R}' = (\mathcal{R}, R_u)$  and  $\mathcal{R}_m = (R_1^m)$  for some unary relation schemas  $R_u$  and  $R_1^m$ . Moreover, let  $D_m = (I_1^m = \emptyset)$ . We define  $V$  to consist of a single CC:  $\{\emptyset\} \setminus q' \subseteq \emptyset$ . Clearly, for each instance  $D' = (I'_1, \dots, I'_n, I'_u)$  of  $\mathcal{R}'$  we have that  $(D', D_m) \models V$  iff  $q(I'_1, \dots, I'_n)$  is nonempty or  $(I'_1, \dots, I'_n) = (\emptyset, \dots, \emptyset)$ . Finally, we define a query  $Q$  over  $\mathcal{R}'$  as  $Q(I'_1, \dots, I'_n, I'_u) = Q_1(I'_1, \dots, I'_n) \times I'_u$ , where  $Q_1(I'_1, \dots, I'_n) = \{\{1\}\}$  if  $(I'_1, \dots, I'_n) \neq (\emptyset, \dots, \emptyset)$  and  $Q_1(I'_1, \dots, I'_n) = \emptyset$  otherwise.

We show that  $\text{RCQ}(Q, D_m, V)$  is nonempty iff  $q$  is not satisfiable. Suppose first that  $q$  is not satisfiable. Then  $(D' = (I'_1, \dots, I'_n, I'_u), D_m) \models V$  iff  $(I'_1, \dots, I'_n) = (\emptyset, \dots, \emptyset)$ . We define  $D = (\emptyset, \dots, \emptyset, I_u)$  for an arbitrary instance  $I_u$  of  $R_u$ . Then  $Q(D) = \emptyset$ . Clearly  $D$  is complete for  $Q$  relative to  $(D_m, V)$ . Conversely, if  $q$  is satisfiable then there exists a  $D = (I_1, \dots, I_n, I_u)$  such that  $q(I_1, \dots, I_n) \neq \emptyset$ . Observe that it suffices to consider only such  $D$ 's since those are the only ones that satisfy  $V$  together with  $D_m$ . However,  $Q(D) = \{\{1\}\} \times I_u$ , which shows that  $D$  cannot be complete for  $Q$ . Indeed, for each  $D' = (I_1, \dots, I_n, I'_u)$  with  $I_u \subset I'_u$ ,  $D$  and  $D_m$  satisfy  $V$  but  $Q(D) \neq Q(D')$ . Hence  $\text{RCQ}(Q, D_m, V)$  is empty.

(3) When  $\mathcal{L}_Q$  is FP and  $\mathcal{L}_C$  consists of fixed FP queries. We show the undecidability of  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  by reduction from the emptiness problem for 2-head DFA's. The proof is referred to the Appendix.

(4) When  $\mathcal{L}_C$  is FP and  $\mathcal{L}_Q$  is CQ. We show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is undecidable

again by reduction from the emptiness problem for 2-head DFA's. The proof is referred to the Appendix.

The proofs only use fixed  $(D_m, V)$  when  $\mathcal{L}_Q$  is FO or FP, and thus verify the undecidability for fixed  $D_m$  and  $V$ .  $\square$

#### 4.2 Characterizations of Relatively Complete Queries in CQ

The undecidability results suggest that we consider CQ, UCQ and  $\exists\text{FO}^+$  for  $\mathcal{L}_Q$  and  $\mathcal{L}_C$ . To understand what makes a query  $Q$  allow a relatively complete database for given master data  $D_m$  and a set  $V$  of CCs, we provide sufficient and necessary conditions for  $\text{RCQ}(Q, D_m, V)$  to be nonempty.

We first present conditions for  $Q$  and  $V$  in CQ. We then give a syntactic characterization for relative complete CQ queries  $Q$  when  $\mathcal{L}_C$  consists of INDs. Finally, we extend the conditions for CQ to characterize relatively complete UCQ queries when  $\mathcal{L}_C$  is UCQ. The conditions can be extended to queries and CCs in  $\exists\text{FO}^+$ .

**When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ.** To get insight into the conditions, let us first look at some example queries, complete or incomplete.

**Example 4.1:** Consider the schema  $\text{Supt}(\text{eid}, \text{dept}, \text{cid})$  of Example 1.1. An FD on  $\text{Supt}$  is  $\text{eid} \rightarrow \text{dept}$ , *i.e.*, each employee works in at most one department. The FD can be expressed as a CC  $\phi_3$  in CQ. Consider a query  $Q_4$  in CQ that is to find all  $\text{Supt}$  tuples  $t$  such that  $t[\text{eid}] = 'e_0'$  and  $t[\text{dept}] = 'd_0'$ . Let master data  $D_m$  be an empty relation. Then  $Q_4$  is relatively complete. Indeed, there exists a database  $D^-$  complete for  $Q_4$  relative to  $(D_m, \{\phi_3\})$ , where  $D^-$  consists of a single tuple  $t_- = (e_0, d', c)$ ,  $d' \neq d_0$ . Note that  $Q_4(D^-) = \emptyset$ . Furthermore, for each set  $\Delta$  of tuples such that  $Q_4(\Delta)$  is nonempty,  $D^-$  prevents  $\Delta$  from being added to it since otherwise  $D^- \cup \Delta$  violates the CC  $\phi_3$ .

Now consider the query  $Q_2$  of Example 1.1, which is to find all  $\text{Supt}$  tuples  $t$  with  $t[\text{eid}] = 'e_0'$ . Assume that  $\text{dom}(\text{cid})$  is infinite. Then  $Q_2$  is *not* complete *w.r.t.*  $D_m$  and  $\phi_3$ . Indeed, no matter what database  $D$  we consider, we can always add a new tuple  $t'$  to  $D$  such that  $t'[\text{cid}]$  is a value not in  $D$ , and  $Q_3(D) \neq Q_3(D \cup \{t'\})$ . In contrast, if the FD  $\text{eid} \rightarrow \text{dept}, \text{cid}$  of Example 3.1 (expressed as a set  $\Sigma_2$  of CCs) is in place, then  $Q_2$  is relatively complete. Indeed, a database complete for  $Q_3$  is  $D^+$ , which consists of  $t_+ = (e_0, d_0, c_0)$ . For each set  $\Delta$  of tuples, if  $Q_2(\Delta) \neq \emptyset$  and  $(D^+ \cup \Delta, D_m) \models \Sigma_2$ , then for each  $t' \in \Delta$ ,  $D^+$  enforces  $t'[\text{eid}]$  to take  $'e_0'$  as its value. That is, the values of  $t'[\text{eid}]$  are bounded, and  $Q_2(D^+ \cup \Delta) = Q_2(D^+) = \{t_+\}$ .  $\square$

As suggested by the example, a query  $Q$  is relatively complete iff one of the following two conditions holds. (a) There exists a set  $D^-$  of tuples such that  $(D^-, D_m) \models V$ ,  $Q(D^-) = \emptyset$  and moreover,  $D^-$  prevents those tuples  $\Delta$  from being added to  $D^-$  if  $Q(\Delta)$  is nonempty. That is, there exist no tuples  $\Delta$  such that both  $(D^- \cup \Delta, D_m) \models V$  and  $Q(\Delta) \neq \emptyset$ . (b) There exists a set  $D^+$  of tuples such that  $Q(D^+) \neq \emptyset$ ,  $(D^+, D_m) \models V$ , and moreover,  $D^+$  “bounds” all those variables  $y$  in  $Q$  with an infinite domain, via  $D_m$  and  $V$ . That is, for each such  $y$  and each set of tuples  $\Delta$ , if  $Q(\Delta)$  is nonempty, then either  $(D^+ \cup \Delta, D_m) \not\models V$  or  $Q(\Delta) \subseteq Q(D^+)$ , where  $y$  is instantiated with a value that is in  $D_m$ ,  $D^+$  or is a constant in  $Q$ .

To formalize the intuition, we use the following notations.

(a) We revise the notion of **Adom** given in Section 3.2 such that it consists of all the constants that are in  $D_m, V, Q$  or **New**. As in Section 3.2, we represent CQ query  $Q$  as a tableau query  $(T_Q, u_Q)$ , and define valid valuations of  $T_Q$ .

The domain of a variable  $y$  in  $T_Q$ , denoted by  $\text{dom}(y)$ , is said to be *finite* if  $y$  appears in some column  $A$  in  $T_Q$  such that  $\text{dom}(A)$  is  $\mathbf{d}_f$ , and it is *infinite* otherwise.

(b) Consider a set  $V$  consisting of CCs  $q_i \subseteq p_i$  for  $i \in [1, n]$ , where  $q_i$  is a CQ query. We represent  $q_i$  as a tableau query  $(T_i, u_i)$ . A *valuation*  $\nu$  of  $V$  is  $(\nu_1, \dots, \nu_n)$ , where  $\nu_i$  is a valuation of variables in a *subset* of tuple templates in  $T_i$ .

In contrast to valuations of  $T_Q$ , a valuation  $\nu_i$  is “partial”: it instantiates a subset of  $T_i$  in CCs. To see the need for this, observe that the FD  $\text{eid} \rightarrow \text{dept}$ , when expressed as CC  $\phi_3 : q \subseteq \emptyset$  with  $q = (T, u)$ ,  $T$  consists of two tuple templates (see, e.g.,  $\phi_1$  of Example 2.1). As we have seen in Example 4.1, it is sufficient to instantiate one of the two tuple templates to make  $D^-$ . This is also necessary: if both templates are instantiated then these tuples warrant to violate the CC  $\phi_3$ .

We use  $D_\nu$  to denote  $\bigcup_{i \in [1, n]} \nu_i(T_i)$ . For a set  $\mathcal{V}$  of valuations of  $V$ , we use  $D_\mathcal{V}$  to denote  $\bigcup_{\nu \in \mathcal{V}} D_\nu$ . In particular, when  $\mathcal{V}$  is empty, so is  $D_\mathcal{V}$ .

