Information Systems 45 (2014) 18–34

Contents lists available at ScienceDirect

Information Systems

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

On the data complexity of relative information completeness

Yang Cao [a,b], Ting Deng [b,*], Wenfei Fan [a,b], Floris Geerts [c]

[a] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
[b] School of Computer Science and Engineering, Beihang University, No. 37 XueYuan Road, Haidian District, Beijing, China
[c] Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, B-2020 Antwerpen, Belgium

ARTICLE INFO

Article history:
Received 19 July 2013
Received in revised form
20 February 2014
Accepted 15 April 2014
Recommended by: D. Suciu
Available online 24 April 2014

Keywords:
Incomplete information
Relative completeness
Master data management
Partially closed databases
Data complexity

ABSTRACT

Databases in an enterprise are often *partially closed*: parts of their data must be contained in master data, which has complete information about the core business entities of the enterprise. With this comes the need for studying *relative information completeness*: a partially closed database is said to be *complete* for a query *relative to* master data if it has complete information to answer the query, *i.e.*, extending the database by adding more tuples either does not change its answer to the query or makes it no longer partially closed *w.r.t.* the master data. This paper investigates three problems associated with relative information completeness. Given a query Q and a partially closed database D *w.r.t.* master data D_m, (1) the *relative completeness* problem is to decide whether D is complete for Q relative to D_m; (2) the *minimal completeness* problem is to determine whether D is a minimal database that is complete for Q relative to D_m; and (3) the *bounded extension* problem is to decide whether it suffices to extend D by adding at most K tuples, such that the extension makes a partially closed database that is complete for Q relative to D_m. While the combined complexity bounds of the relative completeness problem and the minimal completeness problem are already known, neither their data complexity nor the bounded extension problem has been studied. We establish upper and lower bounds of these problems for data complexity, all matching, for Q expressed in a variety of query languages.

1. Introduction

When we query a database, we naturally expect the database to have complete information for answering our query. However, databases in the real world are often incomplete, from which tuples are missing. Indeed, it is estimated that "pieces of information perceived as being needed for clinical decisions were missing from 13.6% to 81% of the time" [27].

This gives rise to the following question: for a given query Q, can its complete answer be found from an incomplete database D? That is, the answer to Q in D remains unchanged no matter how D is extended by adding new tuples. In other words, although D is generally incomplete, it still possesses sufficient information to answer Q. The need for studying this problem is evident in practice: if D does not have complete information for answering Q, one can hardly expect that the answer to Q in D is complete or even correct.

The traditional Closed World Assumption (CWA) or the Open World Assumption (OWA) does not help us here. The CWA assumes that a database contains all the tuples representing real-world entities, *i.e.*, it assumes that no tuples are missing from a database. As remarked earlier, this rarely happens in practice. The OWA assumes that *tuples* representing

* Corresponding author.
E-mail addresses: caoyang@act.buaa.edu.cn (Y. Cao), dengting@act.buaa.edu.cn (T. Deng), wenfei@inf.ed.ac.uk (W. Fan), floris.geerts@ua.ac.be (F. Geerts).

http://dx.doi.org/10.1016/j.is.2014.04.001
0306-4379/

real-world entities may be *missing*, but we cannot do much about it (see [2,34] for surveys). Indeed, for few sensible queries Q and databases D, adding tuples to D does not change the answer to Q in D.

The good news is that real-life databases are *neither* entirely closed-world *nor* entirely open-world, in light of the increasing use of master data management (MDM [26]) systems provided by, *e.g.*, IBM, SAP, Microsoft and Oracle. 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 its core business entities. Master data consists of a closed-world database D_m about the enterprise in certain aspects, *e.g.*, employees and products. Other databases of the enterprise are *partially closed w.r.t.* D_m: parts of their data are contained in D_m, *e.g.*, employees and products, while the other parts are not constrained by D_m and are open-world, *e.g.*, product shipments.

To understand partially closed databases, relative information completeness has been proposed in [12] and studied in [13,14]. For a database D and master data D_m, a set V of *containment constraints* is used to specify that D is partially constrained by D_m. A containment constraint is of the form $q(D) \subseteq p(D_m)$, where q is a query in a language $\mathcal{L}_Q$ 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 all containment constraints in V as a *partially closed* database *w.r.t.* (D_m, V).

For a query Q and a partially closed database D *w.r.t.* master data (D_m, V), D is said to be *complete relative* to (D_m, V) if for all databases D', $Q(D) = Q(D')$ as long as $D \subseteq D'$ and D' is also partially closed *w.r.t.* (D_m, V). That is, when D_m is asserted as an "upper bound" of certain information in D, the answer to Q remains unchanged no matter how D is extended. In other words, adding tuples to D either does not change the answer to Q, or makes it no longer partially closed *w.r.t.* (D_m, V).

It is likely to find complete answer to a query in a partially closed database D, even when D is generally incomplete, as illustrated by the following example.

Example 1. Consider a (simplified) product relation of Amazon, specified by the following schema:

product(asin, brand, model, price, sale),

where each item is specified by its id (asin), brand, model and price. A flag sale indicates whether the item is on sale or not. Consider the following queries:

(1) Query Q_1 is to find all wireless reading devices that have brand = "Nook" and price ≤ 150, but are *not* on sale by Sony. The answer to Q_1 in the product relation may not be complete. Indeed, Nook is a brand of Sony, and Amazon may not carry all the products of Sony. Worse still, the answer may not even be correct: the chances are that some device found by Q_1 is actually on sale by Sony, when Amazon does not have complete information about Sony products that are on sale.

(2) Query Q_2 is the same as Q_1 except that it asks for brand = "Kindle" instead. In contrast to Q_1, we may trust the answer to Q_2 in product to be complete. That is, even though the product relation is incomplete in general, we can still find the complete answer to Q_2 in it. Indeed, "Kindle" is Amazon's own brand name, and Amazon maintains complete "master data" about its own products and their promotion sales. In other words, relative to Amazon's master data, the product relation is complete for Q_2 provided that product contains all the information relevant to "Kindle" and sales from the master data.

(3) Query Q_3 is to find all wireless reading devices with brand = "Nook" and model = "PRS-600". One can conclude that the answer to Q_3 in product is complete as long as the answer is nonempty, since (brand, model $\rightarrow$ asin, price, sale) is a functional dependency (FD) defined on product. Note that in the presence of the FD, when the answer to Q_3 is empty, we can make product complete for Q_3 by including at most one tuple with brand = "Nook" and model = "PRS-600". In Example 2, we will show that the FD given above can be expressed as three containment constraints. □

The analysis of relative information completeness has been studied in [12–14], for combined complexity. In practice, one often has to deal with a predefined set of queries. That is, the queries are fixed, and only the underlying databases vary. For instance, the queries given in Example 1 can be issued by using fixed Web forms provided by Amazon's Website. In practice, when queries are fixed, so are the associated constraints. Indeed, people typically first design constraints based on the schema of a database, and then populate and maintain database instances. This highlights the need for studying the data complexity of relative information completeness, for *a fixed set of queries and a fixed set of containment constraints*.

Contributions: Adopting the model of relative information completeness of [12,14], we study the *data complexity* of the following problems associated with relative information completeness. Let $\mathcal{L}_Q$ be a query language.

1. *The relative completeness problem* (RCP($\mathcal{L}_Q$)) is to determine, for a fixed query Q in $\mathcal{L}_Q$ and a fixed set V of containment constraints, given master data D_m and a database D partially closed *w.r.t.* D_m and V, whether D is complete for Q relative to (D_m, V). That is, we want to find out whether the answer to Q in D is complete when D is possibly incomplete.

2. *The minimal completeness problem* (MinP($\mathcal{L}_Q$)) is to decide, for a fixed query Q in $\mathcal{L}_Q$ and fixed V, given D_m and D as above, whether D is a minimal database partially closed *w.r.t.* (D_m, V) and is complete for Q relative to (D_m, V). That is, removing any tuple from D would make it incomplete for Q relative to (D_m, V). Intuitively, we want to know whether D has redundant data when answering Q is concerned.

3. *The bounded extension problem* (BEP($\mathcal{L}_Q$)) is to determine, for a fixed query Q in $\mathcal{L}_Q$ and fixed V, given D_m and D as above and a nonnegative integer K, whether there exists an extension D' of D by adding at most K tuples such that D' is partially closed *w.r.t.* D_m and V, and moreover, D' is complete for Q relative to (D_m, V). Intuitively, when D may not have complete information to answer Q, we want to know whether D can be "made" complete for Q by adding at most K tuples.

The study of these problems helps us find out whether we can get the complete answer to a set of predefined

queries in a possibly incomplete database, what excessive data is in a database for answering the queries, and how we can make a database complete for the queries by minimally extending the database.

We parameterize each of these problems with various query languages $\mathcal{L}_Q$ in which query Q and the query q of containment constraint $q(D) \subseteq p(D_m)$ in V are expressed. We consider the following $\mathcal{L}_Q$, all with equality '$=$' and inequality '$\neq$':

- conjunctive queries (CQ),
- union of conjunctive queries (UCQ),
- first-order queries (FO), and
- datalog (DATALOG).

4. *Complexity results*: We establish upper and lower bounds of these problems parameterized with these languages, *all matching*, for their *data complexity*. We show the following:

(1) It is known that the combined complexity analyses of RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$) are undecidable [12–14], when $\mathcal{L}_Q$ is FO or DATALOG. We show that fixing query Q and containment constraints V does not make our lives easier here. That is, the data complexity analyses of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) are all undecidable when $\mathcal{L}_Q$ is either FO or DATALOG. Furthermore, these complexity results are rather robust: all these problems remain undecidable for FO when master data D_m and containment constraints V are both absent, and for DATALOG when master data D_m is absent and containment constraints V are a fixed set of FDs.

(2) In contrast, when $\mathcal{L}_Q$ is CQ or UCQ, their data complexity becomes much lower: RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$) are tractable; while BEP is NP-complete, it becomes tractable when K is fixed, *i.e.*, when the number of tuples added to database D is bounded by a constant. Compare these with their combined complexity: RCP($\mathcal{L}_Q$) is Π_2^p-complete for CQ and UCQ [12,14], and MinP($\mathcal{L}_Q$) is Δ_3^p-complete for CQ and UCQ [13].

(3) The data complexity results of this paper remain unchanged no matter whether the language for expressing query q in containment constraints $q(D) \subseteq p(D_m)$ is CQ, UCQ, FO or DATALOG. Indeed, (a) RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) are undecidable for FO when master data D_m and containment constraints in V are absent, and for DATALOG when D_m is absent and V is a fixed set of FDs, while FDs can be expressed using q in CQ. (b) When $\mathcal{L}_Q$ is CQ or UCQ, the algorithms for the upper bound proofs in Section 5 have the same data complexity when q is expressed in CQ, FO or DATALOG. Indeed, checking fixed containment constraints is in PTIME no matter whether the constraints are defined with queries in FO or DATALOG. In light of this, we can assume *w.l.o.g.* that containment constraints are defined with queries in the same language that expresses query Q.

Taken together with the combined complexity bounds established in [12–14], these results provide a comprehensive picture of complexity bounds for important decision problems in connection with relatively complete information. While the combined complexity bounds of RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$) have been settled in [12] and [13], respectively, no previous work has studied their data complexity. Furthermore, we are not aware of any previous work that has considered BEP($\mathcal{L}_Q$), an interesting and practical issue. A variety of techniques are used to prove the results, including constructive proofs with algorithms and a wide range of reductions.

Related work. The model of relative information completeness was introduced in [12], which we use in this work. The combined complexity analysis of RCP($\mathcal{L}_Q$) was shown to be undecidable for FO and DATALOG, and Π_2^p-complete for CQ and UCQ in [12,14], referred to as the relatively complete database problem there. In contrast, we show that while the data complexity analysis of RCP($\mathcal{L}_Q$) remains undecidable for FO and DATALOG, it is down to PTIME for CQ and UCQ. The proofs for the data complexity bounds make use of the characterization developed in [12,14], but are more involved than their counterparts for the combined complexity. A revision of RCP($\mathcal{L}_Q$) is studied in [18] for data exchange, a very different setting; no data complexity results are given there.

The model of [12] was extended in [13] by incorporating missing values in terms of representation systems, which we do not consider in this work. The combined complexity of MinP($\mathcal{L}_Q$) was studied there, referred to as the minimality problem; it was shown to be undecidable for FO and DATALOG, and Δ_3^p−complete for CQ and UCQ. In this work we show that the data complexity analysis of MinP($\mathcal{L}_Q$) remains undecidable for FO and DATALOG, and it becomes tractable for CQ and UCQ. Again, the proofs of MinP($\mathcal{L}_Q$) in this work are rather different from their counterparts in [13].

To the best of our knowledge, no previous work has studied either the bounded extension problem BEP($\mathcal{L}_Q$) or the data complexity of RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$). A problem, referred to as the boundedness problem, was studied in [13], which is to decide, given a query Q, master data D_m and a constant K, whether there exists a partially closed database D of size K such that D is complete for Q relative to (D_m, V). Note that BEP($\mathcal{L}_Q$) takes an existing database D as input and looks for bounded extensions of D. The boundedness problem of [13] is a special case of BEP($\mathcal{L}_Q$), when D is empty. The proof of [13] for the boundedness problem does not carry over to BEP($\mathcal{L}_Q$).

A few other problems were investigated in [12–14], to decide, *e.g.*, given Q and D_m, whether there exists a partially closed database such that D is complete for Q relative to (D_m, V). We do not consider those problems in this work since their data complexity analysis is not very sensible in practice.

Several approaches have been proposed to represent or query databases with missing tuples. In [35], 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 π is the projection operator, L is the set of attributes in D, and Σ is a set of integrity constraints. Complexity bounds for computing the set of complete and consistent extensions of D *w.r.t.* Σ are established there. A notion of *open null* is introduced in [19] to model locally controlled open-world databases: parts of a database D, values or tuples, can be marked with open null and are assumed to be open-world, while the rest is closed. Relational operators are extended to tables with open null values. In contrast to [19], 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 [19].

Partially complete databases D have also been studied in [29], 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 [23] assumes that D contains some CQ views of D_c. It reduces the query answer completeness problem to the independence problem for deciding independence of queries from updates [24]. As opposed to [23,29], we assume neither D_c with complete information nor an incomplete database D containing 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 rewriting problem nor the independence problem (see below).

We now clarify the difference between our decision problems and the independence problem (*e.g.*, [9,24]). 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 Δ denotes updates generated by Q^u. In contrast, we study problems to decide, for a fixed query Q, (a) whether a given database D is relatively complete *w.r.t.* master data, where D and D_m satisfy containment constraints V; (b) whether a given D is a minimal witness for Q to be relatively complete, and (c) whether D can be minimally extended such that it is relatively complete for Q *w.r.t.* master data. Due to the difference between the problems, results for the independence problem do not carry over to ours, and vice versa.

