Completeness of Queries over Incomplete Databases

Simon Razniewski
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
razniewski@inf.unibz.it

Werner Nutt
Free University of Bozen-Bolzano
Dominikanerplatz 3
39100 Bozen, Italy
nutt@inf.unibz.it

ABSTRACT

Data completeness is an important aspect of data quality as in many scenarios it is crucial to guarantee completeness of query answers. We develop techniques to conclude the completeness of query answers from information about the completeness of parts of a generally incomplete database. In our framework, completeness of a database can be described in two ways: by table completeness (TC) statements, which say that certain parts of a relation are complete, and by query completeness (QC) statements, which say that the set of answers of a query is complete. We identify as core problem to decide whether table completeness entails query completeness (TC-QC). We develop decision procedures and assess the complexity of TC-QC inferences depending on the languages of the TC and QC statements. We show that in important cases weakest preconditions for query completeness can be expressed in terms of table completeness statements, which means that these statements identify precisely the parts of a database that are critical for the completeness of a query. For the related problem of QC-QC entailment, we discuss its connection to query determinacy. Moreover, we show how to use the concrete state of a database to enable further completeness inferences.

1. INTRODUCTION

Incompleteness is a ubiquitous problem in practical data management. Since the very beginning, relational databases have been designed so that they are able to store incomplete data [4]. The theoretical foundations for representing and querying incomplete information were laid by Imielinski and Lipski [15] who captured earlier work on *Codd-, c-* and *v-tables* with their conditional tables and introduced the notion of representation system. Later work on incomplete information has focussed on the concepts of certain and possible answers, which formalize the facts that certainly hold and that possibly hold over incomplete data [1, 12, 17].

Data quality investigates how well data serves its purpose. Aspects of data quality concern accuracy, currency, correctness and similar issues. Especially when many users are supposed to insert data into a database, some tuples may be missing and completeness becomes an essential aspect of data quality.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 37th International Conference on Very Large Data Bases, August 29th - September 3rd 2011, Seattle, Washington.
Proceedings of the VLDB Endowment, Vol. 4, No. 11
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

As an example, consider a problem arising in the management of school data in the province of Bolzano, Italy, which motivated the technical work reported here. The IT department of the provincial school administration runs a database for storing school data, which is maintained in a decentralized manner, as each school is responsible for its own data. Since there are numerous schools in this province, the overall database is notoriously incomplete. However, periodically the statistics department of the province queries the school database to generate statistical reports. These statistics are the basis for administrative decisions such as the opening and closing of classes, the assignment of teachers to schools and others. It is therefore important that these statistics are correct. Therefore, the IT department is interested in finding out which data has to be complete in order to guarantee correctness of the statistics, and on which basis the guarantees can be given.

The problem described above gives rise to several research questions:

1. How can one describe completeness of parts of a possibly incomplete database?
2. How can one characterize the completeness of query answers?
3. How can one infer completeness of query answers from such completeness descriptions?

Reasoning about data completeness has first been investigated by Motro [19] and Halevy [18]. Motro described how knowledge about the completeness of some query answers can allow one to conclude that other query answers are complete as well. Halevy tried to infer whether a query delivers all answers over an incomplete database, given that parts of some database relations are complete. Both papers introduce important concepts, however, they do not set up a framework in which it is possible to give satisfactory answers to the questions above. Later work focussed on answer completeness in the presence of master data [13, 14], resoning about partially complete information in the context of planning [11, 9] or on approximations of possible and certain answers over incomplete databases [8]. In parallel, other researchers developed approaches to quantifying completeness of data and query answers [3, 20].

We proceed as follows. In Section 2, we discuss related work on reasoning about completeness in partially incomplete databases. In Section 3, we formalize partially complete databases and answer Question 1 by formalizing statements to express partial completeness. In Section 4 we deal with Question 2 by discussing characterizations of query completeness. In Sections 5 and 6, we answer Question 3 by discussing inferences of query completeness from table completeness and query completeness, respectively. Section 7 discusses general practical aspects of completeness information. With Section 8, we conclude our work.

2. RELATED WORK

Motro [19] introduced the notion of partially incomplete and incorrect databases as databases that can both miss facts that hold in the real world or contain facts that do not hold there. He described partial completeness in terms of *query completeness* (QC) statements, which express that the answer of a query is complete. He studied how the completeness of a given query can be deduced from the completeness of other queries. His solution was based on rewriting queries using views: to infer that a given query is complete whenever a set of other queries are complete, he would search for a conjunctive rewriting in terms of the complete queries. This solution is correct, but not complete, as later results on query determinacy show: the given query may be complete although no conjunctive rewriting exists [22].

Halevy [18] suggested *local completeness* statements, which we, for a better distinction from the QC statements, call table completeness (TC) statements, as an alternate formalism for expressing partial completeness of an incomplete database. These statements allow one to express completeness of parts of relations independent from the completeness of other parts of the database. The main problem he addressed was how to derive query completeness from table completeness (TC-QC). He reduced TC-QC to the problem of queries independent of updates (QIU) [10]. However, this reduction introduces negation, and thus, except for trivial cases, generates QIU instances for which no decision procedures are known. As a consequence, the decidability of TC-QC remained largely open. Moreover, he demonstrated that by taking into account the concrete database instance and exploiting the key constraints over it, additional queries can be shown to be complete.

Etzioni et al. [11] discussed completeness statements in the context of planning and presented an algorithm for querying partially complete data. Doherty et al. [9] generalized this approach and presented a sound and complete query procedure. Furthermore, they showed that for a particular class of completeness statements, expressed using semi-Horn formulas, querying can be done efficiently in PTIME w.r.t. data complexity.