(c) A variable  $y$  in  $u_Q$  is said to be *bounded by*  $\mathcal{V}$  *w.r.t.* a valuation  $\mu$  of  $T_Q$  if there exist  $\nu \in \mathcal{V}$  and  $j \in [1, n]$  such that  $\mu(y)$  appears in  $\nu_j(u_j)$ , i.e.,  $\mu(y) = \nu_j(z)$  for some  $z$  in  $u_j$ .

We next identify conditions for  $\text{RCQ}(Q, D_m, V)$  to be nonempty, based on a notion of bounded queries.

A CQ query  $Q = (T_Q, u_Q)$  is said to be *bounded by*  $(D_m, V)$  iff either

(E1) all variables in  $u_Q$  have a finite domain, or

(E2) there exists a set  $\mathcal{V}$  of valuations of  $V$  such that  $(D_\mathcal{V}, D_m) \models V$ , and moreover, for each valid valuation  $\mu$  of  $T_Q$ , if  $(D_\mathcal{V} \cup \mu(T_Q), D_m) \models V$  then each variable  $y$  with an infinite domain in  $u_Q$  is bounded by  $\mathcal{V}$  *w.r.t.*  $\mu$ .

Intuitively, when all the variables in  $u_Q$  have a finite domain (condition E1),  $Q$  is trivially relatively complete *w.r.t.*  $(D_m, V)$ . Suppose that some variables have an infinite domain, then the condition E2 allows us to determine the existence of a set  $D^-$  or  $D^+$  with the properties mentioned above. More specifically, when there exists a set  $\mathcal{V}$  of valuations of  $V$  satisfying the condition E2, then  $D^-$  (resp.  $D^+$ ) can be constructed from  $D_\mathcal{V}$  when  $Q(D_\mathcal{V}) = \emptyset$  (resp.  $Q(D_\mathcal{V}) \neq \emptyset$ ).

Bounded queries indeed characterize relatively complete queries in CQ.

**Proposition 4.2:** For each CQ query  $Q$ , master data  $D_m$ , and each set  $V$  of CCs in CQ,  $\text{RCQ}(Q, D_m, V)$  is nonempty iff  $Q$  is bounded by  $(D_m, V)$ , i.e.,  $Q$  satisfies either the condition E1 or E2.  $\square$

**PROOF.** We first show that if the condition E1 or E2 holds, then  $\text{RCQ}(Q, D_m, V)$  is nonempty. We consider two cases: (a) when E1 holds and (b) when E2 holds.

(a) When E1 holds. Consider a maximal collection  $\Theta$  of valuations of  $T_Q$ , not necessarily drawing values from **Adom**, such that  $(D, D_m) \models V$ , where  $D = \bigcup_{\mu \in \Theta} \mu(T_Q)$ . Obviously the database  $D$  is finite since there are finitely many valuations of  $T_Q$  given the condition E1. Furthermore,  $D$  is relatively complete: for each valuation  $\mu'$  of  $T_Q$ , either  $\mu' \in \Theta$  or  $(D \cup \mu'(T_Q), D_m) \not\models V$ . Thus  $\text{RCQ}(Q, D_m, V)$  is nonempty.

(b) When E2 holds, *i.e.*, there exists a set  $D_{\mathcal{V}}$  of tuples as specified by E2. We expand it with all constant tuples in  $T_Q$ , *i.e.*, tuple templates in  $T_Q$  that do not contain any variable. We denote the extended set also by  $D_{\mathcal{V}}$ . We show that  $D_{\mathcal{V}}$  is relatively complete. Assume by contradiction that there exists a valuation  $\mu'$  of  $T_Q$  such that  $Q(\mu'(T_Q)) \neq \emptyset$  and  $(D_{\mathcal{V}} \cup \mu'(T_Q), D_m) \models V$ , but  $Q(\mu'(T_Q)) \notin Q(D_{\mathcal{V}})$ . Then there exists a variable  $y$  in  $u_Q$  such that  $\text{dom}(y)$  is infinite and  $\mu'(y) \notin \text{Adom}$ , by the definition of  $\text{Adom}$  and  $D_{\mathcal{V}}$ . Define a valid valuation  $\mu$  of  $T_Q$  such that for each variable  $x$  of  $Q$ ,  $\mu(x) = \mu'(x)$  if  $\mu'(x) \in \text{Adom}$  and  $\mu(x)$  is a distinct value in  $\text{New}$  otherwise. By the choice of values in  $\text{New}$ , we have that  $(D_{\mathcal{V}} \cup \mu(T_Q), D_m) \models V$ . By E2, there must exist a CC  $q_i \subseteq p_i$  in  $V$ , where  $q_i = (T_i, u_i)$ , and a valuation  $\nu_i$  of  $T_i$  in some valuation  $\nu$  of  $\mathcal{V}$ , such that  $\mu(y)$  appears in  $\nu_i(u_i)$ . By the definition of  $\text{New}$ , the values of  $y$  must be constrained by the values in  $D_m$ . In other words, if  $\mu'(y) \notin \text{Adom}$  then either  $(D_{\mathcal{V}} \cup \mu'(T_Q), D_m) \not\models V$  or  $Q(\mu'(T_Q)) \in Q(D_{\mathcal{V}})$ , contradicting the assumption. Hence  $D_{\mathcal{V}}$  is in  $\text{RCQ}(Q, D_m, V)$ .

Conversely, suppose that  $\text{RCQ}(Q, D_m, V)$  contains a database  $D$ . We consider two cases: (a) when  $V = \emptyset$  and (b) when  $V \neq \emptyset$ .

(a) When  $V = \emptyset$ . We show that condition E1 must hold. Assume by contradiction that there exists a variable  $y$  in  $u_Q$  such that  $\text{dom}(y)$  is infinite. Since we consider satisfiable CQ queries, there exists a valuation  $\mu'$  of  $T_Q$  such that  $Q(\mu'(T_Q))$  is nonempty. Let  $c$  be a value in  $\text{dom}(y)$  such that  $c$  is in none of  $D, D_m, Q$  and  $V$ . Define  $\mu'$  such that  $\mu'(y) = c$  and  $\mu'(x) = \mu(x)$  for  $x \neq y$ . Then  $(D \cup \mu'(T_Q), D_m) \models V$  since  $V$  is empty. Furthermore,  $Q(\mu'(T_Q)) \in Q(D \cup \mu'(T_Q))$  by the monotonicity of CQ queries, but  $Q(\mu'(T_Q)) \notin Q(D)$ . That is,  $Q(D) \neq Q(D \cup \mu'(T_Q))$ , contradicting the assumption that  $D$  is relatively complete.

(b) When  $V \neq \emptyset$ . Assume that neither condition E1 nor E2 holds. Define  $\mathcal{V}$  to be a maximal set of valuations of  $V$  such that  $(D \cup D_{\mathcal{V}}, D_m) \models V$ . Then there is a valid valuation  $\mu$  of  $T_Q$  such that  $(D_{\mathcal{V}} \cup \mu(T_Q), D_m) \models V$ , but some variable  $y$  in  $u_Q$  with an infinite domain is not bounded by  $\mathcal{V}$  *w.r.t.*  $\mu$ . It suffices to show that  $(D \cup \mu(T_Q), D_m) \models V$ . For if it holds, one can pick a value  $c \in \text{dom}(y)$  that is in none of  $D, D_m, Q$  and  $V$ . Define  $\mu'(y) = c$  and  $\mu'(x) = \mu(x)$  for  $x \neq y$ . Then  $(D \cup \mu'(T_Q), D_m) \models V$  since  $y$  is not bounded by  $V$  and  $D_m$ , but  $Q(D) \neq Q(D \cup \mu'(T_Q))$ . This contradicts the assumption that  $D$  is relatively complete.

Assume by contradiction that  $(D \cup \mu(T_Q), D_m) \not\models \phi$ , where  $\phi_i$  is a CC  $q_i \subseteq p_i$  in  $V$ . Let  $q_i$  be represented as  $(T_i, u_i)$ . Since  $(D_{\mathcal{V}} \cup \mu(T_Q), D_m) \models V$ , by the monotonicity of CQ queries we have that  $(\mu(T_Q), D_m) \models V$ . In addition,  $(D, D_m) \models V$  since  $D$  is assumed to be in  $\text{RCQ}(Q, D_m, V)$ . Thus there must exist a nonempty subset  $\Delta_1$  of  $D$ , a nonempty subset  $\Delta_2$  of  $\mu(T_Q)$  and a valuation  $\rho'$  of variables in  $T_i$  such that  $\rho'(T_i) = \Delta_1 \cup \Delta_2$ ,  $(\Delta_1 \cup \Delta_2, D_m) \not\models \phi$ , but  $(\Delta_1 \cup D, D_m) \models V$ . In addition, since  $\mu$  draws values from  $\text{Adom}$  and  $(D \cup D_{\mathcal{V}}, D_m) \models V$ , one can get  $\Delta'_1$  from  $\Delta_1$  such that all values in  $\Delta'_1$  are in  $\text{Adom}$ ,  $(\Delta'_1 \cup \Delta_2, D_m) \not\models \phi$ , but  $(\Delta'_1 \cup D, D_m) \models V$ . From  $\Delta'_1$  and  $\nu_\phi$  one can readily induce a subset  $T'_i$  of  $T_i$  and a mapping  $\rho$  from variables in  $T'_i$  to  $\text{Adom}$  such that  $\rho(T'_i) = \Delta'_1$ . Define a valuation  $\nu = (\nu_1, \dots, \nu_n)$  of  $V$  such that  $\nu_i$  is  $\rho$  and  $\nu_j$  is an empty mapping for  $j \neq i$ . Let  $\mathcal{V}' = \mathcal{V} \cup \{\nu\}$ . Then  $(D \cup D_{\mathcal{V}'}, D_m) \models V$ . This contradicts the assumption that  $\mathcal{V}$  is maximal. Thus  $(D \cup \mu(T_Q), D_m) \models \phi$ .  $\square$

Observe that the size of each set  $\mathcal{V}$  of valuations of  $V$  is at most exponential in the sizes of  $Q, V$  and  $D_m$ , and that each valid valuation  $\mu$  of  $T_Q$  is no larger than  $Q$ . It is based on this small model property that we give the complexity bounds for RCQP(CQ, CQ) in Section 4.3.