A revision of the models of [23,12,29] has recently been introduced in [31], to study partially complete databases. The problems investigated there are quite different from $\mathsf{RCP}(\mathcal{L}_Q)$, $\mathsf{MinP}(\mathcal{L}_Q)$ and $\mathsf{BEP}(\mathcal{L}_Q)$ considered in this work.

One may also 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.*, [1,5,25,32]), to determine certain answers [1], compute complete answers from views with limited access patterns [7,25], or to decide whether views determine queries [32] or are lossless [5]. This work differs from that line of research in that one may not find a definable view 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 via containment constraints, while D_m itself may not contain the complete information of the entities that D intends to represent.

There has also been work on modeling negative information and incomplete information via logic programming (see [34] for a survey). For instance, protected circumscription is studied in [28], where databases may contain null values that are not known to be true or false under the closed world assumption. The prior work considers neither partially complete databases nor the decision problems studied in this work.

Representation systems have also been studied for incomplete information, *e.g.*, *c*-tables [20,21]. Such systems aim to represent databases with missing values rather than missing tuples (see [2,34] for surveys). Master data and the problems investigated in this work are not considered in the prior work.

There has also been recent work on consistent query answering (*e.g.*, [3,4,6]). 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 work. Note that most containment constraints in this paper are *not* expressible as integrity constraints studied for data consistency.

Organization. Section 2 reviews the model of relative completeness. Section 3 states the decision problems studied in this paper. Section 4 provides the undecidability results for FO and DATALOG, followed by the decidable cases for CQ and UCQ in Section 5. Finally, Section 6 summarizes the main results of the paper and identifies open questions.

2. Relative information completeness

In this section, we review the model of relative completeness proposed in [12]. We start with basic notations.

Databases and master data. A database is specified by a relational schema $\mathcal{R}$, which is a collection of relation schemas $(R_1, \ldots, R_n)$. Each schema R_i in $\mathcal{R}$ is defined over a fixed set of attributes. For each attribute A of R, its domain is specified in R, denoted by $\mathsf{dom}(A)$. To simplify the discussion we assume that all attributes have a countably infinite domain $\mathbf{d}$, a setting commonly adopted in database theory (see, *e.g.*, [2]).

A relation (instance) over a relation schema $R(A_1, \ldots, A_m)$ is a finite set I of m-arity tuples $t(a_1, \ldots, a_m)$ such that for each $i \in [1, m]$, a_i is in $\mathsf{dom}(A_i)$. A database (instance) over a relational schema $\mathcal{R} = (R_1, \ldots, R_n)$ is a collection of finite sets $(I_1, \ldots, I_n)$, where each I_i is a relation over R_i.

We will use the following notion. Consider instances $D = (I_1, \ldots, I_n)$ and $D' = (I'_1, \ldots, I'_n)$ of the same schema $\mathcal{R}$. We say that D is *contained in* D', denoted by $D \subseteq D'$, if $I_j \subseteq I'_j$ for all $j \in [1, n]$. If $D \subseteq D'$, we say that D' is an *extension* of D.

Master data (*a.k.a.* reference data) D_m is specified by a relational schema $\mathcal{R}_m$. As remarked earlier, an enterprise typically maintains master data that is assumed to be consistent and complete about certain information of the enterprise [8,30]. We do not impose any restriction on the relational schemas $\mathcal{R}$ and $\mathcal{R}_m$.

Partially closed database. Databases D are usually partially constrained by master data D_m. We specified such relationship between D and D_m in terms of *containment constraints* (CCs). Let $\mathcal{L}_C$ be a query language. A CC ϕ in $\mathcal{L}_Q$ is of the form

$$q(\mathcal{R}) \subseteq p(\mathcal{R}_m),$$

where q is a query in $\mathcal{L}_Q$ defined over schema $\mathcal{R}$, and p is a projection query over schema $\mathcal{R}_m$. That is, p is a query

of the form $\exists \vec{x}\ R_i^m(\vec{x}, \vec{y})$ for some relation schema R_i^m in $\mathcal{R}_m$.

Intuitively, constraint ϕ assures that D_m is an "upper bound" of the information extracted by $q(D)$. In other words, the CWA is asserted for D_m, which constrains the part of data identified by $q(D)$ from D. More specifically, while this part of D can be extended, the expansion cannot go beyond the information already in D_m. On the other hand, the OWA is assumed for the part of D that is not constrained by any CC ϕ.

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

Example 2. Recall schema product described in Example 1. Suppose that there exists a master relation product_m specified by schema $\text{R}_m(\text{asin, model, price, sale})$, which maintains a complete record of Kindle products. We specify a CC $q(\text{product}) \subseteq \text{R}_m$, where $q(\text{product})$ is a query defined as $q(a, m, p, s) = \exists b(\text{product}(a, b, m, p, s) \wedge b =$ 'Kindle'). This CC assures that product_m is an upper bound on the Kindle product information possibly contained in relation product.

As shown in [12,13] and as will be seen shortly, many integrity constrains commonly used in practice can be expressed as CCs. For example, consider a functional dependency (FD) ψ: (brand, model → asin, price, sale), which assures that if two products have the same brand and model, then they refer to the same item with the same id, price and status of sale. Assume that there exists an empty relation $\text{product}_\varnothing$ in master data D_m. Then ψ can be written as CCs included in V:

$q_{\text{asin}}(\text{product}) \subseteq \text{product}_\varnothing$,
$q_{\text{price}}(\text{product}) \subseteq \text{product}_\varnothing$,
$q_{\text{sale}}(\text{product}) \subseteq \text{product}_\varnothing$,

where

$$q_{\text{asin}}(b, m) = \exists a_1, a_2, p_1, p_2, s_1, s_2\ (\text{product}(a_1, b, m, p_1, s_1) \wedge \text{product}(a_2, b, m, p_2, s_2) \wedge a_1 \neq a_2),$$

which detects violations of FD (brand, model → asin); similarly one can specify the other CCs $q_{\text{price}}(\text{product}) \subseteq \text{product}_\varnothing$ and $q_{\text{sale}}(\text{product}) \subseteq \text{product}_\varnothing$. □

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

A database D is said to be a *partially closed w.r.t.* (D_m, V) if $(D, D_m) \models V$. That is, the information in D is partially bounded by D_m via the CCs in V.

A database D' is *a partially closed extension* of D *w.r.t.* (D_m, V) if $D \subseteq D'$ and D' is partially closed *w.r.t.* (D_m, V).

Relative completeness: We are now ready to introduce the notion of relative information completeness. Consider a database D of schema $\mathcal{R}$, master data D_m of schema $\mathcal{R}_m$ and a set V of CCs, such that D is partially closed *w.r.t.* (D_m, V).

We say that D is *complete for query* Q *relative to* (D_m, V) if $Q(D) = Q(D')$ for every *partially closed extension* D' of D, *i.e.*, $D \subseteq D'$ such that $(D', D_m) \models V$. The *set of complete databases* for Q *w.r.t.* (D_m, V), denoted by $\text{RCQ}(Q, D_m, V)$, is the set of all complete databases for Q relative to (D_m, V).

Intuitively, if D is complete for Q relative to (D_m, V), then no matter how D is expanded by including new tuples, as long as the extension does not violate containment constraints V, the answer to query Q remains unchanged. In other words, D has already got complete information for answering Q.

To simplify the discussion, we assume that query Q and the CCs in V are expressed in the same language $\mathcal{L}_Q$. As remarked in Section 1, this does not lose generality. All the results of this paper remain the same if Q and V are expressed in the different languages CQ, UCQ, FO or DATALOG.

Example 3. Recall the Amazon instance of product (also referred to as product), queries Q_1, Q_2 and Q_3 from Example 1, and master data product_m and CCs V from Example 2. As shown in Example 1, product is complete for Q_2 relative to $(\text{product}_\text{m}, V)$ if $Q_2(\text{product})$ returns all wireless reading devices in product_m with brand = "Kindle" and price ≤ 150.

Consider Q_3, to find all wireless reading devices with brand = "Nook" and model = "PRS-600". Suppose that there exist such device records in product_m, but $Q_3(\text{product})$ is empty. Then product is not complete for Q_3. Nonetheless, we can make product complete for Q_3 by adding at most one product with brand = "Nook" and model = "PRS-600". Indeed, V includes the CCs encoding the FD ψ, assuring that there exists at most one product with this brand and model. Thus the expanded product is complete for Q_3 relative to $(\text{product}_\text{m}, V)$.

In contrast, consider Q_1, to find all wireless reading devices that have brand = "Nook" and price ≤ 150, but are *not* on sale by Sony. Then master data product_m does not help when we want to make product complete: product_m has no complete information about Sony products with brand = "Nook". In this case we cannot make product complete for Q_1 relative to $(\text{product}_\text{m}, V)$ by adding tuples of product_m to product. □

Relative completeness and consistency. Several classes of constraints have been used to capture inconsistencies in relational data (see *e.g.*, [6,10] for recent surveys), notably denial constraints, conditional functional dependencies (CFDs, which are an extension of functional dependencies (FDs)), and conditional inclusion dependencies (CINDs, which are an extension of inclusion dependencies (INDs)). As shown in [12,14], denial constraints and CFDs can be expressed as CCs in CQ, and CINDs can be expressed as CCs in FO. Moreover, in all three cases only an empty master data relation is required. This allows us to capture both data consistency and relative information completeness in a uniform logic framework [14].

3. Determining relative information completeness

In this section, we formulate three decision problems in connection with relative complete databases, each of them parameterized by a query language $\mathcal{L}_Q$. Consider a query $Q \in \mathcal{L}_Q$, master data D_m, a set V of CCs defined in terms of queries in $\mathcal{L}_Q$, and a partially closed database D *w.r.t.* (D_m, V).

The first problem is referred to as the *relative completeness problem*. It is to decide whether a given D is complete for a query Q relative to (D_m, V). The need for studying this problem is evident: one naturally wants to know whether one can trust their databases to yield complete answers to queries.

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

To decide what data should be collected in a database in order to answer a query Q, we want to identify a minimal amount of information that is complete for Q. To capture this, we use a notion of minimality given as follows.

A database D is called *a minimal database complete for a query Q relative to* (D_m, V) if it is in RCQ(Q, D_m, V) and moreover, for any $D' \subsetneq D$, D' is not in RCQ(Q, D_m, V).

These suggest that we study the following problem, referred to as the *minimal completeness problem.*

MinP($\mathcal{L}_Q$): *The minimal completeness problem*
INPUT: Q, D_m, V, D as in RCP.
QUESTION: Is D a minimal database complete for Q relative to (D_m, V)?

When a database D is not complete for Q, one naturally wants to extend D with minimal information to make it complete. We use ΔD to denote a set of tuples to be inserted into D and $D \cup \Delta D$ to denote the database obtained by adding all tuples of ΔD to D. Given a positive integer $K \geq 1$, we call ΔD a *bounded set of updates* for (Q, D_m, V, D, K) if (a) $|\Delta D| \leq K$, and (b) $D \cup \Delta D$ is complete for Q relative to (D_m, V).

There is a practical need for studying the following problem, referred to as the *bounded extension problem*. Indeed, this problem may assist practitioners to identify how much additional data needs to be collected to make the database complete for Q.

BEP($\mathcal{L}_Q$): *The bounded extension problem*
INPUT: Q, D_m, V and D as in RCP, and a positive integer $K \geq 0$.
QUESTION: Does there exist a bounded set of updates ΔD for (Q, D_m, V, D, K)?

Query languages. We study these problems when $\mathcal{L}_Q$ ranges over the following query classes (see, *e.g.*, [2], for the details):

(1) 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 $\exists$;

(2) union of conjunctive queries (UCQ) of the form $Q_1 \cup \cdots \cup Q_k$, where for each $i \in [1, k]$, Q_i is in CQ;

(3) first-order logic queries (FO) built from atomic formulas using $\wedge$, $\vee$, *negation* $\neg$, $\exists$ and universal quantification $\forall$; and

(4) datalog queries (DATALOG), defined as a collection of rules $p(\overline{x}) \leftarrow p_1(\overline{x}_1), \ldots, p_n(\overline{x}_n)$, where each p_i is either an atomic formula (a relation atom in $\mathcal{R}$, $=$, $\neq$) or an IDB predicate.

One might also want to consider positive existential FO queries ($\exists$FO$^+$), which is built from atomic formulas by closing under $\wedge$, *disjunction* $\vee$ and $\exists$. Note that any *fixed* $\exists$FO$^+$ query can be unfolded into a UCQ in constant time. Thus all the complexity results of this paper for UCQ carry over to $\exists$FO$^+$.

As remarked earlier, we express both the user's query Q and CCs of V in the same query language $\mathcal{L}_Q$, with $\mathcal{L}_Q$ as one of the languages given above.

Data complexity. In the rest of the paper, we investigate the data complexity of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$), *i.e.*, when *both the query Q and the set V of* CCs are *predefined and fixed*, while databases D and master data D_m may vary (see, *e.g.*, [2] for details about data complexity). As mentioned earlier, in practice the containment constraints are often predefined, and users execute a fixed set of queries, while the underlying database D and master data D_m may vary from time to time. We establish the complexity of these problems in this setting, when $\mathcal{L}_Q$ ranges over all the query languages given above.

4. Undecidability results for FO and DATALOG

In this section we establish the *data complexity* of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) when $\mathcal{L}_Q$ is either FO or DATALOG.

It is known that for the *combined complexity*, RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$) are undecidable when $\mathcal{L}_Q$ is FO or DATALOG[12–14]. One might think that fixing queries and containment constraints would make our lives easier. The results in this section tell us, however, that these two problems remain undecidable when data complexity is concerned (Theorems 1 and 2). Furthermore, we also show that BEP($\mathcal{L}_Q$) is undecidable when $\mathcal{L}_Q$ is FO or DATALOG (Theorem 3).

In addition, the undecidability results are rather robust: RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) remain undecidable for FO even in the absence of both master data D_m and containment constraints V; moreover, they are undecidable for DATALOG when D_m is absent and V is a fixed set of FDs, which can be expressed as CCs in CQ (see Example 2 and [12–14]).

In fact, we show the undecidability of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) for these special cases in Theorems 1, 2 and 3, respectively. Clearly, this implies the undecidability for the general case of these problems.

Deciding relative completeness. We start with RCP($\mathcal{L}_Q$), the relative completeness problem. We show that for the data complexity analysis, RCP($\mathcal{L}_Q$) is undecidable when $\mathcal{L}_Q$ is FO or DATALOG. The proofs of the undecidability of the data complexity analyses are rather different from their combined complexity counterparts given in [12–14].