Demolombe [6, 7] captured Motro's definition of completeness in epistemic logic and showed that in principle this encoding allows for automated inferences about completeness.

Recently, Denecker et al. [8] studied how to compute possible and certain answers over a database instance that is partially complete. They showed that for first-order TC statements and queries, the data complexity of TC-QC entailment wrt. a database instance is in coNP and coNP-hard for some TC statements and queries. Then they focused on approximations for certain and possible answers and proved that under certain conditions their approximations are exact.

Fan and Geerts [13, 14] discussed the problem of query completeness in the presence of master data. In this setting, at least two databases exist: one master database that contains complete information in its tables, and other, possibly incomplete periphery databases that must satisfy certain inclusion constraints wrt. the master data. Then, in the case that one detects that a query over a periphery database contains already all tuples that are maximally possible due to the inclusion constraints, one can conclude that the query is complete. The work is not comparable because in addition to the different setting it always considers a database instance.

Abiteboul et al. [2] discussed representation and querying of incomplete semistructured data. They showed that the problem of deciding query completeness from stored complete query answers, which corresponds to the QC-QC problem raised in [19] for relational data, can be solved in PTIME w.r.t. data complexity.

All results presented in this paper are new.

3. FORMALIZATION

3.1 Standard Definitions

We assume a set of relation symbols Σ, the *signature*. A *database instance* D is a finite set of ground atoms with relation symbols from Σ. For a relation symbol $R \in \Sigma$ we write $R(D)$ to denote the interpretation of R in D, that is, the set of atoms in D with relation symbol R.

A *condition* G is a set of atoms using relations from Σ and possibly the comparison predicates $<$ and $\leq$. As common, we write a condition as a sequence of atoms, separated by commas. A condition is *safe* if each of its variables occurs in a relational atom. A *conjunctive query* is written in the form $Q(\bar{s}) :- B$, where B is a safe condition, $\bar{s}$ is a vector of terms, and every variable in $\bar{s}$ occurs in B. We often refer to the entire query by the symbol Q. As usual, we call $Q(\bar{s})$ the *head*, B the *body*, the variables in $\bar{s}$ the *distinguished variables*, and the remaining variables in B the *nondistinguished variables* of Q. We generically use the symbol L for the subcondition of B containing the relational atoms and M for the subcondition containing the comparisons. If B contains no comparisons, then Q is a *relational conjunctive query*.

The result of evaluating Q over a database instance D is denoted as $Q(D)$. Containment and equivalence of queries are defined as usual. A conjunctive query is *minimal* if no relational atom can be removed from its body without leading to a non-equivalent query.

3.2 Running Example

For our examples throughout the paper, we will use a drastically simplified extract taken from the schema of the Bolzano school database, containing the following four tables:

- *student(name, level, code),*
- *person(name, gender),*
- *language_attendance(name, language),*
- *class(level, code, primary_language).*

The table *student* contains records about students, that is, their names and the level and code of the class they are in. The table *person* contains records about persons (students, teachers, etc.), that is, their names and genders. The table *language_attendance* describes who is attending courses on which language. The table *class* contains classes described by level and code together with the primary language of a class which, since the province is trilingual, can be German, Italian, or Ladin (a minority language spoken in the Alps).

3.3 Completeness

Partial Database. The first and very basic concept is that of a partially complete database or partial database [19]. A database can only be incomplete with respect to another database that is considered to be complete. So we model a partial database as a pair of database instances: one instance that describes the complete state, and another instance that describes the actual, possibly incomplete state. Formally, a *partial database* is a pair $\mathcal{D} = (\hat{D}, \check{D})$ of two database instances $\hat{D}$ and $\check{D}$ such that $\check{D} \subseteq \hat{D}$. In the style of [18], we call $\hat{D}$ (read "D hat") the *ideal* database, and $\check{D}$ (read "D check") the *available* database. The requirement that $\check{D}$ is included in $\hat{D}$ formalizes the intuition that the available database contains no more information than the ideal one.

Example 1. Consider a partial database $\mathcal{D}_S$ for a school with two students, Hans and Maria, and one teacher, Carlo, as follows:

$$\hat{D}_S = \{\textit{student(Hans, 3, A)}, \textit{student(Maria, 5, C)}, \textit{person(Hans, male)}, \textit{person(Maria, female)}, \textit{person(Carlo, male)}\},$$

$$\check{D}_S = \hat{D}_S \setminus \{\textit{person(Carlo, male)}, \textit{student(Maria, 5, C)}\},$$

that is, the available database misses the facts that Maria is a student and that Carlo is a person. □

Next, we define statements to express that parts of the information in $\check{D}$ are complete with regard to the ideal database $\hat{D}$. We distinguish query completeness and table completeness statements.

Query Completeness. For a query Q, the *query completeness* statement $Compl(Q)$ says that Q can be answered completely over the available database. Formally, $Compl(Q)$ is *satisfied* by a partial database $\mathcal{D}$, denoted as $\mathcal{D} \models Compl(Q)$, if $Q(\check{D}) = Q(\hat{D})$.

Example 2. Consider the above defined partial database D_S and the query

$$Q_1(n) :- \textit{student}(n, l, c), \textit{person}(n, \textit{'male'}),$$

asking for all male students. Over both, the available database $\check{D}_S$ and the ideal database $\hat{D}_S$, this query returns exactly *Hans*. Thus, D_S satisfies the query completeness statement for Q_1, that is,

$$\mathcal{D}_S \models Compl(Q_1). \quad \square$$

Table completeness. A table completeness (TC) statement allows one to say that a certain part of a relation is complete, without requiring the completeness of other parts of the database [18]. It has two components, a relation R and a condition G. Intuitively, it says that all tuples of the ideal relation R that satisfy condition G in the ideal database are also present in the available relation R.