The characterization can be equivalently expressed as follows. For a database  $D$  and a variable  $y$  in  $u_Q$ , let  $\text{val}(D, y)$  denote the set of  $\mu(y)$ 's when  $\mu$  ranges over all valuations of  $T_Q$  such that  $Q(\mu(T_Q)) \neq \emptyset$  and  $(D \cup \mu(T_Q), D_m) \models V$  ( $\mu(y)$  is not necessarily in  $\text{Adom}$ ). Then  $\text{RCQ}(Q, D_m, V)$  is nonempty iff there exists a set  $\mathcal{V}$  of valuations of  $V$  such that  $(D_{\mathcal{V}}, D_m) \models V$  and  $\text{val}(D_{\mathcal{V}}, y)$  is *finite* for all  $y$  in  $u_Q$ .

**When  $\mathcal{L}_C$  is the class of INDs.** In this setting, there is a syntactic characterization for relatively complete queries.

A CQ query  $Q = (T_Q, u_Q)$  is *bounded by*  $(D_m, V)$  if for all variables  $y$  in  $u_Q$ ,

(E3) either  $\text{dom}(y)$  is finite, or

(E4) there exists an IND  $\pi_{(A, \dots)} \subseteq p$  in  $V$  such that  $y$  appears in column  $A$  in  $T_Q$ , where  $\pi$  is the projection operator.

**Proposition 4.3:** For each CQ query  $Q = (T_Q, u_Q)$ , master data  $D_m$ , and each set  $V$  of INDs,  $\text{RCQ}(Q, D_m, V)$  is nonempty iff either  $Q$  is bounded by  $(D_m, V)$ , *i.e.*,  $Q$  satisfies the condition E3 or E4, or there exists no valid valuation  $\mu$  of  $T_Q$  such that  $(\mu(T_Q), D_m) \models V$ .  $\square$

**PROOF.** First assume that  $Q$  is bounded by  $(D_m, V)$ . We construct a relatively complete database  $D$ . If for all valid valuations  $\mu$  of  $T_Q$ ,  $(\mu(T_Q), D_m) \not\models V$ , then obviously the empty database  $D$  is in  $\text{RCQ}(Q, D_m, V)$ . Below we assume that there exists a valid valuation  $\mu$  of  $T_Q$  such that  $(\mu(T_Q), D_m) \models V$ . We define  $D$  to be a minimal collection of tuples such that for each variable  $y$  in  $u_Q$  and each  $c \in \text{adom}(y)$ , if there exists a valid valuation  $\mu$  of  $T_Q$  such that  $\mu(y) = c$  and  $(\mu(T_Q), D_m) \models V$ , then  $D$  includes one of such  $\mu(T_Q)$ 's. Observe the following: (i)  $D$  is finite since it is defined using valid valuations of  $T_Q$  only; (ii)  $(D, D_m) \models V$  since  $V$  is a set of INDs and  $(\mu(T_Q), D_m) \models V$  for each  $\mu(T_Q)$  that is included in  $D$ ; and (iii) for each partially closed extension  $D'$  of  $D$ ,  $Q(D) = Q(D')$ ; indeed, by the definition of  $D$ , for each  $s \in Q(D')$ , there exists a valid valuation  $\mu$  of  $T_Q$  such that  $s \in Q(\mu(T_Q))$  and  $\mu(T_Q) \subseteq D$ . Thus  $D$  is in  $\text{RCQ}(Q, D_m, V)$  and  $\text{RCQ}(Q, D_m, V)$  is nonempty.

Conversely, suppose that there exists a database  $D$  in  $\text{RCQ}(Q, D_m, V)$ . Assume by contradiction that there is a valid valuation  $\mu$  of  $T_Q$  such that  $(\mu(T_Q), D_m) \models V$ , and that  $Q$  is not bounded by  $(D_m, V)$ , *i.e.*, there exists a variable  $y$  in  $T_Q$  such that  $\text{dom}(y)$  is infinite and  $y$  is not constrained by any INDs in  $V$ . Then we can find a constant  $c \in \text{adom}(y)$  that appears in none of  $D, D_m, Q$  and  $V$ . Define a valuation  $\mu'$  of  $T_Q$  such that  $\mu'(y) = c$  and  $\mu'(x) = \mu(x)$  for all variables  $x \neq y$  in  $T_Q$ . Then it is easy to see that  $Q(\mu'(T_Q)) \not\subseteq Q(D')$  and moreover,  $(D \cup \mu'(T_Q), D_m) \models V$  since  $y$  is not constrained by  $V$ . This contradicts the assumption that  $D$  is complete, since  $D' = D \cup \mu'(T_Q)$  is an extension of  $D$ ,  $(D', D_m) \models V$  but  $Q(D) \neq Q(D')$ .  $\square$

**When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ.** A containment constraint in UCQ is of the form

$(q_1 \cup \dots \cup q_m) \subseteq p$ , and is equivalent to a set of CCs in CQ, consisting of  $q_j \subseteq p$  for each  $j \in [1, m]$ . Thus the notions of valuations of  $V$  and  $D_{\mathcal{V}}$  for a set  $\mathcal{V}$  of valuations of  $V$  are also well defined in this setting.

Consider a query  $Q = Q_1 \cup \dots \cup Q_k$  in UCQ, where  $Q_i$  is represented as a tableau query  $(T_i, u_i)$ . We use the notion of valid valuations of  $Q$  given in Section 3.2. We also defined bounded variables defined UCQ queries, similar to their CQ counterpart.

A UCQ query  $Q$  is said to be *bounded by*  $(D_m, V)$  iff for all  $i \in [1, k]$ , either

(E5) all variables in  $u_i$  have a finite domain, or

(E6) there exists a set  $\mathcal{V}$  of valuations of  $V$  such that  $(D_{\mathcal{V}}, D_m) \models V$ , and for each valid valuation  $\mu = (\mu_1, \dots, \mu_k)$  of  $Q$ , if  $(D_{\mathcal{V}} \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$ , then for each variable  $y$  with an infinite domain in  $u_i$ ,  $y$  is bounded by  $\mathcal{V}$  w.r.t.  $\mu$ .

**Corollary 4.4:** For each UCQ query  $Q$ , master data  $D_m$ , and each set  $V$  of CCs in UCQ,  $\text{RCQ}(Q, D_m, V)$  is nonempty iff  $Q$  is bounded by  $(D_m, V)$ , i.e.,  $Q$  satisfies either the condition E5 or or the condition E6.  $\square$

PROOF. Consider a UCQ query  $Q = Q_1 \cup \dots \cup Q_k$ , where  $Q_i$  is represented as  $(T_i, u_i)$ . We first show that if the condition E5 or E6 holds, then  $\text{RCQ}(Q, D_m, V)$  is nonempty. When E5 holds, we can define a database complete for  $Q$  relative to  $(D_m, V)$  along the same lines as in the proof of Proposition 4.2. Suppose now that E6 holds, i.e., there exists  $D_{\mathcal{V}}$  as specified by E6. We expand it by including constant tuples in  $T_i$  for all  $i \in [1, k]$ , and also denote it as  $D_{\mathcal{V}}$ . We show that  $D_{\mathcal{V}}$  is in  $\text{RCQ}(Q, D_m, V)$ . Assume by contradiction that there exists a valuation  $\mu' = (\mu'_1, \dots, \mu'_k)$  such that  $(D_{\mathcal{V}} \cup \bigcup_{i \in [1, k]} \mu'_i(T_i), D_m) \models V$ , but  $Q_i(\mu'_i(T_Q)) \notin Q(D_{\mathcal{V}})$  for some  $i \in [1, k]$ . Then there exists a variable  $y$  in  $u_i$  such that  $\text{dom}(y)$  is infinite and  $\mu'_i(y) \notin \text{Adom}$ . Define a valid valuation  $\mu = (\mu_1, \dots, \mu_k)$  of  $T_Q$  such that for each  $j \in [1, k]$  and each variable  $x$  of  $T_j$ ,  $\mu_j(x) = \mu'_j(x)$  if  $\mu'_j(x) \in \text{Adom}$  and  $\mu_j(x)$  is a distinct value in  $\text{New}$  otherwise. Then similar to the proof of Proposition 4.2, it can be verified that  $(D_{\mathcal{V}} \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$ . In addition, by E6,  $y$  is bounded by  $\mathcal{V}$  w.r.t.  $\mu$ . By the definition of  $\text{New}$ , the values of  $y$  must be bounded by  $D_m$  via  $V$ . This leads to either  $Q_i(\mu'_i(T_Q)) \in Q(D_{\mathcal{V}})$  or  $(D_{\mathcal{V}} \cup \bigcup_{i \in [1, k]} \mu'_i(T_i), D_m) \not\models V$ , contradicting the assumption. Hence  $\text{RCQ}(Q, D_m, V)$  contains  $D_{\mathcal{V}}$  and is nonempty.

Conversely, suppose that there exists a database  $D$  in  $\text{RCQ}(Q, D_m, V)$ . We show that either E5 or E6 must hold. When  $V = \emptyset$ , the argument for Proposition 4.2 suffices to show that E5 must hold, i.e., all variables in  $u_i$  have a finite domain for all  $i \in [1, k]$ . When  $V \neq \emptyset$ , define  $\mathcal{V}$  to be a maximal set of valuations of  $V$  such that  $(D \cup D_{\mathcal{V}}, D_m) \models V$ . Assume by contradiction that there is a valid valuation  $\mu = (\mu_1, \dots, \mu_k)$  of  $T_Q$  such that  $(D_{\mathcal{V}} \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$ , but some variable  $y$  in  $u_i$  with an infinite domain is not bounded by  $\mathcal{V}$  w.r.t.  $\mu$ . Then along the same lines as Proposition 4.2, it can be verified that  $(D \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$ . Since  $\text{adom}(y)$  is infinite, there exists a value  $c$  that is in none of  $D, D_m, Q$  and  $V$ . Define a valuation  $\mu' = (\mu'_1, \dots, \mu'_k)$  such that  $\mu'_i(y) = c$  and  $\mu'_i(x) = \mu_i(x)$  for all variable  $x \neq y$ , and for all  $j \neq i$ ,  $\mu'_j$  is  $\mu_j$ . Then similar to the proof of Proposition 4.2, one can verify that  $(D \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$  but  $Q(D) \neq Q(D \cup \bigcup_{i \in [1, k]} \mu_i(T_i))$ . This contradicts the assumption that  $D$  is relatively complete.  $\square$

### 4.3 The Complexity of RCQP for CQ, UCQ and $\exists\text{FO}^+$

We have seen from Theorem 3.6 that the absence of negation and recursion in  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  simplifies the analysis of  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ . Below we show that this is also the case for  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$ , which is settled in positive in these settings.