Theorem 1. *The data complexity of* RCP($\mathcal{L}_Q$) *is undecidable when $\mathcal{L}_Q$ is* FO *or* DATALOG. *The problem remains undecidable*

- *for* FO, *even when master data D_m and containment constraints V are absent*; *and*
- *for* DATALOG, *even when D_m is absent and V is a fixed set of FDs.* □

Proof. We first settle the data complexity of RCP($\mathcal{L}_Q$) when $\mathcal{L}_Q$ is FO, and then consider RCP($\mathcal{L}_Q$) when $\mathcal{L}_Q$ is DATALOG.

When $\mathcal{L}_Q$ is FO. We show that RCP($\mathcal{L}_Q$) is undecidable by reduction from the embedding problem for the class of all

finite semigroups, which is known to be undecidable [22]. To formulate the embedding problem we need the following notions.

A *semigroup* $\mathcal{A}$ is a structure of the form $\mathcal{A}=(A,f)$ such that A is a nonempty set, called the domain of $\mathcal{A}$, and f is an associative binary function on A; this means that, for every a, b, $c\in A$, we have that $f(f(a,b),c)=f(a,f(b,c))$. A *finite semigroup* is a semigroup whose domain is a finite set. A *partial semigroup* is a structure $\mathcal{B}$ of the form $\mathcal{B}=(B,g)$ where, as before, B is a nonempty set but now g is a *partial* binary function that is associative. Let $\mathcal{B}=(B,g)$ be a partial finite semigroup and $\mathcal{A}=(A,f)$ a finite semigroup. We say that $\mathcal{B}$ is *embeddable* in $\mathcal{A}$ if $B\subseteq A$ and f is an extension of g, that is, whenever $g(b_1,b_2)$ is defined, we have that $f(b_1,b_2)=g(b_1,b_2)$.

The *embedding problem* for finite semigroups is to decide whether a given partial finite semigroup is embeddable in some finite semigroup. This problem is undecidable [22].

Given a finite partial semigroup $\mathcal{B}=(B,g)$, we define a *fixed* relational schema $\mathcal{R}$, a database D on $\mathcal{R}$, a *fixed* FO query Q such that D is partially closed *w.r.t.* (D_m,V), where D_m and V are both *empty*. We show that $D\in \mathsf{RCQ}(Q,D_m=\emptyset,V=\emptyset)$ if and only if $\mathcal{B}$ is not embeddable.

(1) Let $\mathcal{R}$ consist of a single schema $R_g(A,X,Y,Z)$, where attributes A,X,Y and Z have a countably infinite domain, and D consist of a single relation I_g over R_g, which is defined as follows. For any three elements a,b and c in B, there exists a tuple $(0,a,b,c)$ in I_g if $g(a,b)=c$. Intuitively, I_g encodes the function g of the finite partial semigroup $\mathcal{B}$. Extensions g' of g are encoded by extensions I'_g of I_g by means of tuples of the form $(1,a',b',c')$ such that $g'(a',b')=c'$.

We say that an instance I'_g of R_g is *well-formed* if (a) each tuple of the form $(0,a,b,c)$ in I'_g has a counterpart of the form $(1,a,b,c)$ in I'_g; and (b) $I'_g(1,x,y,z)$ encodes an associative binary function f such that $z=f(x,y)$. Obviously, an extension I'_g of I_g that is well-formed encodes an extension of g that is an associative binary function.

(2) The query Q is a boolean query that encodes the conditions (a) and (b) given above. It returns true on an instance of R_g if and only if this instance is well-formed. More specifically, Q is the conjunction of sub-queries Q_1, Q_2, Q_3, and Q_4, which are defined as follows:

$$
\begin{aligned}
Q_1 &= \forall x,y,z\ (R_g(0,x,y,z)\rightarrow R_g(1,x,y,z)),\\
Q_2 &= \forall x,y,z,z'\ (R_g(1,x,y,z)\wedge R_g(1,x,y,z')\rightarrow z=z'),\\
Q_3 &= \forall x,y,z,u,v,w\ (R_g(1,x,y,u)\wedge R_g(1,y,z,v)\wedge\\
&\qquad R_g(1,u,z,w)\rightarrow R_g(1,x,v,w)),\\
Q_4 &= \forall x,y,z,x',y',z'\ (R_g(1,x,y,z)\wedge R_g(1,x',y',z')\rightarrow\\
&\qquad \exists w_1,\ldots,w_9\ (R_g(1,x,x',w_1)\wedge R_g(1,x,y',w_2)\\
&\qquad \wedge R_g(1,x,z',w_3)\wedge R_g(1,y,x',w_4)\wedge R_g(1,y,y',w_5)\wedge\\
&\qquad R_g(1,y,z',w_6)\wedge R_g(1,z,x',w_7)\wedge R_g(1,z,y',w_8)\wedge\\
&\qquad R_g(1,z,z',w_9))).
\end{aligned}
$$

Clearly, for any database $D'=(I'_g)$ on $\mathcal{R}$, $Q_1(D')\neq\emptyset$ if and only if the condition (a) given above is satisfied, and $Q_2(D')\neq\emptyset$ if and only if the subset $I'_g(1,x,y,z)$ encodes a function. Furthermore, for such databases D', $Q_3(D')\neq\emptyset$ if and only if $I'_g(1,x,y,z)$ encodes an associative function. Finally, $Q_4(D')\neq\emptyset$ if and only if for any two elements that occur in two triples in $I'_g(1,x,y,z)$, function f is defined on the values of these elements and is encoded in $I'_g(1,x,y,z)$. In other words, $Q_4(D')\neq\emptyset$ if and only if $I'_g(1,x,y,z)$ encodes a total function. Hence, $Q(D')\neq\emptyset$ if and only if the set $I'_g(1,x,y,z)$ encodes an associative binary function f such that $f(x,y)=z$, and moreover, it is an extension of g.

Observe that since V is empty, D is partially closed *w.r.t.* (D_m,V) and so is any D' of $\mathcal{R}$ such that $D\subseteq D'$. Furthermore, $Q(D)=\emptyset$ since $Q_1(D)=\emptyset$ by the definition of D.

We next show that we have indeed defined a reduction, *i.e.*, $D\in\mathsf{RCQ}(Q,D_m=\emptyset,V=\emptyset)$ if and only if $\mathcal{B}$ cannot be embedded in a finite semigroup.

($\Rightarrow$) First assume that $D\in\mathsf{RCQ}(Q,\emptyset,\emptyset)$. Then for each partially closed extension D' of D, $Q(D')=Q(D)=\emptyset$. Suppose by contradiction that there exists a finite subgroup $\mathcal{A}=(A,f)$ such that $\mathcal{B}$ can be embedded in $\mathcal{A}$. Let I'_g be an instance of R_g such that $(0,a,b,c),(1,a,b,c)\in I'_g$ if and only if $f(a,b)=c$. Let $D'=(I'_g)$. It is easy to see that D' is an extension of D since $\mathcal{B}$ can be embedded in $\mathcal{A}$. Moreover, one can readily verify that $Q(D')\neq\emptyset$ by the definition of Q. Obviously, as discussed above, D' is a partially closed extension of D since V is empty. This contradicts the assumption that $D\in\mathsf{RCQ}(Q,\emptyset,\emptyset)$.

($\Leftarrow$) Conversely, assume that $D\notin\mathsf{RCQ}(Q,\emptyset,\emptyset)$. Then there exists a partially closed extension $D'=(I'_g)$ of D such that $Q(D')\neq\emptyset$. Thus $I'_g(1,x,y,z)$ encodes an associative binary function g' that is an extension of g, *i.e.*, for each $a,b\in B$, $g'(a,b)=g(a,b)$ if $g(a,b)$ is defined. We next construct a semigroup $\mathcal{A}=(A,f)$ such that $\mathcal{B}$ can be embedded in $\mathcal{A}$. Note that I_g is defined in terms of the function g and that even though I'_g encodes a total function, I'_g may not contain all values in B.

Given I'_g, we therefore let A consist of (i) all elements in B, (ii) all values of attributes X,Y or Z that appear in a tuple of the form $(1,a,b,c)$ in I'_g, and (iii) a fresh constant ϵ that does not appear in B or I'_g. Moreover, we define a function f such that for each pair of elements a and b in A, (a) $f(a,b)=c$ if $g'(a,b)=c$ for some $c\in A\backslash\{\epsilon\}$; (b) $f(a,b)=\epsilon$ if $a\neq\epsilon$, $b\neq\epsilon$, and $g'(a,b)$ is not defined (*i.e.*, $a,b\in B$, and $g(a,b)$ and $g'(a,b)$ are both undefined); and (c) $f(a,b)=a$ if $b=\epsilon$ and $f(a,b)=b$ if $a=\epsilon$. Obviously, by the definition of A and f, we have that $B\subseteq A$ and f is an extension of g. Moreover, one can readily verify that f is an associative binary function on A. Thus $\mathcal{A}$ is a semigroup and $\mathcal{B}$ can be embedded in $\mathcal{A}$.

When $\mathcal{L}_Q$ is DATALOG. We show that RCP(DATALOG) is undecidable by reduction from the emptiness problem for deterministic finite 2-head automata, which is known to be undecidable [33]. Our proof closely follows the reduction presented in [33, Theorem 3.4.1], which shows that the satisfiability of the existential fragment of transitive-closure logic, E+TC, is undecidable over a schema having at least two non-nullary relation schemas, one of them being a function symbol. Although E+TC allows the negation of atomic expression as opposed to DATALOG, the undecidability proof only uses a very restricted form of negation, which can be simulated using $\neq$ and a fixed set of FDs.

For readers' convenience, we present necessary definitions taken from [33]. A *deterministic finite 2-head automaton* (or 2-head DFA for short) is a quintuple $\mathcal{A}=(S,\Sigma,\Gamma,s_0,s_{acc})$ consisting of a finite set of states S, an input alphabet $\Sigma=\{0,1\}$, an initial state s_0, an accepting

state s_{acc}, and a transition function Γ: $S \times \Sigma_\epsilon \times \Sigma_\epsilon \to S \times \{0,+1\} \times \{0,+1\}$, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. A *configuration of* $\mathcal{A}$ is a triple $(s,\omega_1,\omega_2) \in S \times \Sigma^* \times \Sigma^*$, representing that $\mathcal{A}$ is in state s, and the first head and the second head of $\mathcal{A}$ are positioned on the first symbol of ω_1 and ω_2, respectively. On an input string $\omega \in \Sigma^*$, $\mathcal{A}$ starts from the initial configuration (s_0,ω,ω); and the successor configuration is defined as usual.

We say that $\mathcal{A}$ *accepts* ω if a configuration $(s_{acc},\omega_1,\omega_2)$ can be reached, based on the successor relation, from the initial configuration for (s_0,ω,ω); otherwise we say that $\mathcal{A}$ rejects ω. The *language accepted by* $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, consists of all strings that are accepted by $\mathcal{A}$. The *emptiness problem for 2-head* DFAs is to determine, given a 2-head DFA $\mathcal{A}$, whether $\mathcal{L}(\mathcal{A})$ is empty. This problem is known to be undecidable [33].

Given a 2-head DFA $\mathcal{A} = (S,\Sigma,\Gamma,s_0,s_{acc})$, we define a *fixed* relational schema $\mathcal{R}$, empty master schema $\mathcal{R}_m$, a database D on $\mathcal{R}$, a *fixed* DATALOG-query Q, a *fixed* set V of FDs and *empty* master data D_m. We show that $\mathcal{L}(\mathcal{A})$ is empty if and only if $D \in \mathrm{RCQ}(Q, D_m = \varnothing, V)$.

(1) Let $\mathcal{R}$ consist of four relation schemas $R_P(U,A)$, $R_F(W,A_1,A_2)$, $R_T(B_1,B_2,S_1,In_1,In_2,S_2,M_1,M_2)$ and $R_C(C_1,C_2)$, where all attributes in $\mathcal{R}$ have a countably infinite domain. Intuitively, instances I'_P and I'_F of R_P and R_F, respectively, are to represent a string $\omega \in \Sigma^*$ such that (i) elements in $\sigma_{U=1}(I'_P)$ represent the positions in ω where an 1 occurs, (ii) $\sigma_{U=0}(I_{P'})$ records those positions in ω that are 0; and (iii) I'_F is to represent a successor relation over these positions. More specifically, the successor relation will be given by $\pi_{A_1,A_2}(\sigma_{A_1 \neq A_2}(I'_F)) \cup \pi_{A_1,A_2}(\sigma_{A_1 = A_2 \wedge W=1}(I'_F))$ in which the last part identifies the final position in the successor relation. This will be further explained when considering the CCs below. Furthermore, the instance I_T of R_T is to encode thetransitions in Γ of $\mathcal{A}$. More specifically, for each transition Γ: $(s,\mathsf{in}_1,\mathsf{in}_2) \to (s',\mathsf{move}_1,\mathsf{move}_2)$, there exists a tuple $(b_1,b_2,s,\mathsf{in}_1,\ \mathsf{in}_2,s',\ \mathsf{move}_1,\mathsf{move}_2)$ in I_T, such that the first two attributes of all tuples in I_T result in a sequence $0 \to 1 \to \cdots \to n$, where n is the number of transition in Γ. That is, $\pi_{B_1,B_2}(I_T)$ consists of all tuples $(0,1),(1,2),\ldots,(n-1,n)$. We set $I_C = \{(0,n)\}$. We define $D = (I_P,I_F,I_T,I_C)$, where I_P and I_F are *empty* instances of R_P and R_F, respectively, which encode an *empty* string, and I_T and I_C are defined above.