Formally, let $R(\bar{s})$ be an R-atom and let G be a condition such that $R(\bar{s}), G$ is safe. We remark that G can contain relational and built-in atoms and that we do not make any safety assumptions about G alone. Then $Compl(R(\bar{s}); G)$ is a *table completeness statement*. It has an *associated query*, which is defined as $Q_{R(\bar{s});G}(\bar{s}) :- R(\bar{s}), G$. The statement is satisfied by $\mathcal{D} = (\hat{D}, \check{D})$, written $\mathcal{D} \models Compl(R(\bar{s}); G)$, if $Q_{R(\bar{s});G}(\hat{D}) \subseteq R(\check{D})$. Note that the ideal instance $\hat{D}$ is used to determine those tuples in the ideal version $R(\hat{D})$ that satisfy G and that the statement is satisfied if these tuples are present in the available version $R(\check{D})$. In the sequel, we will denote a TC statement generically as C and refer to the associated query simply as Q_C.

If we introduce different schemas $\hat{\Sigma}$ and $\check{\Sigma}$ for the ideal and the available database, respectively, we can view the TC statement $C = Compl(R(\bar{s}); G)$ equivalently as the TGD (= tuple-generating dependency) $\delta_C : \hat{R}(\bar{s}), \hat{G} \rightarrow \check{R}(\bar{s})$ from $\hat{\Sigma}$ to $\check{\Sigma}$. It is straightforward to see that a partial database satisfies the TC statement C if and only if it satisfies the TGD δ_C.

Example 3. In the partial database $\mathcal{D}_S$ defined above, we can observe that in the available relation *person*, the teacher *Carlo* is missing, while all students are present. Thus, *person* is complete for all students. The available relation *student* contains *Hans*, who is the only male student. Thus, *student* is complete for all male persons. Formally, these two observations can be written as table completeness statements:

$$C_1 = Compl(\textit{person}(n, g); \textit{student}(n, l, c)),$$
$$C_2 = Compl(\textit{student}(n, l, c); \textit{person}(n, \textit{'male'})),$$

which, as seen, are satisfied by the partial database $\mathcal{D}_S$. □

One can prove that table completeness cannot be expressed by query completeness, because the latter requires completeness of the relevant parts of all the tables that appear in the statement, while the former only talks about the completeness of a single table.

Example 4. As an illustration, consider the table completeness statement C_1 that states that *person* is complete for all students. The corresponding query Q_{C_1} that asks for all persons that are students is

$$Q_{C_1}(n, g) :- \textit{person}(n, g), \textit{student}(n, l, c).$$

Evaluating Q_{C_1} over $\hat{D}_S$ gives the result $\{\textit{Hans}, \textit{Maria}\}$. However, evaluating it over $\check{D}_S$ returns only $\{\textit{Hans}\}$. Thus, $\mathcal{D}_S$ does not satisfy the completeness of the query Q_{C_1} although it satisfies the table completeness statement C_1. □

Reasoning. As usual, a set $\mathcal{S}_1$ of TC- or QC-statements *entails* another set $\mathcal{S}_2$ (we write $\mathcal{S}_1 \models \mathcal{S}_2$) if every partial database that satisfies all elements of $\mathcal{S}_1$ also satisfies all elements of $\mathcal{S}_2$.

While TC statements are a natural way to describe completeness of available data ("These parts of the data are complete"), QC statements capture requirements for data quality ("For these queries we need complete answers"). Thus, checking whether a set of TC statements entails a set of QC statements (TC-QC entailment) is the practically most relevant inference. Checking TC-TC entailment is useful when managing sets of TC statements. Moreover, as we will show later on, TC-QC entailment for aggregate queries with count and sum can be reduced to TC-TC entailment for non-aggregate queries. If completeness guarantees are given in terms of query completeness, also QC-QC entailment is of interest.

4. DESCRIBING QUERY COMPLETENESS BY TABLE COMPLETENESS

In this section we discuss whether and how query completeness can be characterized in terms of table completeness. Suppose we want the answers for a query Q to be complete. An immediate question is which table completeness conditions our database should satisfy so that we can guarantee the completeness of Q.

To answer this question, we introduce canonical completeness statements for a query. Intuitively, the canonical statements require completeness of all parts of relations where tuples can contribute to answers of the query. Consider a query $Q(\bar{s}) :- A_1, \ldots, A_n, M$, with relational atoms A_i and comparisons M. The *canonical completeness statement* for the atom A_i is the TC statement

$$C_i = Compl(A_i; A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n, M).$$

We denote by $\mathcal{C}_Q = \{C_1, \ldots, C_n\}$ the set of all canonical completeness statements for Q.

Example 5. Consider the query

$$Q_2(n) :- \textit{student}(n, l, c), \textit{class}(l, c, \textit{'Ladin'}),$$

asking for the names of all students that are in a class with *Ladin* as primary language. Its canonical completeness statements are the table completeness statements

$$C_1 = Compl(\textit{student}(n, l, c); \textit{class}(l, c, \textit{'Ladin'}))$$
$$C_2 = Compl(\textit{class}(l, c, \textit{'Ladin'}); \textit{student}(n, l, c)). \quad \square$$

As a first result, we find that query completeness can equivalently be expressed by the canonical completeness statements in certain cases.

THEOREM 1. *Let Q be a conjunctive query. Then for all partial database instances $\mathcal{D}$,*

$$\mathcal{D} \models \mathit{Compl}(Q) \quad \textit{iff} \quad \mathcal{D} \models \mathcal{C}_Q,$$

provided one of the following conditions holds: (i) Q is evaluated under multiset semantics, or (ii) Q is a projection-free query.