In contrast to Theorem 3.6, the complexity bounds for  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  are no longer the same when  $\mathcal{L}_C$  is  $\exists\text{FO}^+$  and when  $\mathcal{L}_C$  is the class of INDs. In addition, when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  becomes NEXPTIME-complete, *i.e.*, the analysis is harder than its  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$  counterpart. On the other hand, when  $\mathcal{L}_Q$  is the class of INDs, the complexity is down to  $\text{coNP}$ -complete, better than its  $\Pi_2^p$ -complete counterpart for  $\text{RCDP}(\mathcal{L}_Q, \mathcal{L}_C)$ . The proofs for the upper bounds below make use of the characterizations of relatively complete queries given above.

**Theorem 4.5:**  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is

- (1)  $\text{coNP}$ -complete when  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ, UCQ or  $\exists\text{FO}^+$ ; and
- (2) NEXPTIME-complete when
  - (a)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ,
  - (b)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ, or
  - (c)  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ . □

PROOF. We prove the complexity for cases (1) and (2.a)–(2.c) one by one.

(1) *When  $\mathcal{L}_C$  is the class of INDs.* We show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\text{coNP}$ -complete when  $\mathcal{L}_C$  consists of INDs and when  $\mathcal{L}_Q$  is CQ, UCQ or  $\exists\text{FO}^+$ .

*Lower bound.* It suffices to show that the problem is  $\text{coNP}$ -hard when  $\mathcal{L}_Q$  is CQ, since the lower bound readily carries over to the case when  $\mathcal{L}_Q$  is UCQ or  $\exists\text{FO}^+$ . The  $\text{coNP}$ -hardness is verified by reduction from 3SAT to the complement of  $\text{RCQP}(\text{CQ}, \text{INDs})$ , where 3SAT is known to be NP-complete (cf. [Garey and Johnson 1979]).

Given a 3SAT instance  $\phi = C_1 \wedge \dots \wedge C_r$ , we define relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , fixed master data  $D_m$ , a set  $V$  of fixed INDs and a CQ query  $Q$ . We show that  $\phi$  is satisfiable iff  $\text{RCQ}(Q, D_m, V)$  is empty. This suffices to show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\text{coNP}$ -hard.

Let  $X = \{x_1, \dots, x_n\}$  be the set of variables in  $\phi$ .

(a) The database schema  $\mathcal{R}$  consists of three relation schemas: two fixed schemas  $R_t$  and  $R_v$ , and a schema  $R$ . More specifically,

- $R_t = (x, \bar{x})$ , which is used to encode boolean values;
- $R_v = (l_1, l_2, l_3)$ , to encode disjunction; and
- $R = (A, x_1, \bar{x}_1, \dots, x_n, \bar{x}_n)$ , to encode truth assignments. Here  $\text{dom}(A)$  is infinite.

(b) We define a fixed master data schema  $\mathcal{R}_m = (R_t^m, R_v^m)$ . We use fixed master data instance  $D_m = (I_t^m, I_v^m)$  of  $\mathcal{R}_m$  given as follows:

- the instance  $I_t^m$  of schema  $R_t^m = (z, \bar{z})$  consists of  $(0, 1)$  and  $(1, 0)$ ; intuitively,  $I_t^m$  is to enforce valid truth assignments for propositional variables;
- the instance  $I_v^m$  of schema  $R_v^m(l_1, l_2, l_3)$  consists of the seven truth assignments that satisfy  $l_1 \vee l_2 \vee l_3$ .

(c) We define a set  $V$  of CCs consisting of two fixed INDs, given as follows:

- $R_t(x, \bar{x}) \subseteq R_t^m(z, \bar{z})$ , to ensure that only valid truth assignments are considered for variables;
- $R_{\vee}(l_1, l_2, l_3) \subseteq R_{\vee}^m(l_1, l_2, l_3)$ , to encode disjunction.

(d) Based on the 3SAT instance  $\phi$ , we define the CQ query  $Q$  as

$$Q(z) = \exists x_1 \dots x_n \bar{x}_1 \dots \bar{x}_n (R(z, x_1, \bar{x}_1, \dots, x_n, \bar{x}_n) \wedge R_t(x_1, \bar{x}_1) \wedge \dots \wedge R_t(x_n, \bar{x}_n) \wedge q_1 \wedge \dots \wedge q_r),$$

where for each clause  $C_i = l_1 \vee l_2 \vee l_3$ , for  $i \in [1, r]$ , the expression  $q_i$  is  $R_{\vee}(l_1, l_2, l_3)$ , which encodes  $C_i$ .

We now verify that  $\phi$  is satisfiable iff  $\text{RCQ}(Q, D_m, V)$  is empty. Suppose that  $\phi$  is satisfiable. Then there exists a truth assignment  $\mu_0$  of  $X$  such that  $\phi$  is true. Assume by contradiction that there exists  $D$  in  $\text{RCQ}(Q, D_m, V)$ . Then we construct a tuple  $t$  such that  $t[X]$  is  $\mu_0(X)$  and  $t[A]$  is a distinct value not in  $D$ . This is possible since  $\text{dom}(A)$  is infinite. Let  $D'$  be the database obtained by adding  $t$  to the instance of  $R$  in  $D$ . Then obviously  $Q(D) \neq Q(D')$  while  $(D', D_m) \models V$ . This contradicts the assumption that  $D$  is complete. Conversely, suppose that  $\phi$  is not satisfiable. Then for each truth assignment of  $X$ ,  $\phi$  is false. In other words, for each database instance  $D$  of  $\mathcal{R}$ ,  $Q(D)$  is always empty. Let  $D_0$  consist of empty instances of  $R, R_t$  and  $R_{\vee}$ . Then  $D_0$  is complete for  $Q$  relative to  $D_m$  and  $V$ . That is,  $\text{RCQ}(Q, D_m, V)$  is nonempty.

As a result,  $\phi$  is not satisfiable iff  $\text{RCQ}(Q, D_m, V)$  is nonempty. Therefore,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\text{conp}$ -hard when  $\mathcal{L}_Q$  is CQ and  $\mathcal{L}_C$  is the class of INDs.

*Upper bound.* Based on Proposition 4.3, we first develop an NP algorithm that, given a CQ query  $Q$ , master data  $D_m$ , and a set  $V$  of INDs, determines whether  $\text{RCQ}(Q, D_m, V)$  is empty. This suffices to show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is in  $\text{conp}$ . We then show that the algorithm can be readily extended to UCQ and  $\exists\text{FO}^+$  queries.

Given  $Q, D_m$  and  $V$ , the algorithm first constructs the tableau representation  $(T_Q, u_Q)$  of  $Q$ . Then it takes two steps.

- (1) First, it tests whether there exists a valid valuation  $\mu$  of  $T_Q$ . This step can be done by in NP as follows:
  - (a) Guess a valuation  $\mu$  of the variables in  $T_Q$  using values in  $\text{Adom}$ .
  - (b) Test whether  $\mu$  is valid, *i.e.*,  $Q(\mu(T_Q)) \neq \emptyset$  and  $(\mu(T_Q), D_m) \models V$ . All these steps can be done in PTIME since the CCs are simple INDs.
 If it is not the case then the algorithm returns “no”.
- (2) Otherwise, it tests whether  $Q$  is not bounded. For all variables  $y$  in  $u_Q$ , if  $\text{dom}(y)$  is infinite, then it checks whether there exists an IND  $\pi_{(A, \dots)} \subseteq p$  in  $V$  such that  $y$  appears in column  $A$  in  $T_Q$ . This can obviously be done in PTIME in the size of  $Q$  and  $V$ . If such a variable exists, it returns “yes”; otherwise it returns “no”.

It is clear that the algorithm is in NP. As a result, one can decide whether  $\text{RCQ}(Q, D_m, V)$  is nonempty in  $\text{conp}$ .

We next extend the algorithm to deal with UCQ and  $\exists\text{FO}^+$  queries.

UCQ. Consider a UCQ query  $Q = Q_1 \cup \dots \cup Q_k$ , where  $Q_i$  is in CQ for each  $i \in [1, k]$ .

The proof of Proposition 4.3 can be readily extended to verify that  $\text{RCQ}(Q, D_m, V)$  is nonempty iff either each  $Q_i$  is bounded, or there exists no valid valuation of  $Q$ .

From this a  $\text{coNP}$  algorithm follows immediately: for each  $i \in [1, k]$ , it checks whether there exists a valid valuation for variables in  $Q_i$ . If not, it returns “no”. Otherwise for each  $i \in [1, k]$ , it checks whether every  $Q_i$  is bounded. It returns “yes” if there exists a  $Q_i$  that is not bounded, and “no” otherwise. The first step is in  $\text{NP}$ , and the second step can be done in  $\text{PTIME}$ . Thus we can conclude that it is in  $\text{coNP}$  to decide whether  $\text{RCQ}(Q, D_m, V)$  is nonempty when  $Q$  is in  $\text{UCQ}$ .

$\exists\text{FO}^+$ . We next consider the case when  $Q$  is a query in  $\exists\text{FO}^+$ . Observe that  $Q$  is equivalent to a (possibly exponentially large)  $\text{UCQ}$  query  $Q'$ . Based on this and the discussion above for  $\text{UCQ}$  queries, we know that  $\text{RCQ}(Q, D_m, V)$  is nonempty iff either the  $\text{UCQ}$  query  $Q'$  is bounded, or there exists no valid valuation for  $Q'$ .