(2) The set V consists of five FDs to assure that we only consider *well-formed* instances of $\mathcal{R}$. An instance $D' = (I'_P,I'_F,I'_T,I'_C)$ of $\mathcal{R}$ is well-formed if (a) $\sigma_{U=1}(I'_P)$ and $\sigma_{U=0}(I'_P)$ are disjoint (*i.e.*, a string can only have one letter at each position); and $\pi_{A_1,A_2}(\sigma_{A_1 \neq A_2}(I'_F)) \cup \pi_{A_1,A_2}(\sigma_{A_1 = A_2 \wedge W=1}(I'_F))$ must (b) be a function and (c) contain a unique tuple of the form (k,k) for some constant k indicating the *final* position. We additionally require that I'_F contains a tuple of the form $(w,0,i)$, where 0 represents the *initial* position and i is some constant. Similarly, we require the presence of a tuple $(1,k,k)$ in I'_F representing the *final* position, where k is some constant. These two extra requirements will be assured by the DATALOG-queries Q_{ini} and Q_{fin} to be defined shortly. Furthermore, (d) $\pi_{C_2}\sigma_{C_1=0}I'_C(C_1,C_2)$ is to contain a single value only, (e) $\pi_{B_1,B_2}(I'_T)$ encodes a bijection, and finally, (f) there is a unique transition in I'_T for each value in $\pi_{B_1}(I'_T)$. More specifically, the set V consists of the following FDs:

- $A \to U$, enforcing that for any instance $D' = (I'_P,I'_F,I'_T,I'_C)$ of $\mathcal{R}$ such that $\mathcal{I}' \models V$, condition (a) is satisfied for I'_P;
- $A_1 \to A_2$, ensuring that $\pi_{A_1,A_2}(I'_F)$ encodes a function; hence condition (b) is satisfied;
- $W \to A_1,A_2$, ensuring that there can be at most one tuple with its W-attribute set to 1 in I'_F. As a result, $\pi_{A_1,A_2}(\sigma_{A_1=A_2 \wedge W=1}(I'_F))$ contains at most one tuple, and condition (c) is satisfied;
- $C_1 \to C_2$, ensuring that $\pi_{C_2}\sigma_{C_1=0}I'_C(C_1,C_2)$ consists of a single value only, ensuring that (d) is satisfied;
- $B_1 \to B_2$ and $B_2 \to B_1$, ensure that $\{(b_1,b_2)|\pi_{B_1,B_2}(I'_T)\}$ is bijection from $\pi_{B_1}(I'_T)$ to $\pi_{B_2}(I'_T)$, and hence condition (e) is satisfied; and finally,
- $B_1 \to B_2,S_1,In_1,In_2,S_2,M_1,M_2$, ensuring that condition (f) is satisfied.

Recall that FDs can be encoded by CCs in CQ together with an empty master database (Example 2 and [12–14]).

In summary, any instance $D' = (I'_P,I'_F,I'_T,I'_C)$ of $\mathcal{R}$ that satisfies V 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'_F of R_F in D'. Obviously, we have that $(D,D_m) \models V$.

(3) We next define the query Q. To do this, we first give some auxiliary DATALOG queries, and then show how the non-emptiness of $\mathcal{L}(\mathcal{A})$ can be expressed in terms of these queries. Let $\Pi_P(u,a) \leftarrow R_P(u,a), u=0$ and $\Pi_P(u,a) \leftarrow R_P(u,a)$, $u=1$. Furthermore, let $\Pi_F(a_1,a_2) \leftarrow R_F(w,a_1,a_2), a_1 \neq a_2$ and $\Pi_F(a_1,a_2) \leftarrow R_F(w,a_1,a_2), a_1 = a_2, w=1$. These DATALOG-queries are to extract the strings and successor relation on strings from the database instances. Let $\mathsf{TC}(b_1,b_2) \leftarrow R_T(b_1,b_2,s,\mathsf{in}_1,\mathsf{in}_2,s',\mathsf{move}_1,\mathsf{move}_2)$ and $\mathsf{TC}(b_1,b_2) \leftarrow \mathsf{TC}(b_1,b_3)$, $\mathsf{TC}(b_3,b_2)$. That is, TC contains the transitive closure of $\pi_{B_1,B_2}(R_T)$. We define

$$\Pi_{post}(b_2) \leftarrow \mathsf{TC}(b_1,b_2), b_1 = 0$$
$$\Pi_{pre}(b_2) \leftarrow \mathsf{TC}(b_1,b_2), R_C(c_1,b_2), c_1 = 0,$$

and define $\Pi_\Gamma(s,\mathsf{in}_1,\mathsf{in}_2,s',\mathsf{move}_1,\mathsf{move}_2)$ as

$$R_T(s,\mathsf{in}_1,\mathsf{in}_2,s',\mathsf{move}_1,\mathsf{move}_2), \Pi_{post}(b_2), \Pi_{pre}(b_1).$$

It can be readily verified that for each extension $D' = (I'_P,I'_F,I'_T,I'_C)$ of D, if $(D',D_m) \models V$ then $\Pi_\Gamma(D')$ returns exactly all tuples in I_T. Indeed, this follows from the fact that by $(D',D_m) \models V$, $\pi_{B_1,B_2}(I'_T)$ encodes a bijection; Π_{pre} returns all transitions reachable from 0; Π_{post} returns all transitions that can reach n; and that I'_T contains a unique transition for each B_1-value. Here n is the number of transitions in Γ.

Finally, from Π_Γ we construct the following queries to represent how $\mathcal{A}$ run on the string encoded by I'_P and I'_F: for each $i_1 \in \{\epsilon,0,1\}$, $i_2 \in \{\epsilon,0,1\}$, $m_1 \in \{0,+1\}$, and $m_2 \in \{0,+1\}$,

$$\Pi_{i_1,i_2,m_1,m_2}(x,y,z,x',y',z') \leftarrow \Pi_\Gamma(x,i_1,i_2,x',m_1,m_2),$$
$$\psi_{i_1,i_2,m_1,m_2}(y,z,y',z'),$$

where

$$\psi_{i_1,i_2,m_1,m_2}(y,z,y',z') \leftarrow \alpha_1(i_1,y),\alpha_2(i_2,z),$$
$$\beta_1(m_1,y,y'),\beta_2(m_2,z,z'),$$

and $\alpha_1(i_1,y) \leftarrow \Pi_F(y,y'),\Pi_P(i_1,y), y \neq y'$ if $i_1 = 0,1$; and $\alpha_1(i_1,y) \leftarrow \Pi_F(y,y)$ if $j = \epsilon$; similarly for $\alpha_2(i_2,z)$.

Furthermore, $\beta_1(m_1, y, y') \leftarrow \Pi_F(y, y')$ if $m_1 = +1$ and $\beta_1(m_1, y, y') \leftarrow y = y'$ if $m_1 = 0$; similarly for $\beta_2(m_2, z, z')$.

Intuitively, $\alpha_i(j, y)$ enforces y to be a position in the string coded by $\Pi_P(1, y)$ (when $j=1$) or $\Pi_P(0, y)$ (when $j=0$) that has a successor, unless y is the final position (when $j = \epsilon$), where $\alpha_i(j, y)$ demands $\Pi_F(y, y)$. Moreover, $\beta_i(y, y')$ ensures that y and y' are consecutive positions when $\mathcal{A}$ makes a move (with head i) and $y = y'$ otherwise.

Putting these together, $\psi_{i_1,i_2,m_1,m_2}(y, z, y', z')$ expresses valid moves of $\mathcal{A}$ on the string encoded by I'_P and I'_F. Then,

$$\Pi_{trans}(x, y, z, x', y', z') \leftarrow \bigwedge_{i_1,i_2,m_1,m_2} \Pi_{i_1,i_2,m_1,m_2}(x, y, z, x', y', z')$$

$$\Pi_{trans}(x, y, z, x', y', z') \leftarrow \Pi_{trans}(x, y, z, x'', y'', z''),$$
$$\Pi_{trans}(x'', y'', z'', x', y', z')$$

represents all possible valid transitions in $\mathcal{A}$; hence, the query

$$Q'() = \exists y_1 y_2 \Pi_{trans}(q_0, 0, 0, q_{acc}, y_1, y_2)$$

is satisfiable if and only if $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

Clearly, we can express Q' in DATALOG. Recall that we still need to assure the existence of an initial and a final position in well-formed instance of R_F. The final DATALOG-query Q is therefore defined as the conjunction of $Q'()$, Q_{ini} and Q_{fin}, where $Q_{\text{ini}}() \leftarrow R_F(w, 0, x)$ and $Q_{\text{fin}}() \leftarrow R_F(1, x, x)$ so that initial and final positions in I_P and I_F are also checked.

We next show that it is indeed a reduction. Recall that $(D, D_m) \models V$; since $Q_{\text{fin}}(D) = \emptyset$, we have that $Q(D) = \emptyset$. It remains to show that $L(\mathcal{A}) = \emptyset$ if and only if for each partially closed extension $D' = (I'_P, I'_F, I'_T, I'_C)$ of D, $Q(D') = \emptyset$. Observe that for such D', the addition of extra tuples in I_T does not affect the query results since Q only selects tuples already in I_T, and V does not allow the addition of other tuples in I_C representing the number of transitions in Γ. Thus I'_P and I'_F encode a string ω such that $Q(D')$ is nonempty if and only if $\omega \in L(\mathcal{A})$. As a result, $\mathcal{L}(A) = \emptyset$ if and only if for each partially closed extension D' of D, $Q(D') = \emptyset$, *i.e.*, $D \in \text{RCQ}(Q, \emptyset, V)$.

This completes the proof of Theorem 1. □

Determining minimal completeness. When it comes to the minimal complete problem $\text{MinP}(\mathcal{L}_Q)$, we show that it is also beyond reach in practice when $\mathcal{L}_Q$ is FO or DATALOG. Indeed, we get results similar to Theorem 1: the data complexity of MinP(FO) is undecidable in the absence of master data D_m and CCs V (*i.e.*, $D_m = \emptyset$ and $V = \emptyset$); and moreover, MinP(DATALOG) is undecidable even when D_m is absent and V is a fixed set of FDs (*i.e.*, V can be expressed in CQ).

Theorem 2. *The data complexity of* $\text{MinP}(\mathcal{L}_Q)$ *is undecidable when* $\mathcal{L}_Q$ *is* FO *or* DATALOG. *The problem remains undecidable*

- *for* FO, *even when both the master data* D_m *and containment constraints* V *are empty*; *and*
- *for* DATALOG, *even when* D_m *is empty and* V *is a fixed set of FDs*. □

Proof. We first study $\text{MinP}(\mathcal{L}_Q)$ when $\mathcal{L}_Q$ is FO, and then investigate it when $\mathcal{L}_Q$ is DATALOG.

When $\mathcal{L}_Q$ *is* FO. We show that MinP(FO) is undecidable by *Turing reduction* from RCP(FO) to MinP(FO). By Theorem 1, RCP(FO) is undecidable even when master data D_m and containment constraints V are absent. We consider such special case of RCP(FO) in the reduction. To give the reduction, we first show the following lemma.

Lemma 1. *For any* FO *query* Q, *empty master data* D_m, *empty* V *of* CCs, *and any database* D *that is partially closed w.r.t.* (D_m, V), $D \in \text{RCQ}(Q, D_m = \emptyset, V = \emptyset)$ *if and only if there exists a database* $D_0 \subseteq D$ *such that* D_0 *is a minimal database complete for* Q *relative to* $(D_m = \emptyset, V = \emptyset)$. □

Lemma 1 can be easily verified as follows. First, assume that $D \in \text{RCQ}(Q, D_m = \emptyset, V = \emptyset)$. Then there must be $D_0 \subseteq D$ such that D_0 is a minimal database complete for Q relative to (D_m, V), by the definition of minimal relatively complete databases. Conversely, assume that there exists a minimal complete database $D_0 \subseteq D$ for Q relative to $(D_m = \emptyset, V = \emptyset)$. Then for any extensions D'_0 of D_0, $Q(D_0) = Q(D'_0)$ and $(D'_0, D_m) \models V$. We next show that $D \in \text{RCQ}(Q, D_m = \emptyset, V = \emptyset)$. Indeed, for each partially extension D' of D, $Q(D') = Q(D_0) = Q(D)$, since D' and D are both extensions of D_0. Thus, we have that $D \in \text{RCQ}(Q, D_m = \emptyset, V = \emptyset)$.

We next give the Turing reduction. Let TMMinP(Q, D, D_m, V) be an oracle that returns "yes" if D is a minimal database complete for a query Q relative to (D_m, V); otherwise, it returns "no". We give an algorithm Ω for RCP(FO) that calls TMMinP(Q, D, V, D_m) at most $O(2^{|D|})$ times, where D_m and V are both empty. Algorithm Ω works as follows:

1. enumerate all databases $D' \subseteq D$ and do the following;
2. check whether $\text{TMMinP}(Q, D', D_m = \emptyset, V = \emptyset)$ returns "yes"; if so return "yes";
3. return "no" otherwise if no such D' exists.

The correctness of algorithm Ω follows from Lemma 1. Moreover, Ω calls TMMinP(Q, D, V, D_m) at most $O(2^{|D|})$ times. Therefore Ω is a Turing reduction from RCP(FO) to MinP(FO), in the absence of D_m and V. Thus MinP(FO) is undecidable even when D_m and V are absent.

When $\mathcal{L}_Q$ *is* DATALOG. The proof is similar to its counterpart for FO above. First, the lemma below can be easily verified.

Lemma 2. *For any* DATALOG *query* Q, *empty master data* D_m, *a set* V *of FDs, and any database* D *that is partially closed w.r.t.* (D_m, V), $D \in \text{RCQ}(Q, D_m, V)$ *if and only if there exists a database* $D_0 \subseteq D$ *that is a minimal database complete for* Q *relative to* (D_m, V). □

It is known that RCP(DATALOG) is undecidable when D_m is absent and V is a fixed set of FDs (Theorem 1). We construct a Turing reduction from such a special case of RCP(DATALOG) to MinP(DATALOG) along the same lines as the one given above for FO, which show that MinP(DATALOG) is undecidable even when D_m is absent and V is a fixed set of FDs.

This completes the proof of Theorem 2. □

Determining bounded extensions. We next study the bounded extension problem BEP($\mathcal{L}_Q$). Just like RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$), we show that BEP($\mathcal{L}_Q$) is undecidable when $\mathcal{L}_Q$ is FO or DATALOG. Moreover, we show that the problem remains undecidable (a) for FO, when master data D_m and containment constraints V are both absent; and (b) for DATALOG, when V is a fixed set of FDs and master data D_m is empty. Furthermore, all the results hold for any positive integer $K \geq 1$. We remark that BEP($\mathcal{L}_Q$) has not been studied by previous work.

Theorem 3. *The data complexity of* BEP($\mathcal{L}_Q$) *is undecidable when* $\mathcal{L}_Q$ *is* FO *or* DATALOG. *The problem remains undecidable for any positive integer* $K \geq 1$, *and*

- *for* FO, *even when master data and containment constraints are absent*; *and*
- *for* DATALOG, *even when master data is absent and containment constraints are a fixed set of FDs.* □

Proof. We first study the data complexity of BEP($\mathcal{L}_Q$) when $\mathcal{L}_Q$ is FO, and then investigate it when $\mathcal{L}_Q$ is DATALOG.

When $\mathcal{L}_Q$ *is* FO. We show that BEP($\mathcal{L}_Q$) is undecidable even when both master data and containment constraints are absent, by reduction from the embedding problem for the class of all finite semigroups. We refer to the proof of RCP(FO) in Theorem 1 for the statement of the embedding problem. The reduction below is similar to the one given in that proof.