PROOF. See Appendix A. □

From the theorem we conclude that the canonical completeness statements of a query are sufficient conditions for the completeness of that query, not only under multiset but also under set semantics.

COROLLARY 2. *Let Q be a conjunctive query. Then*

$$\mathcal{C}_Q \models \mathit{Compl}(Q).$$

PROOF. The claim for multiset semantics is shown in Theorem 1. For set semantics, we consider the projection-free variant Q' of Q. Note that $\mathcal{C}_Q = \mathcal{C}_{Q'}$. Thus, by the preceding theorem, if $\mathcal{D} \models \mathcal{C}_Q$, then $\mathcal{D} \models \mathit{Compl}(Q')$, and hence, $Q'(\check{D}) = Q'(\hat{D})$. Since the answers to Q are obtained from the answers to Q' by projection, it follows that $Q(\check{D}) = Q(\hat{D})$ and hence, $\mathcal{D} \models \mathit{Compl}(Q)$. □

Let Q be a conjunctive query. We say that a set $\mathcal{C}$ of TC statements is *characterizing* for Q if for all partial databases $\mathcal{D}$ it holds that $\mathcal{D} \models \mathcal{C}$ if and only if $\mathcal{D} \models \mathit{Compl}(Q)$.

From Corollary 2 we know that the canonical completeness statements are a sufficient condition for query completeness under set semantics. However, on can show that they fail to be a necessary condition for queries with projection. One may wonder whether there exist other sets of characterizing TC statements for such queries. The next theorem tells us that this is not the case.

THEOREM 3. *Let Q be a conjunctive query with at least one non-distinguished variable. Then no set of table completeness statements is characterizing for $\mathit{Compl}(Q)$ under set semantics.*

PROOF. See Appendix B. □

By Theorem 3, for a projection query Q the statement $\mathit{Compl}(Q)$ is not equivalent to any set of TC statements. Thus, if we want to perform arbitrary reasoning tasks, no set of TC statements can replace $\mathit{Compl}(Q)$. However, if we are interested in TC-QC inferences, that is, in finding out whether $\mathit{Compl}(Q)$ *follows* from a set of TC statements $\mathcal{C}$, then, as the next result shows, $\mathcal{C}_Q$ can take over the role of $\mathit{Compl}(Q)$ provided Q is a minimal relational query and the statements in $\mathcal{C}$ are relational.

THEOREM 4. *Let Q be a minimal relational conjunctive query and $\mathcal{C}$ be a set of table completeness statements containing no comparisons. Then*

$$\mathcal{C} \models \mathit{Compl}(Q) \quad \textit{implies} \quad \mathcal{C} \models \mathcal{C}_Q.$$

PROOF. See Appendix C. □

By the previous theorems, we have seen that in several cases satisfaction of the canonical completeness statements is a characterizing condition for query completeness. As a consequence, in these cases the question of whether TC statements imply completeness of a query Q can be reduced to the question of whether these TC statements imply the canonical completeness statements of Q.

This raises the question how to decide TC-TC entailment. Table completeness statements describe parts of relations, which are stated to be complete. Therefore, one set of such statements entails another statement if the part described by the latter is contained in the parts described by the former. Thus, that TC-TC entailment naturally corresponds to query containment.

Example 6. Consider the TC statements C_1 and C_2, stating that the *person* table is complete for all persons and for all female persons, respectively:

$$C_1 = \mathit{Compl}(\textit{person}(n, g);\ \textit{true}),$$
$$C_2 = \mathit{Compl}(\textit{person}(n, g);\ g = \textit{'female'}).$$

It is obvious that C_1 entails C_2. Consider the associated queries Q_{C_1} and Q_{C_2}, describing the parts that are stated to be complete, thus asking for all persons and for all female persons, respectively:

$$Q_{C_1}(n, g) :- \textit{person}(n, g),$$
$$Q_{C_2}(n, g) :- \textit{person}(n, g), g = \textit{'female'}.$$

Clearly, Q_{C_2} is contained in Q_{C_1}. In summary, we can say that C_1 entails C_2 because Q_{C_2} is contained in Q_{C_1}. □

The example can easily be generalized to a linear time reduction under which entailment of a TC statement by other TC statments is translated into containment of a conjunctive query in a union of conjunctive queries. The next theorem shows that there is also a reduction in the opposite direction.

THEOREM 5. *Let $\mathcal{L}$ be a class of conjunctive queries that (i) contains for every relation the identity query, and (ii) is closed under intersection. Then the two problems of TC-TC entailment and containment of unions of queries can be reduced to each other in linear time.*

5. TABLE COMPLETENESS ENTAILING QUERY COMPLETENESS

In this section we discuss the problem of TC-QC entailment and its complexity. First we study general TC-QC entailment, that is, entailment w.r.t. all instances, and then consider entailment w.r.t. a fixed instance of the available database. Finally, we apply our results on general TC-QC entailment to aggregate queries.

5.1 General TC-QC Entailment

First we concentrate on TC-QC entailment and its complexity. We distinguish between four languages of conjunctive queries:

- linear relational queries ($\mathcal{L}_{\mathrm{LRQ}}$): conjunctive queries without repeated relation symbols and without comparisons,
- relational queries ($\mathcal{L}_{\mathrm{RQ}}$): conjunctive queries without comparisons,
- linear conjunctive queries ($\mathcal{L}_{\mathrm{LCQ}}$): conjunctive queries without repeated relation symbols,
- conjunctive queries ($\mathcal{L}_{\mathrm{CQ}}$).