We show that it is in  $\text{coNP}$  to decide whether  $\text{RCQ}(Q, D_m, V)$  is nonempty when  $Q$  is in  $\exists\text{FO}^+$ . We develop an  $\text{NP}$  algorithm for its complement, for deciding whether  $\text{RCQ}(Q, D_m, V)$  is empty, by extending the  $\text{CQ}$  counterpart given above. We augment both steps (1) and (2) with an additional guess of disjuncts (*i.e.*, which branch to take at each union) in  $Q$ . This yields a  $\text{CQ}$  query  $Q_c$ , which is obtained by unfolding  $Q$  only for the guessed branches, and is of polynomial size. More specifically, in each of steps (1) and (2), we first guess a  $Q_c$ , and then check whether there exists a valid valuation of  $Q_c$  and whether  $Q_c$  is bounded, respectively. Both steps can be done in  $\text{NP}$ . Thus the algorithm and the problem are in  $\text{coNP}$ .

(2) When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\text{CQ}$ . We now show that it is  $\text{NEXPTIME}$ -complete to determine, given a  $\text{CQ}$  query  $Q$ , master data  $D_m$ , and a set  $V$  of  $\text{CCs}$  in  $\text{CQ}$ , whether  $\text{RCQ}(Q, D_m, V)$  is nonempty.

*Lower bound.* We verify the  $\text{NEXPTIME}$  lower bound by reduction from the  $2^n \times 2^n$ - $\text{TILING}$  problem, which is  $\text{NEXPTIME}$ -complete (see, *e.g.*, [Dantsin and Voronkov 1997]). An instance of  $2^n \times 2^n$ - $\text{TILING}$  consists of a set of tiles  $T = \{t_0, t_1, \dots, t_k\}$ , and vertical and horizontal compatibility conditions between pairs of tiles, given by binary relations  $V \subseteq T^2$  and  $H \subseteq T^2$ , respectively. A *tiling* is a function  $f: \{1, \dots, 2^n\} \times \{1, \dots, 2^n\}$  into  $T$  such that

- $V(f(i, j), f(i + 1, j))$  holds for all  $1 \leq i < 2^n, 1 \leq j \leq 2^n$ ;
- $H(f(i, j), f(i, j + 1))$  holds for all  $1 \leq i \leq 2^n, 1 \leq j < 2^n$ ; and
- $f(1, 1) = t_0$ .

Here  $n$  is given in unary. The  $2^n \times 2^n$ - $\text{TILING}$  problem is to decide, given  $T, V$  and  $H$ , whether there exists a tiling  $f$ . It is known that this problem is  $\text{NEXPTIME}$ -complete (cf. [Papadimitriou 1994]).

Given an instance  $T, V$  and  $H$  of the tiling problem, we define relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , master data  $D_m$ , a set  $S$  of  $\text{CCs}$  in  $\text{CQ}$ , and a  $\text{CQ}$  query  $Q$  such that there exists a solution to the tiling problem for  $T, V$  and  $H$  iff  $\text{RCQ}(Q, D_m, V)$  is nonempty. The reduction closely follows the one presented in [Dantsin and Voronkov 1997].

(a) The master data schema  $\mathcal{R}_m$  consists of five relations:  $R_e^m, R_T^m, R_V^m, R_H^m$  and  $R_b^m$ , of arities 0, 1, 2, 2 and 1, respectively. We define the master data instance  $D_m$  of  $\mathcal{R}_m$  such that  $I_e^m = \emptyset, I_T^m = T, I_V^m = V, I_H^m = H$  and  $I_b^m = \{(0)\}$ .

(b) We define the database schema  $\mathcal{R}$  and the set  $S$  of CCs in steps. In a nutshell,  $\mathcal{R}$  consists of relation schemas  $R_1, \dots, R_n$  such that  $R_i$  encodes a  $2^i \times 2^i$  square of tiles for  $i \in [1, n]$ . We use master data  $D_m$  and the set  $S$  of CCs to assure the vertical and horizontal compatibility of the tiling.

Following the exposition given in [Dantsin and Voronkov 1997], we first generalize tiles to hypertiles. A *hypertile of rank  $i$*  is defined by induction as a  $2^i \times 2^i$  square of tiles. More specifically, a hypertile of rank 0 is any tile of the set  $T$ . Let  $T_1, T_2, T_3$  and  $T_4$  be hypertiles of rank  $i$ . Then the quadruple  $(H_1, H_2, H_3, H_4)$  is a hypertile of rank  $i + 1$ . That is,  $(H_1, H_2, H_3, H_4)$  corresponds to a set of tiles covering a square of size  $2^{(i+1)} \times 2^{(i+1)}$ , as follows:

$T_1$	$T_2$
$T_3$	$T_4$

Clearly, hypertiles of rank  $i$  can be identified with functions from  $\{1, \dots, 2^i\} \times \{1, \dots, 2^i\}$  into  $T$ . We call a hypertile a *tiling* if the corresponding function is a tiling (over the appropriate square size).

The first relation of  $\mathcal{R}$  is defined to be  $R_1(\text{id}, X_1, X_2, X_3, X_4, Z)$ . In an instance  $I_1$  of  $R_1$ , each tuple  $t$  is to encode the following: (i) a hypertile  $(t[X_1], t[X_2], t[X_3], t[X_4])$  of rank 1 that is a tiling; (ii) a unique identifier  $t[\text{id}]$  for the hypertile; and (iii) the top-left corner tile  $t[Z]$  of the hypertile. We will see why we need identifiers when we define relations for storing hypertiles of higher rank. An instance  $I_1$  of  $R_1$  is said to be well-formed if each of its tuples encodes the information as desired.

More precisely, we say that  $I_1$  is well-formed if it satisfies the following CCs in  $S$ , all definable in CQ:

- $V_1^{\text{key}}$ :  $\text{id} \rightarrow X_1, X_2, X_3, X_4$ , which is an FD assuring that  $\text{id}$  is a key;
- $V_1^A$ :  $\pi_A(R_1) \subseteq R_T^m$ , for  $A \in \{X_1, X_2, X_3, X_4, Z\}$ ; these assure that all attributes except  $\text{id}$  take values in  $T$ ;
- $V_1^{\text{vert}_1}$ :  $\pi_{X_1 X_3}(R_1) \subseteq R_V^m$  and  $V_1^{\text{vert}_2}$ :  $\pi_{X_2 X_4}(R_1) \subseteq R_V^m$ ; these ensure that the vertical compatibility conditions are satisfied by each tile encoded in a tuple;
- $V_1^{\text{hor}_1}$  and  $V_1^{\text{hor}_2}$  defined similarly, to ensure the horizontal compatibility conditions; and finally,
- $V_1^{\text{topl}}$ :  $\sigma_{X_1 \neq Z}(R_1) \subseteq \emptyset$  (recall that  $R_e^m = \emptyset$ ); these ensure that the tiles  $X_1$  and  $Z$  are the same.

We next define relations  $R_i$  in  $\mathcal{R}$ , for  $i > 2$ , to encode hypertiles of rank  $i$ . More specifically,  $R_i$  is 11-ary relation  $R_i(\text{id}, \text{id}_1, \text{id}_2, \text{id}_3, \text{id}_4, \text{id}_{12}, \text{id}_{13}, \text{id}_{24}, \text{id}_{34}, \text{id}_{1234}, Z)$ , where the attributes  $\text{id}_j$  refer to the identifiers of hypertiles of rank  $i - 1$ , *i.e.*, elements in  $R_{i-1}$ . As before, the  $\text{id}$ -attribute is to serve as identifier for a hypertile of rank  $i$  stored in an instance of  $R_i$ . A well-formed instance  $I_i$  of  $R_i$  consists of tuples  $t$  such that (i)  $t[\text{id}]$  is a unique identifier for the hypertile of rank  $i$  represented by  $(t[\text{id}_1], t[\text{id}_2], t[\text{id}_3], t[\text{id}_4])$ ; (ii)  $t[Z]$  is to denote the tile at the top-left corner; and (iii) the hypertile  $(t[\text{id}_1], t[\text{id}_2], t[\text{id}_3], t[\text{id}_4])$  satisfies the appropriate vertical and horizontal compatibility conditions. These conditions specify constraints on the hypertiles of rank  $(i - 1)$ :  $t[\text{id}_{12}]$ ,  $t[\text{id}_{13}]$ ,  $t[\text{id}_{24}]$ ,  $t[\text{id}_{34}]$ , and  $t[\text{id}_{1234}]$ . More precisely, assume that  $(t[\text{id}_1], t[\text{id}_2], t[\text{id}_3], t[\text{id}_4])$  corresponds to

$$\begin{aligned}
R_{(i-1)}(\text{id}_1, a_1, a_2, a_3, a_4, -, -, -, -, z) & R_{(i-1)}(\text{id}_2, b_1, b_2, b_3, b_4, -, -, -, -, -) \\
R_{(i-1)}(\text{id}_3, c_1, c_2, c_3, c_4, -, -, -, -, -) & R_{(i-1)}(\text{id}_4, d_1, d_2, d_3, d_4, -, -, -, -, -),
\end{aligned}$$

where ‘-’ denotes an arbitrary value. Then the hypertiles identified by the remaining identifiers in  $t$  must have the form

$$\begin{aligned}
R_{(i-1)}(\text{id}_{12}, a_2, b_1, a_4, b_3, -, -, -, -, -) & R_{(i-1)}(\text{id}_{13}, a_3, a_4, c_1, c_2, -, -, -, -, -) \\
R_{(i-1)}(\text{id}_{24}, b_3, b_4, d_1, d_2, -, -, -, -, -) & R_{(i-1)}(\text{id}_{34}, c_2, d_1, c_4, d_3, -, -, -, -, -) \\
R_{(i-1)}(\text{id}_{1234}, a_4, b_3, c_3, d_1, -, -, -, -, -) &
\end{aligned}$$

That is, such tuples  $t$  in  $R_i$  encode tilings of a square of size  $2^i \times 2^i$ , provided that the identifiers in  $t$  appear in  $R_{(i-1)}$ .