Given a finite partial semigroup $\mathcal{B} = (B, g)$, we define a database D and a fixed query Q in FO, and let the set V of CCs and master data D_m be *empty*. We show that for any positive integer $K \geq 1$, there exists a bounded set of updates ΔD for $(Q, D, D_m = \emptyset, V = \emptyset, K)$ if and only if $\mathcal{B}$ cannot be embedded in a finite semigroup.

(1) Let $\mathcal{R}$ consist of a single relation schema $R_g(A, X, Y, Z)$, where attributes A, X, Y and Z all have a countably infinite domain. The database D of $\mathcal{R}$ consists of a single relation I_g over schema R_g encoding the given finite semigroup $\mathcal{B}$, as described in the proof of Theorem 1. In addition, I_g contains $K+1$ tuples of the form $(2, i, i, i)$ for all $i \in [0, K]$. Furthermore, along the same line as the proof of Theorem 1 for RCP(FO), the extensions of g are encoded by tuples of the form $(1, a', b', c')$. Accordingly, we define that an instance I'_g of R_g is well-formed if (a) each tuple of the form $(0, a, b, c)$ in I'_g has a counterpart of the form $(1, a, b, c)$ in I'_g; (b) $I(1, x, y, z)$ encodes an associative binary function; and (c) each tuple of the form $(2, i, i, i)$ in I'_g has a counterpart of the form $(3, i, i, i)$ in I'_g.

(2) The query Q is a boolean query that encodes the conditions (a), (b) and (c), such that Q returns true on an instance if and only if this instance is well-formed. As in the proof of Theorem 1 for RCP(FO), Q is the conjunction of queries Q_1, Q_2, Q_3, Q_4, and Q_5, where the extra query Q_5 is defined as

$$\forall x\ (R_g(2, x, x, x) \rightarrow R_g(3, x, x, x)),$$

which encodes condition (c). It is easy to see that, for each collection ΔD of tuples, if $|\Delta D| \leq K$, $Q(D \cup \Delta D) = \emptyset$ since $Q_5(D \cup \Delta D) = \emptyset$. Furthermore, for such ΔD, we have that $(D \cup \Delta D, D_m) \models V$ since $V = \emptyset$.

We next show that it is indeed a reduction, *i.e.*, there exists a bounded set of updates ΔD for $(Q, D, D_m = \emptyset, V = \emptyset, K)$ if and only if $\mathcal{B}$ cannot be embedded in a finite semigroup.

($\Rightarrow$) Assume that there exists a bounded set of updates ΔD for $(Q, D, D_m = \emptyset, V = \emptyset, K)$. Then $D \cup \Delta D \in \text{RCQ}(Q, \emptyset, \emptyset)$ and $|\Delta D| \leq K$. Since $Q(D \cup \Delta D)$ is empty, we have that for each partially closed extension D' of $D \cup \Delta D$, $Q(D') = \emptyset$. Along the same line as the proof of Theorem 1 for RCP(FO), one can prove that $\mathcal{B}$ cannot be embedded in a finite semigroup.

($\Leftarrow$) Conversely, if $\mathcal{B}$ cannot be embedded in a finite semigroup, assume by contradiction that there exists no bounded set of updates ΔD for $(Q, D, D_m = \emptyset, V = \emptyset, K)$. This implies that for each set of updates ΔD such that $|\Delta D| \leq K$, we have that $(D \cup \Delta D, D_m = \emptyset) \models V$, $Q(D \cup \Delta D) = \emptyset$, and furthermore, there exists a partially closed extension D' of $D \cup \Delta D$ such that $Q(D')$ is nonempty. Along the line as the proof of Theorem 1 for RCP(FO), we can construct from D' a finite semigroup $\mathcal{A}$ such that $\mathcal{B}$ can be embedded in $\mathcal{A}$, contradicting the assumption that $\mathcal{B}$ cannot be embedded in a finite semigroup.

When $\mathcal{L}_Q$ *is* DATALOG. We next show that BEP(DATALOG) is undecidable by reduction from RCP(DATALOG). The latter has been shown to be undecidable in the proof of Theorem 1, even for a fixed query Q and database D such that $Q(D) = \emptyset$, and when D_m is empty and V is a fixed set of FDs. We consider this special case of RCP(DATALOG). Given such an instance Q, D, D_m and V of RCP(DATALOG), we construct a fixed query Q' in DATALOG, a database D', an empty master database D'_m and a fixed set V' of FDs. We show that for any integer $K \geq 1$, D is in RCQ$(Q, D_m = \emptyset, V)$ if and only if there exists a bounded set of updates for $(Q', D', D_m' = \emptyset, V', K)$.

To simplify the discussion, we assume that D, Q and V are defined over a relation schema R, where R consists of a single relation $R(A_1, \ldots, A_l)$ for a constant l. Indeed, the assumption does not lose the generality, since one can always transform an arbitrary instance of RCP(DATALOG) to an equivalent one defined over a single schema, as shown by the following lemma.

Lemma 3 (*Fan and Geerts [14]*). *For any relational schema* $\mathcal{R} = (R_1, \ldots, R_n)$, *there exist a single relation schema* R, *a linear-time computable bijective function* h_D *from* inst($\mathcal{R}$) *to* inst(R), *a linear-time computable function* $h_Q : \mathcal{L}_Q \rightarrow \mathcal{L}_Q$ *such that for any instance* $\mathcal{I}$ *of* $\mathcal{R}$ *and any query* $Q \in \mathcal{L}_Q$ *over* $\mathcal{R}$, $Q(\mathcal{I}) = h_Q(Q)(h_D(\mathcal{I}))$. *Here* $\mathcal{L}_Q$ *ranges over* CQ, UCQ, FO *and* DATALOG, *and* inst($\mathcal{R}$) *denotes all the instances of schema* $\mathcal{R}$. □

We next give the reduction. By Lemma 3 and Theorem 1, we consider a database $D = (I)$ and a *fixed* DATALOG query Q both defined over schema $R(A_1, \ldots, A_l)$ such that $Q(D)$ is empty, along with empty master data D_m and a set V of FDs, where l can be taken as a constant since Q and V are fixed.

(1) Let $\mathcal{R}'$ consist of two relation schemas $R'(G, A_1, \ldots, A_l)$ and $R_E(C)$, where $R'(G, A_1, \ldots, A_l)$ extends R with a fresh

attribute G that has an infinite domain, and $R_E(C)$ is a unary relation schema consisting of a single attribute C with an infinite domain. We denote by $I(g)$ and $I_E(c)$ the instances of R' and R_E, respectively, where $I(g)$ consists of $\{g\} \times I$, for some constant g in $\mathsf{dom}(G)$, and $I_E(c) = \{(c)\}$ for some constant c in $\mathsf{dom}(C)$. In particular, we consider the database instance D' of $\mathcal{R}'$ consisting of the two relations $I(g_0)$ and $I_E(c_0)$ for some constants g_0 in $\mathsf{dom}(G)$ and c_0 in $\mathsf{dom}(C)$.

(2) The master data D'_m is assumed to be an empty relation.

(3) We define V' such that for each FD $X \to A$ in V, there exists an FD $(G, X) \to A$ in V' defined over R'. It is easy to verify that the following two are equivalent: for any instance I of R defined with constants $g \in \mathsf{dom}(G)$ and $c \in \mathsf{dom}(C)$ as above,

- $(I, D_m = \varnothing) \models V$;
- $((I(g), I_E(c)), D'_m = \varnothing) \models V'$.

In particular, we have that $(D', \varnothing) \models V'$ since $(D, \varnothing) \models V$, for D and D' given above.

(4) To define Q', we first construct a query Q_1 on R' by substituting $R'(z, \vec{y})$ for each occurrence of $R(\vec{y})$ in Q, where z is a common variable shared across all the atoms in Q_1. Obviously, for each instance I of R and any $g \in \mathsf{dom}(G)$, $Q(I)$ is nonempty if and only if $Q_1(I(g))$ is nonempty. We next define

$$Q'(x) \leftarrow Q_1(g, \vec{y}), R_E(x).$$

Intuitively, for any instance I' of R' and instance I_E of R_E, Q' returns the relation I_E if there exists g such that $Q_1(I'_g)$ is nonempty, where I'_g is the subset of I' consisting of tuples t such that $t[G] = g$, and Q' returns empty otherwise. As a consequence, for any instance I of R, any $g \in \mathsf{dom}(G)$, and any nonempty instance I_E of R_E, the following two are equivalent:

- $(I, \varnothing) \models V$ and $Q(I)$ is nonempty;
- $((I(g), I_E), \varnothing) \models V'$ and $Q'(I(g), I_E)$ is nonempty,

In particular, $Q'(D') = \varnothing$ since $Q(D) = \varnothing$.

We next show that this is indeed a reduction, *i.e.*, for any integer $K \geq 1$, D is in $\mathsf{RCQ}(Q, D_m = \varnothing, V)$ if and only if there exists a bounded set of updates for $(Q', D', D'_m = \varnothing, V', K)$.

($\Rightarrow$) Assume that D is in $\mathsf{RCQ}(Q, D_m = \varnothing, V)$. Recall that we assume that $Q(D) = \varnothing$. Then for any partially closed extension D'' of D, we have that $Q(D'') = Q(D) = \varnothing$. Let $\Delta D' = \varnothing$. We show that $\Delta D'$ is a bounded set of updates for $(Q', D', D'_m = \varnothing, V', K)$, *i.e.*, $D' \in \mathsf{RCQ}(Q', D'_m = \varnothing, V')$. Recall that $D' = (I(g_0), I_E(c_0))$. As argued above, $(D', \varnothing) \models V'$ and $Q'(D') = \varnothing$. Since $\Delta D' = \varnothing$, it remains to show that for any partially closed extension (I', I'_E) of D', $Q'(I', I'_E) = \varnothing$. Assume by contradiction that there exists a partially closed extension (I', I'_E) of D' such that $(I', I'_E) \neq D'$ and $Q'(I', I'_E)$ is nonempty. Then by the definition of Q', there exists $g \in \mathsf{dom}(G)$ such that $Q_1(I'_g)$ is nonempty. Thus $Q(\pi_{A_1,\ldots,A_l}(I'_g))$ is nonempty, as discussed above. Obviously, $\pi_{A_1,\ldots,A_l}(I'_g)$ is a partially closed extension of D, which contradicts the assumption that D is in $\mathsf{RCQ}(Q, \varnothing, V)$ since $Q(D) = \varnothing$. Hence $D' \in \mathsf{RCQ}(Q', \varnothing, V')$ and $\Delta D = \varnothing$ is a bounded set of updates for $(Q', D', D'_m = \varnothing, V', K)$, for any integer $K \geq 1$.

($\Leftarrow$) Conversely, assume that D is not in $\mathsf{RCQ}(Q, \varnothing, V)$. Then there exists a partially closed extension $D^e = I^e$ of D such that $D^e \neq D$, $(D^e, D_m) \models V$ and $Q(D^e)$ is nonempty. Assume by contradiction that there exists a bounded set of updates $\Delta D' = (\Delta I', \Delta I_E)$ for $(Q', D', D'_m = \varnothing, V', K)$, where $\Delta I'$ and ΔI_E are instances of R' and R_E, respectively. Then $D' \cup \Delta D'$ is in $\mathsf{RCQ}(Q', D'_m = \varnothing, V')$. Recall that $D' = (I(g_0), I_E(c_0))$. Then $D' \cup \Delta D' = (I(g_0) \cup \Delta I', I_E(c_0) \cup \Delta I_E)$. By the definition of Q', $Q'(D' \cup \Delta D')$ must be empty, since otherwise for any extension I'_E of $I_E(c_0) \cup \Delta I_E$ such that $I'_E \neq I_E(c_0) \cup \Delta I_E$, we have that $(I(g_0) \cup \Delta I', I'_E)$ is a partially closed extension of $D' \cup \Delta D'$, but $Q'(I(g_0) \cup \Delta I', I'_E) = I'_E \neq I_E(c_0) \cup \Delta I_E = Q'(D' \cup \Delta D')$. Now consider the following extension $I'' = I(g_0) \cup \Delta I' \cup I^e(g_1)$ of $I(g_0) \cup \Delta I'$, where g_1 is a fresh constant in $\mathsf{dom}(G)$ but it does not appear in any tuple in $I(g_0) \cup \Delta I'$. Obviously, $D'' = (I'', I_E(c_0) \cup \Delta I_E)$ is a partially closed extension of $(D' \cup \Delta D')$ since $(I(g_0) \cup \Delta I', \varnothing) \models V'$, $(I^e(g_1), \varnothing) \models V'$ and the tuples in $I(g_0) \cup \Delta I'$ differ in their G-attribute with tuples in $I^e(g_1)$. We next show that $Q'(D'')$ is nonempty, and thus $D' \cup \Delta D \notin \mathsf{RCQ}(Q', \varnothing, V')$. Recall that $D^e = I^e$ and $Q(D^e) \neq \varnothing$. Then as argued above, $Q_1(I^e(g_1))$ is nonempty, and hence $Q_1(I''_{g_1})$ is not empty since $I^e(g_1) = I''_{g_1}$. As a result, $Q'(D'')$ is nonempty by the definition of Q', and thus $D' \cup \Delta D \notin \mathsf{RCQ}(Q', \varnothing, V')$. As a consequence, there exist no bounded sets of updates $\Delta D' = (\Delta I', \Delta I_E)$ for $(Q', D', D'_m = \varnothing, V', K)$ for any positive integer $K \geq 1$.

This completes the proof of Theorem 3. □

5. Decidable cases for CQ and UCQ

In this section we study RCP, MinP and BEP, focusing on query languages CQ and UCQ. We show that both RCP and MinP are tractable (Theorems 4 and 5). In addition, we show that BEP is NP-complete when the number K is a variable, while it is tractable when K is a constant (Theorem 6).

5.1. Preliminaries

Before we present the proofs, we first present some notations of [12,14] that will be used in the proofs in this section.

To simplify the discussion, we consider CQ queries that are defined over a single relation. This does not lose generality by Lemma 3, which we have seen in Section 4.

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.*, [2] for details). For each variable x in Q, we use $\mathsf{eq}(x)$ to denote the set of variables y in Q such that $x = y$ is induced from equalities in Q. In T_Q, we represent atomic formula $x = y$ by assigning the same distinct variable to all variables in $\mathsf{eq}(x)$, and $x = c$ by substituting constant 'c' for each occurrence of y in $\mathsf{eq}(x)$. 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. We assume *w.l.o.g.* that distinct tableaus carry distinct variables.

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.