We say that a TC statement is in one of these languages if its associated query is in it. For $\mathcal{L}_1$, $\mathcal{L}_2$ ranging over the above languages, we denote by $\mathsf{TC\text{-}QC}(\mathcal{L}_1, \mathcal{L}_2)$ the problem to decide whether a set of TC statements in $\mathcal{L}_1$ entails completeness of a query in $\mathcal{L}_2$.

As a first result, we show that TC-QC entailment can be reduced to a certain kind of query containment. It also corresponds to a simple containment problem w.r.t. tuple-generating dependencies. From this reduction we obtain upper bounds for the complexity of TC-QC entailment.

To present the reduction, we define the unfolding of a query w.r.t. to a set of TC statements. Let $Q(\bar{s}) :- A_1, \ldots, A_n, N$ be a conjunctive query where N is a set of comparisons and the relational atoms are of the form $A_i = R_i(\bar{s}_i)$, and let $\mathcal{C}$ be a set of TC statements, where each $C_j \in \mathcal{C}$ is of the form $\mathit{Compl}(R_j(\bar{t}_j); G_j)$.

Then the unfolding of Q w.r.t. $\mathcal{C}$, written $Q^{\mathcal{C}}$, is defined as follows:

$$Q^{\mathcal{C}}(\bar{s}) = \bigwedge_{i=1,..,n} \Big(R_i(\bar{s}_i) \wedge \bigvee_{C_j \in \mathcal{C}, R_j = R_i} (G_j \wedge \bar{s}_i = \bar{t}_j) \Big) \wedge N.$$

Intuitively, $Q^{\mathcal{C}}$ is a modified version of Q that uses only those parts of tables that are asserted to be complete by $\mathcal{C}$.

THEOREM 6. *Let $\mathcal{C}$ be a set of TC statements and Q be a conjunctive query. Then*

$$\mathcal{C} \models \mathit{Compl}(Q) \quad \textit{iff} \quad Q \subseteq Q^{\mathcal{C}}.$$

Thus, a query is complete w.r.t. a set of TC statements, iff its results are already returned by the modified version that uses only the complete parts of the database. This will give us upper complexity bounds of TC-QC entailment for several combinations of languages for TC statements and queries. The containment problems arising are more complicated than the ones commonly investigated. The first reason is that queries and TC statements can belong to different classes of queries, thus giving rise to asymmetric containment problems with different languages for container and containee. The second reason is that in general $Q^{\mathcal{C}}$ is not a conjunctive query but a conjunction of unions of conjunctive queries.

To prove Theorem 6, we need a definition and a lemma. Let C be a TC-statement for relation R. Then we define the function f_C that maps database instances to R-facts as $f_C(D) = \{\, R(\bar{t}) \mid \bar{t} \in Q_C(D) \,\}$. That is, if $\hat{D}$ is an ideal database, then $f_C(\hat{D})$ returns those R-facts that must be in $\check{D}$, if $(\hat{D}, \check{D})$ is to satisfy C. We define $f_{\mathcal{C}}(D) = \bigcup_{C \in \mathcal{C}} f_C(D)$ if $\mathcal{C}$ is a set of TC-statements.

LEMMA 7. *Let $\mathcal{C}$ be a set of TC statements. Then*

(i) $f_{\mathcal{C}}(D) \subseteq D$, for all database instances D;

(ii) $(\hat{D}, \check{D}) \models \mathcal{C}$ iff $f_{\mathcal{C}}(\hat{D}) \subseteq \check{D}$, for all $\check{D} \subseteq \hat{D}$;

(iii) $Q^{\mathcal{C}}(D) = Q(f_{\mathcal{C}}(D))$, for all conjunctive queries Q and database instances D.

PROOF. See Appendix D. □

PROOF OF THEOREM 6. "⇒" Suppose $\mathcal{C} \models \mathit{Compl}(Q)$. We want to show that $Q \subseteq Q^{\mathcal{C}}$. Let D be a database instance. Define $\hat{D} = D$ and $\check{D} = f_C(D)$. Then $\mathcal{D} = (\hat{D}, \check{D})$ is a partial database, due to Lemma 7(i), which satisfies $\mathcal{C}$, due to Lemma 7(iii). Exploiting that $\mathcal{D} \models \mathit{Compl}(Q)$, we infer that $Q(D) = Q(\hat{D}) = Q(\check{D}) = Q(f_{\mathcal{C}}(D)) = Q^{\mathcal{C}}(D)$.

"⇐" Suppose $Q \subseteq Q^{\mathcal{C}}$. Let $\mathcal{D} = (\hat{D}, \check{D})$ be a partial database such that $\mathcal{D} \models \mathcal{C}$. Then we have $Q(\hat{D}) \subseteq Q^{\mathcal{C}}(\hat{D}) = Q(f_{\mathcal{C}}(\hat{D})) \subseteq Q(\check{D})$, where the first inclusion holds because of the assumption, the equality holds because of Lemma 7(iii), and the last inclusion holds because of Lemma 7(ii), since $\mathcal{D} \models \mathcal{C}$. □

We show that for linear queries Q the entailment $\mathcal{C} \models \mathit{Compl}(Q)$ can be checked by evaluating the function $f_{\mathcal{C}}$ over test databases derived from Q. If $\mathcal{C}$ does not contain comparisons, one test database is enough, otherwise exponentially many are needed. We use the fact that containment of queries with comparisons can be checked using test databases obtained by instantiating the body of the containee with representative assignments (see [16]). A set of assignments Θ is *representative* for a set of variables X and constants K relative to M, if the $\theta \in \Theta$ correspond to the different ways to linearly order the terms in $X \cup K$ in accordance with M.