To assure that only well-formed instances of  $R_i$  are considered, we include certain CCs in  $S$ , assuring that no invalid tuples are in instances of  $R_i$ . These can be easily expressed as CCs of the form  $q \subseteq \emptyset$ , where  $q$  is a CQ query. An example CC is:

$$\exists t s_1 s_2 (R_i(t) \wedge R_{(i-1)}(t[\text{id}_1], s_1) \wedge R_{(i-1)}(t[\text{id}_{12}], s_2) \wedge s_1[R_{(i-1)}.\text{id}_2] \neq s_2[R_{(i-1)}.\text{id}_1]) \subseteq \emptyset.$$

This asserts that if hypertile  $t[\text{id}_1] = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$ , then  $t[\text{id}_{12}]$  is of the form  $\begin{bmatrix} X_2 & - \\ - & - \end{bmatrix}$ . All such conditions can be expressed as CCs in CQ in a similar way. Finally, we also include an FD in  $S$  that ensures that the id-attribute is a key of  $R_i$ .

Inductively we define  $n$  relation  $R_1, \dots, R_n$  in  $\mathcal{R}$ . Let  $t$  be a tuple in a well-formed instance of  $R_n$  with  $t[Z] = t_0$ . Then this tuple intends to encode a solution to the  $2^n \times 2^n$ -TILING problem, provided that all the identifiers in  $t$  can be traced all the way back to tiles in  $R_1$ . The last CC to be included in  $S$  checks the existence of such a tuple. Furthermore, if such a tuple (hence a tiling) exists it bounds instances of  $R_b$  by  $R_b^m$ . Otherwise it leaves instances of  $R_b$  unbounded.

More specifically, we define the CC using a sequence of queries: for each  $1 < i < n$ ,  
—  $Q_i^p = \pi_{\text{id}}(Q_i^s)$ , and  
—  $Q_{(i+1)}^s$  selecting all tuples from  $R_{(i+1)}$  that only have identifiers in  $Q_i^p$ .

For example,  $Q_1^p = \pi_{\text{id}}(R_1)$ , and  $Q_2^s$  is to select all tuples from  $R_2$  that only have identifiers in  $Q_1^p$ . Observe that the result of  $Q_n^s$  consists of tuples that are hypertiles of rank  $n$  with the following properties: (i) they can be traced all the way back to tiles in  $R_1$ , and (ii) they satisfy the compatibility conditions.

Leveraging these queries, we then define the CC  $\phi: q(w) \subseteq R_b^m$ , where

$$q(w) = \exists t (Q_n^s(t) \wedge t[Z] = t_0) \wedge R_b(w).$$

That is, if a tiling of  $2^n \times 2^n$  with top-left tile  $t_0$  exists, then any instance of  $I_b$  of  $R_b$  is either empty or  $\{(0)\}$ .

(c) Finally, we define the query  $Q$  such that it simply returns  $R_b$ .

This completes the construction of the coding of the tiling instance.

We now show that the  $2^n \times 2^n$ -TILING problem for  $T$ ,  $V$  and  $H$  has a solution iff  $\text{RCQ}(Q, D_m, V)$  is nonempty. Suppose that there exists a tiling function  $f$  from  $\{1, \dots, 2^n\} \times \{1, \dots, 2^n\}$  to  $T$ . Then let  $D = (I_1, \dots, I_n, I_b)$  be the instance such that  $I_i$  stores all hypertiles of rank  $i$  given by  $f$ , and  $I_b = \{(1)\}$ . Clearly we have

that  $(D, D_m) \models V$ . Furthermore, by the CC  $\phi$  in  $S$ , it is impossible to add tuples to  $I_b$ . Hence for each addition of tuples to any of the  $I_j$ 's, either the resulting  $D'$  is not a partially closed extension of  $D$ , or  $Q(D') = \{(1)\} = Q(D)$ . Hence  $D$  is indeed in  $\text{RCQ}(Q, D_m, V)$ . That is,  $\text{RCQ}(Q, D_m, V)$  is nonempty.

Conversely, suppose that there is no tiling for  $T, V$  and  $H$ . Assume by contradiction that there exists a  $D \in \text{RCQ}(Q, D_m, V)$ . Then clearly,  $I_b$  must be bounded since  $D$  is complete. Hence the condition  $\exists t(Q_n^s(t) \wedge t[Z] = t_0)$  in  $\phi$  is satisfied and moreover, the conditions specified by  $Q_i^p = \pi_{\text{id}}(Q_i^s)$  and  $Q_{(i+1)}^s$  hold for  $(I_1, \dots, I_n)$ . Since  $(D, D_m) \models V$ , it is easily verified that a tiling for  $T, V$  and  $H$  can be induced from the tuples in  $D$ , by tracing the identifiers in  $I_n, I_{n-1}, \dots$ , until  $I_1$  is reached. This contradicts the assumption that there is no tiling for  $T, V$  and  $H$ .

*Upper bound.* From Proposition 4.2 an NEXPTIME algorithm can be readily be developed such that, given a CQ query  $Q$ , master data  $D_m$  and a set  $V$  of CCs in CQ, it determines whether  $\text{RCQ}(Q, D_m, V)$  is nonempty. Indeed, as remarked earlier, the proof of Proposition 4.2 establishes a small model property:  $\text{RCQ}(Q, D_m, V)$  is nonempty iff there exists a relatively complete database  $D$  such that the size of  $D$  is bounded by an exponential in the sizes of  $Q, V$  and  $D_m$ . The small model property can be readily verified by leveraging the monotonicity of CQ queries.

Capitalizing on Proposition 4.2, the algorithm first constructs the tableau representation  $(T_Q, u_Q)$  of  $Q$ . It then takes the following steps.

- (1) Check whether all variables in  $u_Q$  have a finite domain. This takes PTIME. If so, return “yes”.
- (2) Guess a set of  $\mathcal{V}$  valuations of  $V$  and consider  $D_{\mathcal{V}} = \bigcup_{\nu \in \mathcal{V}} D_{\nu}$ . Check whether  $(D_{\mathcal{V}}, D_m) \models V$ . If not, reject the guess and repeat the step. Otherwise proceed to the next step. These can be done in NEXPTIME since there are only exponentially many valuations of  $V$  taken values from  $\text{Adom}$ .
- (3) For all valuations  $\mu$  of  $T_Q$  that take values from  $\text{Adom}$ , check whether the following three conditions hold:
  - (a)  $(\mu(T_Q), D_m) \models V$  and  $Q(\mu(T_Q)) \neq \emptyset$ . If so, *i.e.*, when  $\mu$  is a valid valuation of variables in  $T_Q$ , then proceed to the next step. This can be done in NP.
  - (b)  $(D_{\mathcal{V}} \cup \mu(T_Q), D_m) \models V$ . This can be done by using a conNP oracle: for each CC  $q_i \subseteq p_i$  in  $V$ , where  $q_i$  is represented as a tableau query  $(T_i, u_i)$ , guess a valuation  $\nu$  of  $T_i$  using values in  $D_{\mathcal{V}} \cup \mu(T_Q)$ , and check whether  $(\nu(T_i), D_m) \not\models \phi$ . The condition on  $\mu$  is then satisfied if the conNP process returns “no”.
  - (c) Every variable  $y$  in  $u_Q$  with an infinite domain is bounded by  $\mathcal{V}$  *w.r.t.*  $\mu$ . This again can be checked in NP.
- (4) If all the conditions are satisfied, return “yes”; otherwise reject the current guess of  $\mathcal{V}$  and repeat steps (2)–(3).

Note that all the processes in step (3) can clearly be done in EXPTIME. Putting these together, the algorithm is in NEXPTIME.

(3) *When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ.* The problem is obviously NEXPTIME-hard as shown by the lower bound proof for (2) given above.

We show that the problem is in NEXPTIME. Consider a query  $Q = Q_1 \cup \dots \cup Q_k$  in UCQ, where  $Q_i$  is represented as a tableau query  $(T_i, u_i)$ . Let  $D_m$  be master data, and  $V$  be a set of CCs in UCQ. We modify the NEXPTIME algorithm given in (2) as follows. In step (1), we check whether all the variables in  $u_i$  have a finite domain, for all  $i \in [1, k]$ . This can still be done in PTIME. Step (2) remains unchanged. Indeed, as remarked earlier,  $V$  can be treated *w.l.o.g.* as a set of CCs in CQ. In step (3), we inspect all valuations  $\mu = (\mu_1, \dots, \mu_k)$  of  $Q$ , and check whether  $\mu$  is valid, whether  $(D_V \cup \bigcup_{i \in [1, k]} \mu_i(T_i), D_m) \models V$  and whether for all  $i \in [1, k]$  and for all variable  $y$  with an infinite domain in  $u_i$ ,  $y$  is bounded by  $V$  *w.r.t.*  $\mu$ , in steps (3.a)–(3.c), respectively. Clearly these changes do not increase the EXPTIME complexity of step (3). Thus the algorithm is still in NEXPTIME. By Corollary 4.4, the algorithm suffices to determine whether  $\text{RCQ}(Q, D_m, V)$  is nonempty.

(4) When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ . The problem is NEXPTIME-hard in this setting, since  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is already NEXPTIME-hard when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ.

For the upper bound, observe that this case can be reduced to the case when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ. Indeed, given an  $\exists\text{FO}^+$  query  $Q$  and a set  $V$  of CCs in  $\exists\text{FO}^+$ , we can rewrite  $Q$  and the CCs in  $V$  into an equivalent UCQ query and an equivalent set of CCs in UCQ. Although the rewriting incurs an exponential blowup in the size of the queries, the overall complexity of the algorithm given for UCQ above remains in NEXPTIME. Therefore,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  remains in NEXPTIME-hard when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ .  $\square$

As remarked earlier, master data  $D_m$  and containment constraints  $V$  are often predefined and fixed in practice. Recall from Corollary 3.7 that fixed  $D_m$  and  $V$  have no impact on the complexity of  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$ . In contrast, we show below that fixed  $D_m$  and  $V$  do make our lives easier, to some extent. (a) When  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ, UCQ or  $\exists\text{FO}^+$ ,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  becomes  $\Sigma_3^p$ -complete, down from NEXPTIME-complete. (b) On the other hand, when  $\mathcal{L}_Q$  is CQ and  $\mathcal{L}_C$  is the class of INDs, the problem remains conP-complete. (c) Fixed  $D_m$  and  $V$  do not help when either  $\mathcal{L}_Q$  or  $\mathcal{L}_C$  is FO or FP, as we have seen in Theorem 4.1.