A valuation μ for variables in T_Q is said to be *valid w.r.t.* D if (a) for each variable y in T_Q, $\mu(y)$ is a value from Adom, and (b) $Q(\mu(T_Q))$ is nonempty, *i.e.*, μ observes inequality conditions $x \neq y$ and $x \neq b$ specified in Q.

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

Now consider a UCQ query $Q = Q_1 \cup \cdots \cup Q_n$ where each Q_i is a CQ query. For each $i \in [1, n]$, we represent Q_i as a tableau query (T_i, u_i), where T_i denotes formulas in Q_i and μ_i is the output summary of Q_i. A *valuation* μ for Q in UCQ is $(\mu_1, \ldots, \mu_n)$ such that for each $i \in [1, n]$, μ_i is a valuation for variables in T_i and moreover, for each variable y in T_i, $\mu_i(y) \in$ Adom. The valuation is *valid w.r.t.* D if there exists some $j \in [1, n]$, such that $Q_j(\mu_i(T_j))$ is nonempty, *i.e.*, μ observes inequality conditions $x \neq y$ and $x \neq b$ specified in Q_i.

Consider master data D_m and a set V of CCs. A database D is said to be *bounded by* (D_m, V) for a UCQ query Q if for each valid valuation $\mu = (\mu_1, \ldots, \mu_n)$ for Q, either $(D \cup \Delta, D_m) \not\models V$, or for each $i \in [1, \ell]$, $\mu_i(u_i) \in Q(D)$, where Δ denotes $\mu_1(T_1) \cup \cdots \cup \mu_k(T_n)$.

As shown in [12,14], when Q is in CQ or UCQ, this notion of bounded databases provides us with a sufficient and necessary condition for a database D to be in $\mathsf{RCQ}(Q, D_m, V)$.

Example 4. The following examples illustrate the intuition behind the notion of bounded databases. Recall schema product from Example 1. Let $\text{product}_\varnothing$ be the empty instance of product. Consider a CC ϕ_1: $q(\text{product}) \subseteq \text{product}_\varnothing$, where

$$q_1(b) = \exists a_1, m_1, p_1, s_1, \ldots, a_{k+1}, m_{k+1}, p_{k+1}, s_{k+1} \times \left(\bigwedge_{j \in [1,k+1]} \text{product}(a_j, b, m_j, p_j, s_j) \wedge \bigwedge_{j,l \in [1,k+1]} (a_j \neq a_l) \right).$$

It asserts that each brand has at most k products. Consider query Q_4 that is to find all products with brand = "Kindle". Let D_1 be a database over product and D_m be an empty instance of $\text{product}_\varnothing$, such that $Q_4(D_1)$ returns k distinct tuples. Then one can verify that D_1 is bounded by (D_m, V_1) for Q_4, where V_1 consists of ϕ_1. Indeed, for any valid valuation μ for T_{Q_4}, either (a) $\mu(T_{Q_4})$ contains a new tuple t that is not in D_1 and has t[brand] = "Kindle"; this violates ϕ_1, or (b) $\mu(u_{Q_4}) \in Q_4(D_1)$. It is easy to see that D_1 is complete for Q_4 relative to (D_m, V_1).

As another example, recall from Example 2 the FD ψ: (brand, model → asin, price, sale) on product, which can be expressed as three CCs in CQ, denoted by V_2, using $\text{product}_\varnothing$. Consider the CQ query Q_3 given in Example 3, which is to find all wireless reading devices with brand = "Nook" and model = "PRS-600". Let D_2 be an instance of product such that $Q_3(D_3)$ contains one tuple. Then D_2 is bounded by (D_m, V_2) for Q_3, since for any valid valuation μ' for T_{Q_3}, either $\mu'(T_{Q_3})$ adds a tuple that violates the FD ψ, or the addition of $\mu'(T_{Q_3})$ does not change the answer to Q_3. Again one can see that D_2 is complete for Q_3 relative to (D_m, V_2). □

5.2. Decidability results

We now study the data complexity of $\mathsf{RCP}(\mathcal{L}_Q)$, $\mathsf{MinP}(\mathcal{L}_Q)$ and $\mathsf{BEP}(\mathcal{L}_Q)$ when $\mathcal{L}_Q$ is CQ or UCQ. We show that dropping negation and recursion for $\mathcal{L}_Q$ do make our lives easier: $\mathsf{RCP}(\mathcal{L}_Q)$ and $\mathsf{MinP}(\mathcal{L}_Q)$ are both in PTIME, and $\mathsf{BEP}(\mathcal{L}_Q)$ is NP-complete while it is in PTIME for a fixed K. This is in contrast to the undecidability results shown in the previous section.

Problem $\mathsf{RCP}(\mathcal{L}_Q)$. We start with the relative completeness problem $\mathsf{RCP}(\mathcal{L}_Q)$. We show that its data complexity analysis is tractable when $\mathcal{L}_Q$ is CQ or UCQ. In contrast, as shown in [12,14], the combined complexity of this problem is Π_2^p-complete for the same $\mathcal{L}_Q$.

Theorem 4. *The data complexity of* $\mathsf{RCP}(\mathcal{L}_Q)$ *is in* PTIME *when* $\mathcal{L}_Q$ *is* CQ *or* UCQ. □

Proof. It suffices to show that $\mathsf{RCP}(\text{UCQ})$ is in PTIME. We provide a PTIME algorithm that returns "yes" if the given database D is in $\mathsf{RCQ}(Q, D_m, V)$, and returns "no" otherwise.

The key ingredient of the algorithm is a sufficient and necessary condition for characterizing what databases D are in $\mathsf{RCQ}(Q, D_m, V)$, stated in Lemma 4 below. The lemma is taken from [12,14], where it was verified.

Lemma 4 (*Fan and Geerts [12,14]*). *For any UCQ query Q, any master data D_m, any set V of* CCs *in* UCQ, *and any partially closed database D w.r.t. (D_m, V), D is in* $\mathsf{RCQ}(Q, D_m, V)$ *if and only if D is bounded by (D_m, V) for Q.* □

Capitalizing on the characterization, we next present the PTIME algorithm, denoted by $\mathsf{A}_{\mathsf{RCP}}$. Given a fixed UCQ query $Q = Q_1 \cup \cdots \cup Q_n$, where each Q_i is a CQ query denoted by (T_i, u_i), the tableau query of Q_i, $\mathsf{A}_{\mathsf{RCP}}$ checks whether the given partially closed database D is bounded by (D_m, V) for Q, based on Lemma 4. Note that n is a constant since Q is fixed. More specifically, the algorithm works as follows:

1. for each (T_i, u_i) and each valid valuation μ_i of T_i, do the following:
   (a) let $\Delta_i = \mu_i(T_i)$;
   (b) check whether $(D \cup \Delta_i, D_m) \models V$; if so, continue; otherwise move to the next valid valuation of Q_i;
   (c) check whether $\mu_i(u_i) \notin Q(D)$; if so, return "no"; otherwise move to the next valid valuation of Q_i;
2. return "yes".

Algorithm $\mathsf{A}_{\mathsf{RCP}}$ is correct by Lemma 4: It returns "yes" if and only if the database D is bounded by (D_m, V). We next show that $\mathsf{A}_{\mathsf{RCP}}$ is in PTIME. Since Q is fixed, there are only a constant number of queries Q_i in Q. Thus there are only constantly many T_i's in step 1. For the same reason, there are only polynomially many valid valuations for each query T_i in step 1, since $|\text{Adom}|^{|T_i|}$ is an upper bound on the number of valuations and the size of T_i, denoted by $|T_i|$, is a

constant. Moreover, steps 1(b) and 1(c) are in PTIME since both V and Q are fixed. Thus step 1 is in PTIME. Putting these together, $\mathsf{A}_{\mathsf{RCP}}$ is in PTIME. □

Example 5. We next illustrate how $\mathsf{A}_{\mathsf{RCP}}$ works. Recall from Example 1 the schema product(asin, brand, model, price, sale) and from Example 2 the FD ψ: (brand, model → asin, price, sale) on product which can be expressed as three CCs in CQ, denoted by V_2, and empty master relation D_m. Consider the UCQ query $Q_5 = q \cup q'$, where

$$q(x_a) = \exists x_p, x_s(\text{product}(x_a, \text{Nook}, \text{PRS} - 600, x_p, x_s)),$$

$$q'(x_a) = \exists x_p, x_s(\text{product}(x_a, \text{Kindle}, \text{Paperwhite}, x_p, x_s)),$$

which is to find all wireless reading devices with brand = "Nook" and model = "PRS-600", or brand = "Kindle" and model = "Paperwhite". Let D be as shown in Fig. 1, which consists of two tuples t_1 and t_2 that specify two items. Let master data D_m consist of the empty relation $\text{product}_\emptyset$. Clearly, $Q_5(D) = \{(\text{B002MWYUFU}), (\text{B00AWH595M})\}$.

As shown in Fig. 1, queries q and q' can be represented as tableau queries (T_q, u_q) and $(T_{q'}, u_{q'})$, respectively. To decide whether D is complete for Q_5 relative to $(D_m = \emptyset, V_2)$, $\mathsf{A}_{\mathsf{RCP}}$ checks whether D is bounded by $(D_m = \emptyset, V_2)$ for Q_5. More specifically, $\mathsf{A}_{\mathsf{RCP}}$ carries out steps 1(a)–(c) for every valid valuation of T_q and $T_{q'}$. Assume *w.l.o.g.* that $\mathsf{A}_{\mathsf{RCQ}}$ picks (T_q, u_q) first in step 1. Then Adom = {B002MWYUFU, Nook, PRS-600, \$145, B00AWH595M, Kindle, Paperwhite, \$119, Y, c_a, c_p, c_s, c'_a, c'_p, c'_s}, where c_a, c_p and c_s are new constants in New associated with x_a, x_p and x_s, respectively. Similarly, c'_a, c'_p and c'_s correspond to the variables in $T_{q'}$. (We omit constants denoting variables in V_2 for simplicity.) We assume *w.l.o.g.* that variables x_a, x_p and x_s have an infinite domain that contains Adom. Denote by Γ_q the set of all valid valuation μ_q for variables in T_q, where $\mu_q(x_a), \mu_q(x_p), \mu_q(x_s) \in \text{Adom}$. Let μ_q^0 be the valuation in Γ_q that maps (x_a, x_p, x_s) to (B002MWYUFU, \$145, Y). Obviously, μ_q^0 is the only valuation in Γ_q such that $(D \cup \mu_q^0(T_q), D_m) \models V_2$ and $\mu_q^0(u_q) = (\text{B002MWYUFU}) \in Q_5(D)$, and moreover, for any other valuation μ_q in Γ_q, $(D \cup \mu_q(T_q), D_m) \not\models V_2$.

After this, algorithm $\mathsf{A}_{\mathsf{RCP}}$ moves to $(T_{q'}, u_{q'})$, and gets similar result as above. It returns "yes" and terminates. That is, it concludes that database D is complete for query Q_5 relative to the empty master data D_m and the CCs in V_2. □

Problem MinP($\mathcal{L}_Q$). We show that dropping negation and recursion from queries also makes the minimal completeness problem MinP($\mathcal{L}_Q$) tractable, as opposed to the Δ_3^p-completeness of their combined complexity counterparts [13].

Theorem 5. *The data complexity of* MinP($\mathcal{L}_Q$) *is in* PTIME *when* $\mathcal{L}_Q$ *is* CQ *or* UCQ. □

Proof. We only need to show that MinP(UCQ) is in PTIME. We present a PTIME algorithm to check whether a given database D is a minimal database complete for Q relative to (D_m, V). To do this, we first give a sufficient and necessary condition for characterizing minimal completeness, by the lemma below.

Lemma 5. *For any database D, UCQ query Q, master data D_m, and any set V of* CCs *in* UCQ *such that D is complete for Q relative to (D_m, V), D is not minimal if and only if there exists a tuple $t \in D$ such that $D\backslash\{t\}$ is also complete for Q relative to (D_m, V).* □

We now prove Lemma 5. First assume that there exists a tuple $t \in D$ such that $D\backslash\{t\}$ is in $\text{RCQ}(Q, D_m, V)$. Then obviously, D is not minimal. Conversely, suppose that D is not minimal. Then there exists a subset $D_1 \subsetneq D$ such that D_1 is in $\text{RCQ}(Q, D_m, V)$. Observe that there must exist a subset $D_2 = D\backslash\{t\}$ for some $t \in D$ such that $D_1 \subseteq D_2$ since $D_1 \subsetneq D$, and moreover, $(D_2, D_m) \models V$ since $(D, D_m) \models V$. Indeed, for any containment constraint $\phi \in V$, let ϕ be $q(R) \subseteq p(D_m)$, where q is a UCQ query. We have that $q(D_2) \subseteq q(D) \subseteq p(D_m)$ since $D_2 \subseteq D$ and UCQ queries are monotonic. We next show that $D_2 \in \text{RCQ}(Q, D_m, V)$, *i.e.*, for any partially closed extension D'_2 of D_2, $Q(D'_2) = Q(D_2)$. Indeed, for such D'_2, D'_2 is also a partially closed extension of D_1, andhence, $Q(D'_2) = Q(D_1) = Q(D_2)$ since $D_1 \in \text{RCQ}(Q, D_m, V)$. Thus D_2 is in $\text{RCQ}(Q, D_m, V)$. This concludes the proof of Lemma 5.

Based on Lemma 5, we give a PTIME algorithm, denoted by $\mathsf{A}_{\mathsf{MinP}}$, for determining whether D is a minimal database complete for a query Q *w.r.t.* D_m and V, as follows:

1. check whether D is in $\text{RCQ}(Q, D_m, V)$; if so, continue; otherwise return "no";
2. check whether there exists a tuple $t \in D$ such that $D\backslash\{t\}$ is in $\text{RCQ}(Q, D_m, V)$; if so, return "no"; otherwise return "yes".

Clearly, $\mathsf{A}_{\mathsf{MinP}}$ is correct by Lemma 5. We now prove that $\mathsf{A}_{\mathsf{MinP}}$ is in PTIME. By Theorem 4, it is in PTIME to check whether a database D is in $\text{RCQ}(Q, D_m, V)$ when Q is a fixed UCQ query; so step 1 is in PTIME. Moreover, step 2 is also in PTIME since there are at most $|D|$ tuples $t \in D$ for which we need to check whether $D\backslash\{t\}$ is in $\text{RCQ}(Q, D_m, V)$, which is also in PTIME by Theorem 4. Hence $\mathsf{A}_{\mathsf{MinP}}$ is in PTIME.