LEMMA 8. *Let $Q(\bar{s}) :- L, M$ be a conjunctive query, let $\mathcal{C}$ be a set of TC statements, and let Θ be a set of assignments that is representative for the variables in Q and the constants in L and $\mathcal{C}$ relative to M. Then:*

(i) If $Q \in \mathcal{L}_{\mathrm{LCQ}}$, and $\mathcal{C} \subseteq \mathcal{L}_{\mathrm{RQ}}$, then

$$Q \subseteq Q^{\mathcal{C}} \quad \textit{iff} \quad L = f_C(L).$$

(ii) If $Q \in \mathcal{L}_{\mathrm{LCQ}}$ and $\mathcal{C} \subseteq \mathcal{L}_{\mathrm{CQ}}$, then

$$Q \subseteq Q^{\mathcal{C}} \quad \textit{iff} \quad \theta L = f_{\mathcal{C}}(\theta L) \textit{ for all } \theta \in \Theta.$$

PROOF. See Appendix E. □

THEOREM 9. *We have the following upper bounds:*
(i) $\mathsf{TC\text{-}QC}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is in* PTIME.
(ii) $\mathsf{TC\text{-}QC}(\mathcal{L}_{\mathrm{CQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is in* coNP.
(iii) $\mathsf{TC\text{-}QC}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{RQ}})$ *is in* NP.
(iv) $\mathsf{TC\text{-}QC}(\mathcal{L}_{\mathrm{CQ}}, \mathcal{L}_{\mathrm{CQ}})$ *is in* Π_2^P.

PROOF. *(i)* By Lemma 8(i), the containment test requires to check whether whether $L = f_{\mathcal{C}}(L)$ for a linear relational condition L and a set $\mathcal{C}$ of relational TC statements. Due to the linearity of L, this can be done in polynomial time.

(ii) By Lemma 8(ii), non-containment is in NP, because it suffices to guess an assignment $\theta \in \Theta$ and check that $\theta L \setminus f_C(\theta L) \neq \emptyset$, which can be done in polynomial time, since L is linear.

(iii) Holds because containment of a relational conjunctive query in a positive relational query is in NP(see [21]).

(iv) Holds because containment of a conjunctive query in a positive query with comparisons is in Π_2^P [23]. □

As a preparation for our hardness proofs we show that containment of unions of conjunctive queries can be reduced to TC-QC entailment while preserving classes of queries. For classes of conjunctive queries $\mathcal{L}_1$, $\mathcal{L}_2$ let $\mathsf{Cont}(\mathcal{L}_1, \mathcal{L}_2)$ and $\mathsf{ContU}(\mathcal{L}_1, \mathcal{L}_2)$ denote the problems to decide whether a query in $\mathcal{L}_1$ is contained in a query from $\mathcal{L}_2$, or a union of queries from $\mathcal{L}_2$, respectively.

LEMMA 10. *Let $\mathcal{L}_1$, $\mathcal{L}_2$ be one of the languages $\mathcal{L}_{\mathrm{LRQ}}$, $\mathcal{L}_{\mathrm{LCQ}}$, $\mathcal{L}_{\mathrm{RQ}}$, $\mathcal{L}_{\mathrm{CQ}}$. Then there is a polynomial time many-one reduction from* $\mathsf{ContU}(\mathcal{L}_1, \mathcal{L}_2)$ *to* $\mathsf{TC\text{-}QC}(\mathcal{L}_2, \mathcal{L}_1)$.

PROOF. We show how the reduction works in principle. Consider three queries $Q_i(\bar{t}_i) :- B_i$, where $i = 0, 1, 2$. We define a set of TC statements $\mathcal{C}$ and a query Q such that $\mathcal{C} \models \mathit{Compl}(Q)$ if and only if $Q_0 \subseteq Q_1 \cup Q_2$.

To this end, we introduce a new relation symbol S, with the same arity as the Q_i, and define the new query as $Q(\bar{t}_0) :- S(\bar{t}_0), B_0$. For every relation symbol R in the signature Σ of the Q_i we introduce the statement $C_R = \mathit{Compl}(R(\bar{x}_R); \mathit{true})$, where $\bar{x}_R$ is a vector of distinct variables. Furthermore, for each of Q_i, $i = 1, 2$, we introduce the statement $C_i = \mathit{Compl}(S(\bar{t}_i); B_i)$. Let $\mathcal{C} = \{\, C_1, C_2 \,\} \cup \{\, C_R \mid R \in \Sigma \,\}$. Then it is easy to see that $\mathcal{C}$ and Q do the job. □

To apply this lemma, we need to know the complexity of asymmetric containment problems, which have received little attention so far. To the best of our knowledge, the results in the next lemma have not been shown in the literature before.

LEMMA 11.
(i) $\mathsf{ContU}(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is* coNP*-complete.*
(ii) $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LRQ}})$ *is* NP*-complete.*
(iii) $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is* Π_2^P*-complete.*

PROOF. The upper bounds are straightforward. The lower bounds are proved by a reduction of *(i)* 3-UNSAT, *(ii)* 3-SAT, and *(iii)* ∀∃3-SAT, respectively (see Appendix F). □

The hardness of the TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{CQ}}$) problem is not shown by an examination of the related containment problem. However, using the reduction that proves the hardness of Cont($\mathcal{L}_{\text{RQ}}, \mathcal{L}_{\text{LCQ}}$), we are able to prove the hardness of TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{CQ}}$) directly.

LEMMA 12. *There is a* PTIME *many-one reduction from* $\forall\exists$*3-SAT to* TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{CQ}}$).