**Corollary 4.6:** When master data and CCs are fixed,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is

- (1) conP-complete if  $\mathcal{L}_C$  is the class of INDs and  $\mathcal{L}_Q$  is CQ, UCQ or  $\exists\text{FO}^+$ , and
- (2)  $\Sigma_3^p$ -complete if  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ, UCQ or  $\exists\text{FO}^+$ .  $\square$

PROOF. When master data and CCs are fixed,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  remains conP-complete when  $\mathcal{L}_C$  consists of INDs and  $\mathcal{L}_Q$  ranges over CQ, UCQ and  $\exists\text{FO}^+$ . Indeed, the conP-hardness follows from the proof for the lower bound of Theorem 4.5 (1), in which only fixed master data  $D_m$  and a set  $V$  of fixed INDs are used. In addition, its conP upper bound carries over when  $D_m$  and  $V$  are fixed.

To verify the  $\Sigma_3^p$ -completeness when master data and CCs are fixed, it suffices to show the following: (1) lower bound:  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\Sigma_3^p$ -hard when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ; and (2) upper bound:  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is in  $\Sigma_3^p$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$ . For if these hold, then the lower bound remains intact when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are UCQ or  $\exists\text{FO}^+$ , and  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  remains in  $\Sigma_3^p$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ or UCQ.

Lower bound. We show that  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is  $\Sigma_3^p$ -hard when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ,

by reduction from  $\exists^*\forall^*\exists^*$  3SAT-problem, which is  $\Sigma_3^p$ -complete (cf. [Papadimitriou 1994]). Given an instance  $\varphi = \exists X\forall Y\exists Z\psi(X, Y, Z)$  of the latter problem, we define relational schemas  $\mathcal{R}$  and  $\mathcal{R}_m$ , a fixed set  $V$  of CCs in CQ, fixed master data  $D_m$  and a CQ query  $Q$ . We show that  $\varphi$  is satisfiable iff  $\text{RCQ}(Q, D_m, V)$  is nonempty.

Assume that  $X = \{x_1, \dots, x_n\}$ ,  $Y = \{y_1, \dots, y_m\}$  and  $Z = \{z_1, \dots, z_p\}$ . Assume that  $\psi = C_1 \wedge \dots \wedge C_r$ , which is an instance of 3SAT over  $X \cup Y \cup Z$ . We define  $\mathcal{R}$ ,  $\mathcal{R}_m$ ,  $V$ ,  $D_m$  and  $Q$  as follows.

(a) The relational schema  $\mathcal{R}$  consists six relation schemas, including  $R_1, R_2, R_3$  and  $R_4$  given in the proof for the  $\Pi_2^p$  lower bound of Theorem 3.6. In addition,  $\mathcal{R}$  contains the following:

- $R_X(A_1, \dots, A_n, \text{id})$ , which is to encode a unique truth assignment for variables  $X$ , where  $\text{id}$  is the key of  $R_X$ ;
- $R_b(q, A)$ , where  $A$  has infinite domain  $\mathbf{d}$ ; intuitively, we shall use  $R_b(A)$  to inspect whether variables in the query  $Q$  are bounded or not.

The schema of master data  $\mathcal{R}_m$  is defined in terms of  $R_1, R_2, R_3, R_4$  and  $R_b$  of  $\mathcal{R}$ , and an additional nullary relation  $R_e^m$ , *i.e.*,  $\mathcal{R}_m = (R_1^m = R_1, R_2^m = R_2, R_3^m = R_3, R_4^m = R_4, R_b^m = R_b, R_e^m)$ .

(b) The master data instance  $D_m$  of  $\mathcal{R}_m$  contains the fixed relations  $I_1^m = I_{01}, I_2^m = I_\vee, I_3^m = I_\wedge$ , and  $I_4^m = I_\neg$ , which are defined in the proof for the lower bound of Theorem 3.6; in addition, it includes fixed relations  $I_b^m = \{(0)\}$  and  $I_e^m = \emptyset$ .

(c) The set  $V$  of CCs includes  $R_i \subseteq R_i^m$ , for  $i \in [1, 4]$ , and in addition, the following:

- $V^{\text{key}}$ , expressing that  $\text{id}$  is a key of  $R_X$ ; this CC can be expressed by means of a CQ query and  $R_e^m$  (*i.e.*,  $\emptyset$ );
- $\pi_{A_j}(R_X) \subseteq R_{(0,1)}$ , for  $j \in [1, n]$ ; these CCs ensure that  $A_i \in \{0, 1\}$  for  $j \in [1, n]$ ; and finally
- $q_b(A) \subseteq R_b^m$ , where  $q_b(A) = R_b(1, A)$ ; this CC asserts that the  $A$  attributes in an instance of  $R_b$  are bounded only if  $q = 1$ . We will see shortly how  $q$  is related to the satisfiability of  $\varphi$ .

(d) The query  $Q$  is defined as follows:

$$Q(Y, A) = Q_x(X) \wedge Q_1(x_1, \dots, x_n, Y, q) \wedge R_b(q, A),$$

where  $Q_x(X) = \bigwedge_{i \in [1, n]} R_X(x_i, i)$ , that is, it selects from  $R_X$  the truth assignments for  $x_1, \dots, x_n$ . We use  $\mu_x$  to denote the truth assignment of  $X$  selected by  $Q_x$ . Here the query  $Q_1$  is a variation of the CQ query given in the proof for the lower bound of Theorem 3.6 (referred to as  $Q$  there). More specifically, for a given truth assignment  $\mu_x$  of  $X$ ,  $Q_1$  returns  $(\mu_y, q)$  for all  $\mu_y$  that is a truth assignment for  $Y$ ; in addition, it returns  $q = 1$  when  $\exists Z C_1 \wedge \dots \wedge C_r$  holds for the given  $\mu_x$  and  $\mu_y$ , and  $q = 0$  otherwise.

Putting these together, given any instance  $D$  of  $\mathcal{R}$ ,  $Q(D)$  returns  $(\mu_Y, A)$  for all truth assignment  $\mu_Y$  to  $Y$ , whenever a single truth assignment  $\mu_X$  for  $X$  is selected by  $Q_X$ , no matter whether  $(\mu_X, \mu_Y)$  satisfies  $\forall Z\psi$  or not.

Observe that  $D_m$  and  $V$  are fixed, as desired.

We next show the correctness of the reduction. First, assume that  $\varphi$  is satisfiable. We show that  $\text{RCQ}(Q, D_m, V)$  is nonempty. Since  $\varphi$  is satisfiable, there exists  $\mu_x$

such that for all  $\mu_y$ ,  $\forall Z\psi$  is true. Define a database  $D$  of  $\mathcal{R}$  such that the instance  $I_X$  of  $R_X$  encodes  $\mu_x$ , *i.e.*,  $I_X = \{(\mu_X(x_1), 1), \dots, \mu_X(x_n), n)\}$ , the instance  $I_b$  of  $R_b$  is a singleton  $\{(1, 0)\}$ , and moreover, the instances of  $R_1$ – $R_4$  are fixed as given in the proof of Theorem 3.6. Then  $D$  is complete. Indeed,  $Q(D)$  consists of all possible truth assignments of  $Y$  while the  $A$  attribute is fixed to be 0. Furthermore, adding any tuples to  $D$  either violates  $V$  or does not change the answer to  $Q$ . Note that here it is essential to guarantee that a single assignment  $\mu_X$  to  $X$  is used.

Conversely, assume that there exists a database  $D$  of  $\mathcal{R}$  that is complete relative to  $(D_m, V)$ . Assume by contradiction that  $\varphi$  is not satisfiable. Then for each  $\mu_x$  of  $X$ , there exists  $\mu_y$  such that  $\forall Z\psi$  is false. Add  $(0, a)$  to the instance  $I_b$  of  $R_b$  in  $D$ , where  $a$  is a constant not in  $D$  (this is possible since  $A$  has an infinite domain). Refer to this extension as  $D'$ , and the truth assignment specified by the instance of  $R_X$  in  $D$  as  $X_0$ . Then obviously  $(D', D_m) \models V$  but  $Q(D') \neq Q(D)$ . Indeed, there exists  $(Y_0, a)$  in the answer to  $Q$  in  $D'$  that is not in  $Q(D)$ , where  $(X_0, Y_0)$  does not satisfy  $\forall Z\psi$ , *i.e.*,  $(X_0, Y_0, 0)$  is in the answer to  $Q_1$  embedded in  $Q$ .

Upper bound. To illustrate the main idea, we first consider the case when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are CQ. Recall the NEXPTIME algorithm for determining whether  $\text{RCQ}(Q, D_m, V)$  is nonempty given in the proof of Theorem 4.5. We show that when  $D_m$  and  $V$  are fixed, we can modify the algorithm to be in  $\Sigma_3^P$ . Indeed, when  $V$  and  $D_m$  are fixed, it suffices to inspect “small models”  $D$  of a polynomial size.

More specifically, in step (2) of the algorithm, one can guess sets  $\mathcal{V}$  of valuations of  $V$  in NP since  $V$  is fixed. Step (3) of the algorithm can be done by using a  $\Pi_2^P$  oracle. Indeed, one can guess a valuation  $\mu$  of  $T_Q$  and then check whether any of the conditions specified in steps (3.a)–(3.c) is not satisfied, where  $Q$  is represented as a tableau query  $(T_Q, u_Q)$ . To see these, observe the following: (a) checking whether  $\mu$  is not valid is in coNP (step 3.a), (b) inspecting whether  $(D_{\mathcal{V}} \cup \mu(T_Q), D_m) \not\models V$  is in NP; and (c) checking whether there exists a variable with an infinite domain in  $u_Q$  that is not bounded is also in coNP. Putting these together, step (3) is in coNP<sup>NP</sup>, *i.e.*, in  $\Pi_2^P$ . Hence the algorithm is in NP <sup>$\Pi_2^P$</sup> . That is, it is in  $\Sigma_3^P$  to determine whether  $\text{RCQ}(Q, D_m, V)$  is nonempty.