This completes the proof of Theorem 5. □

Example 6. Consider $Q_5, D, D_m = \emptyset$ and V_2 described in Example 5, where D is complete for Q_5 relative to

T_q: product

asin	brand	model	price	sale
x_a	Nook	PRS-600	x_p	x_s

$u_q = \langle \text{asin} : x_a \rangle$

$T_{q'}$: product

asin	brand	model	price	sale
x'_a	Kindle	Paperwhite	x'_p	x'_s

$u_{q'} = \langle \text{asin} : x'_a \rangle$,

D:

	asin	brand	model	price	sale
t_1:	B002MWYUFU	Nook	PRS-600	\$145	Y
t_2:	B00AWH595M	Kindle	Paperwhite	\$119	Y

Fig. 1. Tableau queries and the database used in Example 5.

$(D_m = \varnothing, V_2)$. To check whether D is a minimal complete database for Q_5, $\mathsf{A}_{\mathsf{MinP}}$ checks whether there exists a tuple $t \in \{t_1, t_2\}$ such that $D\backslash\{t\} \in \mathsf{RCQ}(Q_5, D_m, V_2)$; if so, the algorithm returns "no"; otherwise it returns "yes".

Assume *w.l.o.g.* that the algorithm first checks whether $D\backslash\{t_1\} = \{t_2\}$ is in $\mathsf{RCQ}(Q_5, D_m, V_2)$, in step 2. Here Adom = {Nook, PRS-600, B00AWH595M, Kindle, Paperwhite, $119, Y, c_a, c_p, c_s, c'_a, c'_p, c'_s}, and $Q_5(D\backslash\{t_1\}) =$ {(B00AWH595M)}. By algorithm $\mathsf{A}_{\mathsf{RCP}}$ given in Theorem 4 (for RCP(UCQ)), there exists a valid valuation μ_q^1 of variables in T_q where $\mu_q^1(x_a) = c_a$, $\mu_q^1(x_p) = c_p$ and $\mu_q^1(x_s) = c_s$, such that $((D\backslash\{t_1\}) \cup \mu_q^1(T_q), D_m = \varnothing) \models V$ and $\mu_q^1(u_q) = (c_a) \notin Q_5(D\backslash\{t_1\})$.That is, $(D\backslash\{t_1\}) \notin \mathsf{RCQ}(Q_5, D_m, V_2)$. Then $\mathsf{A}_{\mathsf{MinP}}$ moves to $D\backslash\{t_2\} = \{t_1\}$. Similarly, algorithm $\mathsf{A}_{\mathsf{RCP}}$ finds a valid valuation μ_q^1 of variables in $T_{q'}$ witnessing that $(D\backslash\{t_2\}) \notin \mathsf{RCQ}(Q_5, D_m, V_2)$. In light of these, algorithm $\mathsf{A}_{\mathsf{MinP}}$ returns "yes". That is, it concludes that D is a minimal database complete for Q_5 relative to (D_m, V_2). □

Problem $\mathsf{BEP}(\mathcal{L}_Q)$. Finally, we study the bounded extension problem $\mathsf{BEP}(\mathcal{L}_Q)$. In contrast to $\mathsf{RCP}(\mathcal{L}_Q)$ and $\mathsf{MinP}(\mathcal{L}_Q)$, $\mathsf{BEP}(\mathcal{L}_Q)$ is intractable when $\mathcal{L}_Q$ is CQ or UCQ. However, it is in PTIME when K is fixed, *i.e.*, when the number of tuples in updates ΔD is bounded by a predefined constant K. As remarked earlier, no previous work has studied this problem.

Theorem 6. *When* $\mathcal{L}_Q$ *is* CQ *or* UCQ, *the data complexity of* $\mathsf{BEP}(\mathcal{L}_Q)$ *is NP-complete*; *it is in* PTIME *for fixed* K. □

Proof. We first study $\mathsf{BEP}(\mathcal{L}_Q)$ when K varies, and then investigate it when K is fixed, for CQ and UCQ.

When K varies. It suffices to show that $\mathsf{BEP}(\mathcal{L}_Q)$ is NP-hard when $\mathcal{L}_Q$ is CQ and it is in NP for UCQ.

Lower bound. We show that $\mathsf{BEP}(\mathrm{CQ})$ is NP-hard by reduction from the 3SAT problem, which is known to be NP-complete (cf. [17]). An instance φ of 3SAT is a formula $C_1 \wedge \cdots \wedge C_r$ in which each clause C_i is a disjunction of three variables or negations thereof taken from $X = \{x_1, \ldots, x_n\}$. Given φ, 3SAT is to decide whether φ is satisfiable, *i.e.*, whether there exists a truth assignment for variables in X that satisfies φ.

Given an instance φ of 3SAT above, we define two fixed relational schemas $\mathcal{R}$ and $\mathcal{R}_m$, a database D of $\mathcal{R}$, master data D_m of $\mathcal{R}_m$, a fixed CQ query Q and a set V of fixed CCs in CQ. We show that there exists a bounded set of updates ΔD for (Q, D_m, V, D, K) if and only if φ is satisfiable, where $K = r - 1$. Here r is the number of clauses in φ.

(1) Let $\mathcal{R}$ consist of two relation schemas $R_C(\mathsf{cid}, X_1, V_1, X_2, V_2, X_3, V_3, V)$ and $R_1(A, B)$. We define the database D as (I_C, I_1), where I_C is an *empty* instance of R_C and $I_1 = \{(1, 0), (0, 0)\}$ is an instance of R_1.

(2) Let $\mathcal{R}_m$ consist of three relation schemas: $R_C^m = R_C$, $R_1^m = R_1$ and $R_2^m = R_1$. We first define an instance I_C^m of R_C^m. Intuitively, I_C^m encodes truth assignments of the clauses in φ. For reasons that will become clear later on, we assign variables (or negations thereof) that appear in a *single clause* with a *fixed* truth value: 1 if it concerns a variable and 0 if it concerns a negated variable. More specifically, let X_p (resp. X_n) denote the set of variables (resp. negated variables) in X that occur in a single clause only. For each clause $C_i = \ell_1^i \vee \ell_2^i \vee \ell_3^i$, for $i \in [1, r]$, we include tuples $(i, x_k, v_k, x_j, v_j, x_m, v_m, v)$ such that (i) $x_k = \ell_1^i$ if $\ell_1^i \in X$ and $x_k = \overline{\ell}_1^i$ if $\overline{\ell}_1^i \in X$; (ii) $v_k = 1$ if $\ell_1^i \in X_p$ and $v_k = 0$ if $\overline{\ell}_1^i \in X_n$; and (iii) x_k can be either 0 or 1 if $\ell_1^i \in X\backslash\{X_p \cup X_n\}$. Similarly for x_j, v_j and x_m and v_m. We set $v = 1$ if the truth assignment encoded in the tuple makes C_i true and set $v = 0$ otherwise. Further, we define the instance I_1^m of R_1^m as $\{(1, 0), (0, 0)\}$, *i.e.*, I_1^m is the same as I_1, and let I_2^m be the *empty* instance of R_2^m. We set $D_m = (I_C^m, I_1^m, I_2^m)$.

(3) The set V consists of the following 15 CCs $\phi_1 - \phi_{12}$:

$$\phi_1: R_C \subseteq R_C^m,$$
$$\phi_2: R_1 \subseteq R_1^m,$$
$$\phi_3 - \phi_5: q_x^p(i, i') \subseteq R_2^m, \quad p \in \{1, 2, 3\},$$
$$\phi_6 - \phi_8: q_v^p(i, i') \subseteq R_2^m, \quad p \in \{1, 2, 3\},$$
$$\phi_9: q_v(i, i') \subseteq R_2^m$$
$$\phi_{10} - \phi_{15}: q^{p,p'}(i, i') \subseteq R_2^m, \quad p, p' \in \{1, 2, 3\},\ p \leqslant p',$$

where the queries q_x^p, q_v^p and $q^{p,p'}$ are defined as follows: For $p \in \{1, 2, 3\}$, $q_x^p(i, i')$ is given by

$$\exists z_1, w_1, z_2, w_2, z_3, w_3, w, z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w' \big(R_C(i, z_1, w_1, z_2, w_2, z_3, w_3, w) \wedge R_C(i', z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w') \wedge (i = i') \wedge (z_p \neq z'_p)\big),$$

$q_v^p(i, i')$ is given by

$$\exists z_1, w_1, z_2, w_2, z_3, w_3, w, z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w' \big(R_C(i, z_1, w_1, z_2, w_2, z_3, w_3, w) \wedge R_C(i', z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w') \wedge (i = i') \wedge (w_p \neq w'_p)\big),$$

$q_v(i, i')$ is given by

$$\exists z_1, w_1, z_2, w_2, z_3, w_3, w, z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w' \big(R_C(i, z_1, w_1, z_2, w_2, z_3, w_3, w) \wedge R_C(i', z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w') \wedge (i = i') \wedge (w \neq w')\big),$$

and for each pair $p, p' \in \{1, 2, 3\}$ where $p \leqslant p'$,

$$q^{p,p'}(i, i') = \exists z_1, w_1, z_2, w_2, z_3, w_3, w, z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w' \big(R_C(i, z_1, w_1, z_2, w_2, z_3, w_3, w) \wedge R_C(i', z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w') \wedge (z_p = z_{p'} \wedge w_p \neq w'_{p'})\big).$$

Note that ϕ_1 is relative to master data I_C^m; ϕ_2 to I_1^m; and ϕ_3–ϕ_{15} to the empty master data instance I_2^m. Intuitively, for any extension $D' = (I'_C, I'_1)$ of D, we have that (a) $(D', D_m) \models \phi_1$ if and only if each tuple in I'_C encodes one clause C_i of φ and a truth assignment μ of variables in C_i, as well as the truth value of C_i under μ; (b) $(D', D_m) \models \phi_2$ if and only if $I'_1 = I_1$, *i.e.*, D' keeps I_1 unchanged; (c) $(D', D_m) \models \{\phi_3, \ldots, \phi_9\}$ if and only if all tuples in I'_C have pairwise distinct cid values, *i.e.*, they corresponds to distinct clauses of φ; and finally, (d) $(D', D_m) \models \{\phi_{10}, \ldots, \phi_{15}\}$ if and only if each pair of tuples in I'_C have the same value for common variables. That is, I'_C encodes a partial truth assignment of X.

(4) We define the query Q as follows:

$$Q(i, i') = \exists z_1, w_1, z_2, w_2, z_3, w_3, w, z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w' \big(R_C(i, z_1, w_1, z_2, w_2, z_3, w_3, w) \wedge R_C(i', z'_1, w'_1, z'_2, w'_2, z'_3, w'_3, w') \wedge R_1(w, w') \wedge i \neq i'\big).$$

Intuitively, for any partially closed extension $D' = (I'_C, I'_1)$ of D, since I'_1 must be $\{(1, 0), (0, 0)\}$ by the definition of ϕ_2, $Q(D')$

returns all pairs (i, i') such that there exist two distinct tuples t and t' in I'_C corresponding to clauses C_i and $C_{i'}$, respectively, *i.e.*, $t[\mathsf{cid}] = i$ and $t[\mathsf{cid}] = i'$, where the truth values of C_i and $C_{i'}$ are not both true under the truth assignments encoded by t and t', respectively. That is, Q returns a nonempty result if not all clauses encoded in I'_C are true.

We now show that φ is satisfiable if and only if there exists a bounded set of updates ΔD for (Q, D_m, V, D, K) for $K = r-1$.

($\Rightarrow$) Assume that φ is satisfiable and let μ_X^0 be a truth assignment that makes φ true. We modify μ_X^0 into a truth assignment μ_X^1 such that μ_X^1 coincides with μ_X^0 on all variables in $X\backslash\{X_p \cup X_n\}$, $\mu_X^1(x) = 1$ if $x \in X_p$ and $\mu_X^1(x) = 0$ if $x \in X_n$. Clearly, μ_X^1 makes φ true as well. Let I_C^r consist of tuples $t_1, \ldots, t_r$ in I_C^m, one for each clause in φ, such that the values of the variables in these tuples agree with μ_X^1. We let I_C^{r-1} consist of the first $r-1$ tuples $t_1, \ldots, t_{r-1}$ and $\Delta D = I_C^{r-1}$. Then $|\Delta D| \le K$ and $D \cup \Delta D = (I_C^{r-1}, I_1)$. It is easy to see that $(D \cup \Delta D, D_m) \models V$ and $Q(D \cup \Delta D) = \varnothing$, by the definitions of V and Q. We next show that ΔD is a bounded set of updates for (Q, D_m, V, D, K), *i.e.*, for any partially closed extension D' of $D \cup \Delta D$, $Q(D') = Q(D \cup \Delta D) = \varnothing$. Observe that (I_C^r, I_1) is the only partially closed extension of $D \cup \Delta D$ such that $(I_C^r, I_1) \neq D \cup \Delta D$, by the definitions of V and the truth assignment μ_X^1. Indeed, only a single tuple, corresponding to clause C_r, can be added in any extension. Furthermore, the truth assignment encoded in this tuple is completely determined: for variables in $X\backslash\{X_p \cup X_n\}$, this tuple must take the value of such variables as encoded by I_C^{r-1}; and for variables in $X_p \cup X_n$ we fixed the variables to 1 (for X_p) and 0 (for X_p), as encoded in I_C^m and the definition of V. Moreover, $Q(I_C^r, I_1) = \varnothing$ by the definition of Q, since all the truth assignments encoded by tuples in I_C^r make the corresponding clauses true. Hence ΔD is a bounded set of updates for (Q, D_m, V, D, K) for $K = r-1$.

($\Leftarrow$) Conversely, assume that φ is not satisfiable. Then there exists no truth assignment μ_X that satisfies φ. Let ΔD be an arbitrary set consisting of no more than K tuples such that $D \cup \Delta D$ is a partially closed extension of D. Then by the definition of V, ΔD consists of only tuples over R_C that encodes distinct clauses of φ, and moreover, for each pair of such tuples t and t', they have the same value for each variable appearing in both of them. We next show that $D \cup \Delta D$ is not in $\mathsf{RCQ}(Q, D_m, V)$. Let μ_X^1 be a truth assignment of X variables that agrees with the partial truth assignment stored in ΔD. Let $D' = (I'_C, I_1)$, where I'_C consists of r tuples, one for each clause in φ, such that the values of the variables in these tuples agree with μ_X^1. Obviously, D' is a partially closed extension of $D \cup \Delta D$, and $D' \neq D \cup \Delta D$. Note that μ_X^1 must make φ false since φ is not satisfiable. That is, the $t[V]$ values of tuples t in I'_C cannot be all 1. By the definition of Q, it can be readily verified that $Q(D \cup \Delta D) \neq Q(D')$. Hence $D \cup \Delta D$ is not in $\mathsf{RCQ}(Q, D_m, V)$.As a result, there exists no bounded set of updates for (Q, D_m, V, D, K) where $K = r-1$.