PROOF. See Appendix G. □

THEOREM 13. *We have the following lower bounds:*
(i) TC-QC($\mathcal{L}_{\text{LCQ}}, \mathcal{L}_{\text{LRQ}}$) *is* coNP*-hard.*
(ii) TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{RQ}}$) *is* NP*-hard.*
(iii) TC-QC($\mathcal{L}_{\text{LCQ}}, \mathcal{L}_{\text{RQ}}$) *is* Π_2^P*-hard.*
(iv) TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{CQ}}$) *is* Π_2^P*-hard.*

PROOF. Follows from Lemmas 11 and 12. □

The complexity of TC-QC entailment is summarized in Table 1.

5.2 Reasoning w.r.t. Database Instances

So far, we have studied completeness reasoning on the level of statements and queries. In many cases, however, one has access to the current state of the database, which may be exploited for completeness reasoning. Already Halevy [18] observed that taking into account both a database instance and the functional dependencies holding over the ideal database, additional QC statements can be derived. Denecker et al. [8] showed that for first order queries and TC statements, TC-QC entailment with respect to a database instance is in coNP, and coNP-hard for some queries and statements.

Example 7. As a very simple example, consider the query

$$Q(n) :- student(n, l, c), language\_attendance(n, \text{'Greek'}),$$

asking for the names of students attending Greek language courses.

Suppose that the *language_attendance* table is known to be complete. Then this alone does not imply the completeness of Q, because records in the *student* table might be missing.

Now, assume that we additionally find that in our database that the table *language_attendance* contains no record about Greek.

As the *language_attendance* table is known to be complete, no such record can be missing either. There can be no record about Greek at all. If no record about Greek can be in present in the table *language_attendance*, it does not matter which tuples are missing in the *student* table. The result of Q must always be empty, and hence we can conclude that Q is complete in this case. □

Formally, the question of *TC-QC entailment w.r.t. a database instance* is formulated as follows: given an available database instance $\check{D}$, a set of table completeness statements $\mathcal{C}$, and a query Q, is it the case that for all ideal database instances $\hat{D}$ such that $(\hat{D}, \check{D}) \models \mathcal{C}$, we have that $Q(\check{D}) = Q(\hat{D})$? If this holds, we write

$$\check{D}, \mathcal{C} \models Compl(Q).$$

THEOREM 14. *TC-QC entailment w.r.t. a database instance has polynomial data complexity and is* Π_2^P*-complete in combined complexity for all combinations of languages among* $\mathcal{L}_{\text{LRQ}}$*,* $\mathcal{L}_{\text{LCQ}}$*,* $\mathcal{L}_{\text{RQ}}$*, and* $\mathcal{L}_{\text{CQ}}$.

PROOF. For the Π_2^P-hardness in combined complexity, a reduction from $\forall\exists$3-SAT to TC-QC($\mathcal{L}_{\text{LRQ}}, \mathcal{L}_{\text{LRQ}}$) w.r.t. an instance is included in Appendix H.

For tractability, consider the following naive algorithm: Given Q, $\mathcal{C}$ and $\check{D}$, one first evaluates Q over $\check{D}$. Then, one tries to find an ideal database instance $\hat{D}$ such that Q evaluated over $\hat{D}$ returns a tuple that is not returned over $\check{D}$, and $(\hat{D}, \check{D})$ satisfies $\mathcal{C}$. If such an ideal database instance can be found, the completeness of Q is not entailed by $\mathcal{C}$ and $\check{D}$.

There are only finitely many databases $\hat{D}$ to consider, as it suffices to consider those that are the result of adding instantiations of the body of Q to $\check{D}$. For these instantiations, it suffices to only use the constants already present in the database plus one fresh constant for every variable in Q, thus giving polynomial data complexity. For the combined complexity, observe that for showing that the entailment does not hold, it suffices to guess one such database $\hat{D}$ and evaluate Q and $\mathcal{C}$ over $\hat{D}$ to show that $(\hat{D}, \check{D})$ satisfies $\mathcal{C}$ but violates $Compl(Q)$. □

5.3 Aggregate Queries

As we have seen in our school data example, completeness of statistics, which are essentially aggregate queries, is one of the goals of completeness management. In this subsection we draw upon our results for non-aggregate queries to investigate when TC-statements imply completeness of aggregate queries.

We consider queries with the aggregate functions count, sum, and max. Results for max can easily be reformulated for min. Note that count is a nullary function while sum and max are unary. An *aggregate term* is an expression of the form $\alpha(\bar{y})$, where $\bar{y}$ is a tuple of variables, having length 0 or 1. Examples of aggregate terms are count() or sum(y). If $Q(\bar{x}, \bar{y}) :- L, M$ is a conjunctive query, and α an aggregate function, then we denote by Q^α the aggregate query $Q^\alpha(\bar{x}, \alpha(\bar{y})) :- L, M$. We say that Q^α is a *conjunctive aggregate query* and that Q is the *core* of Q^α. Over a database instance, Q^α is evaluated by first computing the answers of its core Q under multiset semantics, then forming groups of answer tuples that agree on their values for $\bar{x}$, and finally applying for each group the aggregate function α to the multiset of y-values of the tuples in that group.

A sufficient condition for an aggregate query to be complete over $\mathcal{D}$ is that its core is complete over $\mathcal{D}$ under multiset semantics. Hence, Corollary 2 gives us immediately a sufficient condition for TC-QC entailment.

PROPOSITION 15. *Let* Q^α *be an aggregate query and* $\mathcal{C}$ *be a set of TC statements. Then* $\mathcal{C} \models \mathcal{C}_Q$ *implies* $\mathcal{C} \models Compl(Q^\alpha)$.

For count-queries, completeness of Q^{count} is the same as completeness of the core Q under multiset semantics. Thus, we can reformulate Theorem 1 for count-queries.

THEOREM 16. *Let* Q^{count} *be a* count*-query and* $\mathcal{C}$ *be a set of TC statements. Then* $\mathcal{C} \models Compl(Q^{\text{count}})$ *if and only if* $\mathcal{C} \models \mathcal{C}_Q$.