We next show that it remains in  $\Sigma_3^P$  to determine whether  $\text{RCQ}(Q, D_m, V)$  is nonempty when  $Q$  is an  $\exists\text{FO}^+$  query and  $V$  is a fixed set of CCs in  $\exists\text{FO}^+$ . As observed in the proof of Theorem 4.5, the case of  $\exists\text{FO}^+$  can be reduced to the case for UCQ. From this and Corollary 4.4, it follows that it suffices to check the conditions E5 and E6 given in Section 4.2 to determine the non-emptiness of  $\text{RCQ}(Q, D_m, V)$ .

We extend the algorithm for CQ such that E5 and E6 can be checked without rewriting  $Q$  into a UCQ query. Step (1) inspects whether all variables in  $Q$  have a finite domain, and can be done in PTIME in the size of  $Q$ . Step (2) remains unchanged, since the CCs in  $V$  can be rewritten as equivalent CCs in CQ without incurring any increase to the complexity bound when  $V$  is fixed. Step (3) guesses a CQ subquery of  $Q$  and a valuation  $\mu$ , and then checks whether any of the conditions specified in steps (3.a)–(3.c) is not satisfied. Along the same lines as the argument for the CQ case above, one can verify that step (3) can be done by a  $\Pi_2^P$  oracle. Thus the algorithm is in  $\Sigma_3^P$ . That is,  $\text{RCQP}(\mathcal{L}_Q, \mathcal{L}_C)$  is in  $\Sigma_3^P$  when  $\mathcal{L}_Q$  and  $\mathcal{L}_C$  are  $\exists\text{FO}^+$  and in addition, when master data and CCs are fixed.  $\square$

RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ )	Complexity
(FO, CQ) (Th. 3.1(1))	undecidable
(CQ, FO) (Th. 3.1(2))	undecidable
(FP, CQ) (Th. 3.1(3))	undecidable
(fixed(FP), FP) (Th. 3.1(4))	undecidable
(CQ, INDS), ( $\exists$ FO <sup>+</sup> , INDS) (Th. 3.6(1))	$\Pi_2^P$ -complete
(CQ, CQ) (Th. 3.6(2))	$\Pi_2^P$ -complete
(UCQ, UCQ) (Th. 3.6(3))	$\Pi_2^P$ -complete
( $\exists$ FO <sup>+</sup> , $\exists$ FO <sup>+</sup> ) (Th. 3.6(4))	$\Pi_2^P$ -complete

Table I. Complexity of RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ )

## 5. CONCLUSIONS

We have proposed the notion of the relative completeness of information to capture incomplete information in emerging applications such as Master Data Management. We have also formulated and studied two important decision problems associated with this notion, namely, RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ ) and RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ ). For a variety of query languages for expressing queries ( $\mathcal{L}_Q$ ) and containment constraints ( $\mathcal{L}_C$ ), we have provided a comprehensive picture of lower and upper bounds for these problems, all matching. We have also presented sufficient and necessary conditions for a database or a query to be relatively complete *w.r.t.* master data and containment constraints, for certain cases where RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ ) and RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ ) are decidable. We expect that these results will help users determine whether a database has complete information to answer a query, whether a query can find a complete answer at all, and what data should be collected by a database in order to yield a complete answer to a query.

We summarize the complexity bounds for RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ ) and RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ ) in Tables I and II, respectively, annotated with their corresponding theorems, where *fixed*( $\mathcal{L}$ ) indicates a set of fixed queries in  $\mathcal{L}$ . When master data and containment constraints are fixed, we only show complexity bounds that differ from their counterparts in the general settings.

The study of relatively complete information is still preliminary. One issue is about how to incorporate missing values, together with missing tuples, into the framework. To this end, preliminary results have been reported in [Fan and Geerts 2010], which uses presentation systems for possible worlds (conditional tables [Grahne 1991; Imieliński and Lipski 1984]) instead of traditional relations. Another open issue concerns *syntactic* characterizations for relatively complete databases or queries, in certain cases. A third interesting topic is to identify tractable (PTIME) special cases for RCDP( $\mathcal{L}_Q, \mathcal{L}_C$ ) and RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ ). Finally, although the containment constraints proposed in this work are fairly general, in certain applications one might want to formulate containment constraints not only from databases to master data, but also *from* the master data *to* the databases. We defer the treatment of this richer class of constraints to future work.

## ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library by visiting the following URL: <http://www.acm.org/pubs/citations/>

RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ )	Complexity
(FO, fixed(FO)) (Th. 4.1(1))	undecidable
(CQ, FO) (Th. 4.1(2))	undecidable
(FP, fixed(FP)) (Th. 4.1(3))	undecidable
(CQ, FP) (Th. 4.1(4))	undecidable
(CQ, INDS), ( $\exists$ FO <sup>+</sup> , INDS) (Th. 4.5(1))	coNP-complete
(CQ, CQ) (Th. 4.5(2.a))	NEXPTIME-complete
(UCQ, UCQ) (Th. 4.5(2.b))	NEXPTIME-complete
( $\exists$ FO <sup>+</sup> , $\exists$ FO <sup>+</sup> ) (Th. 4.5(2.c))	NEXPTIME-complete
When $D_m$ and $V$ are fixed	
(CQ, CQ) (Cor. 4.6(2))	$\Sigma_3^P$ -complete
(UCQ, UCQ) (Cor. 4.6(2))	$\Sigma_3^P$ -complete
( $\exists$ FO <sup>+</sup> , $\exists$ FO <sup>+</sup> ) (Cor. 4.6(2))	$\Sigma_3^P$ -complete

Table II. Complexity of RCQP( $\mathcal{L}_Q, \mathcal{L}_C$ )

[journals/tods/20YY-V-N/p1-url](#).

## REFERENCES

- ABITEBOUL, S. AND DUSCHKA, O. M. 1998. Complexity of answering queries using materialized views. In *PODS*. 254–263.
- ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.
- ARENAS, M., BERTOSSI, L. E., AND CHOMICKI, J. 1999. Consistent query answers in inconsistent databases. In *PODS*. 68–79.
- BATINI, C. AND SCANNAPIECO, M. 2006. *Data Quality: Concepts, Methodologies and Techniques*. Springer.
- BRAVO, L., FAN, W., AND MA, S. 2007. Extending dependencies with conditions. In *VLDB*. 243–254.
- CALI, A., LEMBO, D., AND ROSATI, R. 2003. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*. 260–271.
- CALVANESE, D., GIACOMO, G. D., LENZERINI, M., AND VARDI, M. Y. 2007. View-based query processing: On the relationship between rewriting, answering and losslessness. *TCS* 371, 3, 169–182.
- CHANDRA, A. K. AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *STOC*. 77–90.
- CHOMICKI, J. 2007. Consistent query answering: Five easy pieces. In *ICDT*. 1–17.
- DANTSIN, E. AND VORONKOV, A. 1997. Complexity of query answering in logic databases with complex values. In *LFCS*. 56–66.
- DEUTSCH, A., LUDÄSCHER, B., AND NASH, A. 2007. Rewriting queries using views with access patterns under integrity constraints. *TCS* 371, 3, 200–226.
- DREIBELBIS, A., HECHLER, E., MATHEWS, B., OBERHOFER, M., AND SAUTER, G. 2007. Master data management architecture patterns. IBM.
- ELKAN, C. 1990. Independence of logic database queries and updates. In *PODS*. 154–160.
- FAN, W. 2008. Dependencies revisited for improving data quality. In *PODS*. 159–170.
- FAN, W. AND GEERTS, F. 2009. Relative information completeness. In *PODS*. 97–106.
- FAN, W. AND GEERTS, F. 2010. Capturing missing tuples and missing values. In *PODS*.
- FAN, W., GEERTS, F., JIA, X., AND KEMENTSIETSIDIS, A. 2008. Conditional functional dependencies for capturing data inconsistencies. *TODS* 33, 2, 1–48.
- GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.
- GOTTLOB, G. AND ZICARI, R. 1988. Closed world databases opened through null values. In *VLDB*. 50–61.

- GRAHNE, G. 1991. *The Problem of Incomplete Information in Relational Databases*. Springer.
- HERZOG, T. N., SCHEUREN, F. J., AND WINKLER, W. E. 2007. *Data Quality and Record Linkage Techniques*. Springer.
- IMIELIŃSKI, T. AND LIPSKI, JR, W. 1984. Incomplete information in relational databases. *JACM* 31, 4, 761–791.
- LEVY, A. Y. 1996. Obtaining complete answers from incomplete databases. In *VLDB*. 402–412.
- LEVY, A. Y. AND SAGIV, Y. 1993. Queries independent of updates. In *VLDB*. 171–181.
- LI, C. 2003. Computing complete answers to queries in the presence of limited access patterns. *VLDB J.* 12, 3, 211–227.
- LOSHIN, D. 2008. *Master Data Management*. Morgan Kaufmann.
- MOTRO, A. 1989. Integrity = validity + completeness. *TODS* 14, 4, 480–502.
- PAPADIMITRIOU, C. H. 1994. *Computational Complexity*. Addison-Wesley.
- RADCLIFFE, J. AND WHITE, A. 2008. Key issues for master data management. Gartner.
- SEGOUFIN, L. AND VIANU, V. 2005. Views and queries: determinacy and rewriting. In *PODS*. 49–60.
- SPIELMANN, M. 2000. Abstract state machines: Verification problems and complexity. Ph.D. thesis, RWTH Aachen.
- VAN DER MEYDEN, R. 1998. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, J. Chomicki and G. Saake, Eds. Kluwer.
- VARDI, M. 1986. On the integrity of databases with incomplete information. In *PODS*. 252–266.

Received Month Year; revised Month Year; accepted Month Year