Upper bound. We show that BEP(UCQ) is in NP by giving an NP algorithm, which returns "yes" if there exists a bounded set of updates ΔD for (Q, D_m, V, K) and returns "no" otherwise.

By Lemma 3, we may assume *w.l.o.g.* that database D is an instance of a single relation schema $R(A_1, \ldots, A_n)$. Let NewV be a set of $K \cdot n$ new constants disjoint from Adom.

The algorithm for BEP(UCQ), denoted by $\mathsf{A}_{\mathsf{BEP}}$, is as follows:

1. guess an instance ΔD of R with no more than K tuples, such that ΔD draws values from Adom ∪ NewV;
2. check whether $D \cup \Delta D$ is in $\mathsf{RCQ}(Q, D_m, V)$; if so, return "yes"; otherwise, reject the guess and go back to step 1.

The algorithm is indeed in NP as it involves guessing K tuples ΔD from a finite set Adom ∪ NewV (step 1) and verifying that $D \cup \Delta D$ is in $\mathsf{RCQ}(Q, D_m, V)$ (which is in PTIME by Theorem 4). We next verify the correctness of the algorithm $\mathsf{A}_{\mathsf{BEP}}$. It suffices to show that there exists a bounded set of updates ΔD for (Q, D_m, V, K) only if there exists a bounded set of updates $\Delta D'$ for (Q, D_m, V, K) which draws values from Adom ∪ NewV.

Given ΔD we construct such a $\Delta D'$ as follows: Let τ be an injective function from the active domain of $D \cup \Delta D$ (*i.e.*, the set of all constants occurring in $D \cup \Delta D$) to Adom ∪ NewV, such that τ when restricted to elements in Adom is the identity mapping. Note that such a function always exists since Adom ∪ NewV contains sufficiently many distinct values. Then, we define $\Delta D' = \{t' = (\tau(a_1), \ldots, \tau(a_n)) | t = (a_1, \ldots, a_n) \in \Delta D\}$. Observe that $|\Delta D'| = |\Delta D|$. We claim that $\Delta D'$ is a bounded set of updates for (Q, D_m, V, K) provided that ΔD is a bounded set of updates.

We first verify that $D \cup \Delta D'$ is partially closed *w.r.t.* $(D_m V)$. Indeed, assume by contradiction that $D \cup \Delta D$ is partially closed but $D \cup \Delta D'$ is not partially closed. This implies that one of the CCs is violated. Assume that $q(D \cup \Delta D') \not\subseteq p(D_m)$ for a UCQ query $q = q_1 \cup \cdots \cup q_k$. Let (T_i, u_i) be the tableau representing q_i, for $i \in [1, k]$. Then there exists a valuation $\mu'_q = (\mu'_1, \cdots, \mu'_k)$ of variables in $T_1, \ldots, T_k$ that draws values from $D \cup \Delta D'$ such that $\mu'_i(u_i) \notin p(D_m)$ for some $i \in [1, k]$. By the definition of $\Delta D'$, one can now verify that there exists a valid valuation μ_i of variables in T_i such that $\mu'_i = \tau \circ \mu_i$ and μ_i draws values from $D \cup \Delta D$,and moreover $\mu_i(u_i) \notin p(D_m)$. Hence, $D \cup \Delta D$ is not partially closed, contradicting the assumption. Thus $D \cup \Delta D'$ is partially closed *w.r.t.* $(D_m V)$.

We next verify that $D \cup \Delta D' \in \mathsf{RCQ}(Q, D_m, V)$. Assume by contradiction that $D \cup \Delta D \in \mathsf{RCQ}(Q, D_m, V)$ but $D \cup \Delta D' \notin \mathsf{RCQ}(Q, D_m, V)$. Let $Q = Q_1 \cup \cdots \cup Q_n$ and denote by (T_i^Q, u_i^Q) the tableau representing Q_i, for each $i \in [1, n]$. By Lemma 4, there must exist a valid valuation $\mu'_Q = (\mu'_1, \ldots, \mu'_n)$ *w.r.t.* $D \cup \Delta D'$ for Q such that $(D \cup \Delta D' \cup \bigcup_{i \in [1,n]} \mu_i'(T_i^Q), D_m) \models V$ and $\mu'_i(u_i^Q) \notin Q(D \cup \Delta D')$. By the definition of $\Delta D'$, one can readily verify that there exists a valid valuation μ_i *w.r.t.* $D \cup \Delta D$ for Q such that $\mu'_i = \tau \circ \mu_i$ and μ_i witnesses that $D \cup \Delta D$ is not bounded by (D_m, V) for Q. This contradicts the assumption above. Thus, $D \cup \Delta D' \in \mathsf{RCQ}(Q, D_m, V)$.

When K is fixed. It suffices to show that BEP(UCQ) is in PTIME for a constant $K \ge 1$. Consider the algorithm given above, in the setting when K is fixed. Clearly, there are polynomially many instances ΔD to guess in step 1 since both Q and V are fixed and K is a constant. So we revise the algorithm such that it returns "no" when all such ΔD are considered and none of them satisfies the condition given in step 2. Otherwise it returns "yes". Denote by $\mathsf{A}^{\mathsf{f}}_{\mathsf{BEP}}$ the revised algorithm above. Obviously, algorithm $\mathsf{A}^{\mathsf{f}}_{\mathsf{BEP}}$ is in PTIME.

Table 1
Data complexity of relative information completeness.

Problems	$\mathcal{L}_Q$	Complexity
RCP, MinP, BEP (Theorems 1–3)	FO ($D_m = V = \emptyset$)	Undecidable
	DATALOG ($D_m = \emptyset$, V is a set of FDs)	
RCP, MinP (Theorems 4 and 5)	CQ, UCQ	PTIME
BEP (Theorem 6)	CQ, UCQ (varied K)	NP-complete
	CQ, UCQ (fixed K)	PTIME

This completes the proof of Theorem 6. □

Example 7. We now illustrate how algorithm $\mathsf{A}^{\mathsf{f}}_{\mathsf{BEP}}$ works. Consider $Q_5, V_2, D_m = \emptyset$ given in Example 5, and an empty database $D_\emptyset$ of schema product. Let $K=2$. Taking these as input, $\mathsf{A}^{\mathsf{f}}_{\mathsf{BEP}}$ checks whether there exists a bounded set ΔD of updates for $(Q_5, D_\emptyset = \emptyset, D_m = \emptyset, V_2, K = 2)$. It enumerates all instances ΔD of product with no more than 2 tuples, by drawing values from Adom ∪ NewV, where Adom={Kindle, Paperwhite, Nook, PRS-600, c_a, c_p, c_s, c'_a, c'_p, c'_s}. and NewV = $\{d_1, d_2, \ldots, d_{10}\}$. For each such instance ΔD, it checks whether $D_\emptyset \cup \Delta D$ is complete for Q_5 relative to (D_m, V_2). For example, consider ΔD_0 consisting of the following two tuples: $t'_1 = \{(c_a,$ "Nook", "PRS−600", $c_p, d_1)\}$ and $t'_2 = \{(d_3,$ "Kindle", "Paperwhite", $d_3, c'_s)\}$. Using the algorithm $\mathsf{A}_{\mathsf{RCP}}$ given in the proof of Theorem 4 for RCP(UCQ), we can see that $D_\emptyset \cup \Delta D_0$ is complete for Q_5 relative to (D_m, V_2). Thus $\mathsf{A}^{\mathsf{f}}_{\mathsf{BEP}}$ returns "yes". That is, there exists a bounded set ΔD_0 of updates for $(Q_5, D_\emptyset, D_m = \emptyset, V_2, K = 2)$. □

6. Conclusions

We have studied the data complexity of three decision problems associated with relative information completeness, namely, RCP($\mathcal{L}_Q$) for deciding whether a database D is complete for a given fixed query Q relative to master data D_m and containment constraints V, MinP($\mathcal{L}_Q$) for determining whether D is a minimal database complete for Q relative to D_m and V, and BEP($\mathcal{L}_Q$) for deciding whether we can complete a database D for answering Q by adding no more than K tuples to D. We have studied these problems when $\mathcal{L}_Q$ ranges over a variety of query languages for expressing queries and containment constraints. We have established the upper and lower bounds of these problems, all matching, for data complexity.

The main complexity results are summarized in Table 1, annotated with their corresponding theorems. Putting these together with the results of [12–14], our main conclusion is that different query languages dominate the complexity, even when *data complexity* is concerned. Indeed, from Table 1 we can see the following. (1) The data complexity analyses of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) are all undecidable when $\mathcal{L}_Q$ is FO or DATALOG. The undecidability is rather robust: when $\mathcal{L}_Q$ is FO, these problems remain undecidable when master data D_m and containment constraints V are both absent. When it comes to DATALOG, these problems are undecidable in the absence of D_m, when containment constraints are fixed FDs. (2) RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) become simpler for query languages without negation and recursion. More specifically, when $\mathcal{L}_Q$ is CQ or UCQ, the data complexity analyses of RCP($\mathcal{L}_Q$) and MinP($\mathcal{L}_Q$) become tractable; BEP($\mathcal{L}_Q$) is NP-complete, but it is in PTIME when K is fixed.

The study of relative information completeness is still in its infancy. A number of issues are targeted for future work. We have focused on incomplete databases from which tuples may be missing. In practice, both tuples and attribute values may be missing. Preliminary results on relative information complexity have been reported in [13], when both tuples and values are missing. Nevertheless, the data complexity analyses of related decision problems have not been studied in that setting.

The data complexity analyses of RCP($\mathcal{L}_Q$), MinP($\mathcal{L}_Q$) and BEP($\mathcal{L}_Q$) are beyond reach in practice when $\mathcal{L}_Q$ is FO or DATALOG. A natural question is to identify special cases of these problems that are decidable and practical. Furthermore, heuristic algorithms are yet to be developed for analyzing these problems, ideally with certain performance guarantees.

Incomplete information is just one of the issues of data quality. Other central data quality issues include data consistency, data accuracy, data currency and entity resolution (see, *e.g.*, [15] for details). To make practical use of the study on data quality, it is necessary to investigate the interaction among these issues. As shown in [12,14], relative information completeness and data consistency can be supported by a uniform framework. Nevertheless, it remains to be studied whether containment constraints can be used to specify currency constraints for data currency [16] and dynamic constraints for entity resolution [11].

Acknowledgements

Cao, Deng and Fan are supported in part by 973 Programs 2014CB340302, China. Cao and Fan are also supported in part by NSFC 61133002, 973 Programs 2012CB316200, China. Fan is also supported in part by Guangdong Innovative Research Team Program 2011D005 and Shenzhen Peacock Program 1105100030834361, China, and EPSRC EP/J015377/1, UK.

References

[1] S. Abiteboul, O.M. Duschka, Complexity of answering queries using materialized views, in: PODS, 1998.

[2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Boston, MA, USA, 1995.
[3] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: PODS, 1999.
[4] A. Cali, D. Lembo, R. Rosati, On the decidability and complexity of query answering over inconsistent and incomplete databases, in: PODS, 2003.
[5] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, View-based query processing: on the relationship between rewriting, answering and losslessness, Theor. Comput. Sci 371 (3) (2007).
[6] J. Chomicki, Consistent query answering: Five easy pieces, in: ICDT, 2007.
[7] A. Deutsch, B. Ludäscher, A. Nash, Rewriting queries using views with access patterns under integrity constraints, Theor. Comput. Sci. 371 (3) (2007).
[8] A. Dreibelbis, E. Hechler, B. Mathews, M. Oberhofer, G. Sauter, Master Data Management Architecture Patterns, IBM, 2007.
[9] C. Elkan, Independence of logic database queries and updates, in: PODS, 1990.
[10] W. Fan, Dependencies revisited for improving data quality, in: PODS, 2008.
[11] W. Fan, H. Gao, X. Jia, J. Li, S. Ma, Dynamic constraints for record matching, VLDB J. 20 (4) (2011) 495–520.
[12] W. Fan, F. Geerts, Relative information completeness, in: PODS, 2009.
[13] W. Fan, F. Geerts, Capturing missing tuples and missing values, in: PODS, 2010.
[14] W. Fan, F. Geerts, Relative information completeness, ACM Trans. Database Syst. 35 (4) (2010).
[15] W. Fan, F. Geerts, Foundations of Data Quality Management, Morgan & Claypool Publishers, USA, 2012.
[16] W. Fan, F. Geerts, J. Wijsen, Determining the currency of data, ACM Trans. Database Syst. 37 (4) (2012) 25.
[17] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York, NY, USA, 1979.
[18] F. Geerts, B. Marnette, Static analysis of schema-mappings ensuring oblivious termination, in: ICDT, 2010.
[19] G. Gottlob, R. Zicari, Closed world databases opened through null values, in: VLDB, 1988.
[20] G. Grahne, The Problem of Incomplete Information in Relational Databases, Springer, Secaucus, NJ, USA, 1991.
[21] T. Imieliński, W. Lipski Jr., Incomplete information in relational databases, J. ACM 31 (4) (1984).
[22] P.G. Kolaitis, J. Panttaja, W.-C. Tan, The complexity of data exchange, in: PODS, 2006, pp. 30–39.
[23] A.Y. Levy, Obtaining complete answers from incomplete databases, in: VLDB, 1996.
[24] A.Y. Levy, Y. Sagiv, Queries independent of updates, in: VLDB, 1993.
[25] C. Li, Computing complete answers to queries in the presence of limited access patterns, VLDB J. 12 (3) (2003).
[26] D. Loshin, Master Data Management, Morgan Kaufmann, San Francisco, CA, USA, 2008.
[27] D.W. Miller, et al., Missing prenatal records at a birth center: a communication problem quantified, in: AMIA Annu Symposium Proceedings, 2005.
[28] J. Minker, D. Perlis, Computing protected circumscription, J. Log. Program 2 (4) (1985).
[29] A. Motro, Integrity=validity + completeness, ACM Trans. Database Syst. 14 (4) (1989).
[30] J. Radcliffe, A. White, Key Issues for Master Data Management, Gartner, USA, 2008.
[31] S. Razniewski, W. Nutt, Completeness of queries over incomplete databases, in: VLDB, 2011.
[32] L. Segoufin, V. Vianu, Views and queries: determinacy and rewriting, in: PODS, 2005.
[33] M. Spielmann, Abstract state machines: verification problems and complexity (Ph.D. thesis), RWTH Aachen, 2000.
[34] R. van der Meyden, Logical approaches to incomplete information: a survey, in: J. Chomicki, G. Saake (Eds.), Logics for Databases and Information Systems, Kluwer, 1998.
[35] M. Vardi, On the integrity of databases with incomplete information, in: PODS, 1986.