In contrast to count-queries, a sum-query can be complete over a partial database $(\hat{D}, \check{D})$ although its core is incomplete. The reason is that it does not hurt if some tuples from $\hat{D}$ that only contribute 0 to the overall sum are missing in $\check{D}$. Nonetheless, we can prove an analogue of Theorem 16 if there are some restrictions on TC statements and query.

We assume that all comparisons range over a dense order, like the rational numbers. We say that a set of comparisons M is *reduced*, if for all terms s, t it holds that $M \models s = t$ only if s and t are syntactially equal. A conjunctive query is *reduced* if its comparisons are reduced. Every satisfiable query can be equivalently rewritten as a reduced query in polynomial time. We say that a sum-query is *nonnegative* if the summation variable y can only be bound to nonnegative values, that is, if $M \models y \geq 0$.

THEOREM 17. *Let* Q^{sum} *be a reduced nonnegative* sum*-query and* $\mathcal{C}$ *be a set of relational TC statements. Then* $\mathcal{C} \models Compl(Q^{\text{sum}})$ *if and only if* $\mathcal{C} \models \mathcal{C}_Q$.

		Query Language			
		LRQ	LCQ	RQ	CQ
TC Statement Language	LRQ	polynomial	polynomial	NP-complete	Π_2^P-complete
	RQ	polynomial	polynomial	NP-complete	Π_2^P-complete
	LCQ	coNP-complete	coNP-complete	Π_2^P-complete	Π_2^P-complete
	CQ	coNP-complete	coNP-complete	Π_2^P-complete	Π_2^P-complete

Table 1: Complexity of deciding TC-QC entailment. Observe the asymmetry of the axes, as the step into NP appears when allowing repeated relation symbols in the query, while the step into coNP appears when having comparisons in the TC statements.

PROOF. See Appendix I. □

In the settings of Theorems 16 and 17, to decide TC-QC entailment, it suffices to decide the corresponding TC-TC entailment problem with the canonical statements of the query core. By Theorem 5, these entailment problems can be reduced in PTIME to containment of unions of conjunctive queries.

We remark without proof that for the query languages considered in this work, TC-TC entailment has the same complexity as TC-QC entailment (cf. Table 1), with the exception of TC-TC($\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{CQ}}$) and TC-TC($\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{CQ}}$). The TC-QC problems for these combinations are Π_2^P-complete, while the corresponding TC-TC problems are in NP.

While for count and sum-queries the multiplicity of answers to the core query is crucial, this has no influence on the result of a max-query. Cohen et al. have characterized equivalence of max-queries in terms of *dominance* of the cores [5]. A query $Q(\bar{s}, y)$ is dominated by query $Q'(\bar{s}', y')$ if for every database instance D and every tuple $(\bar{d}, d) \in Q(D)$ there is a tuple $(\bar{d}, d') \in Q(D)$ such that $d \leq d'$. For max-queries it holds that Q_1^{max} and Q_2^{max} are equivalent if and only if Q_1 dominates Q_2 and vice versa. In analogy to Theorem 6, we can characterize query completeness of max-queries in terms of dominance.

THEOREM 18. *Let $\mathcal{C}$ be a set of TC-statements and Q^{max} be a* max*-query. Then $\mathcal{C} \models Compl(Q^{\mathsf{max}})$ iff Q is dominated by $Q^{\mathcal{C}}$.*

Dominance is a property that bears great similarity to containment. For queries without comparisons it is even equivalent to containment while for queries with comparisons it is characterized by the existence of *dominance mappings*, which ressemble the well-known containment mapppings (see [5]). This allows us to prove that the upper and lower bounds of Theorems 9 and 13 hold also for max-queries. If $\mathcal{L}$ is a class of conjunctive queries, we denote by $\mathcal{L}^{\mathsf{max}}$ the class of max-queries whose core is in $\mathcal{L}$. For languages $\mathcal{L}_1$, $\mathcal{L}_2^{\mathsf{max}}$, the problem TC-QC($\mathcal{L}_1, \mathcal{L}_2^{\mathsf{max}}$) is defined as one would expect. With this notation, we can state the following theorem.

THEOREM 19. *For all languages $\mathcal{L}_1$, $\mathcal{L}_2$ among $\mathcal{L}_{\mathrm{LRQ}}$, $\mathcal{L}_{\mathrm{LCQ}}$, $\mathcal{L}_{\mathrm{RQ}}$ and $\mathcal{L}_{\mathrm{CQ}}$, the complexity of* TC-QC($\mathcal{L}_1, \mathcal{L}_2^{\mathsf{max}}$) *is the same as the one of* TC-QC($\mathcal{L}_1, \mathcal{L}_2$).

6. QUERY COMPLETENESS ENTAILING QUERY COMPLETENESS

To find out whether completeness of a set queries entails completeness of a given query, Motro [19] had the idea of looking for *rewritings* of that query using queries known to be complete. Existence of such a rewriting entails completeness of the query because then the answers of the given query can be computed from the answers of the complete queries.

A problem closely related to the existence of rewritings is the one of *query determinacy*, which had not yet been introduced at the time of Motro's work. Formally, a query Q is *determined* by a set of queries $\mathcal{Q}$, written $\mathcal{Q} \twoheadrightarrow Q$, if for any two database instances D_1 and D_2, we have that $Q'(D_1) = Q'(D_2)$ for all $Q' \in \mathcal{Q}$ implies $Q(D_1) = Q(D_2)$. The decidability of query determinacy for conjunctive queries is an open question so far. But as shown by Segoufin and Vianu [22], for conjunctive queries, the existence of a rewriting and query determinacy coincide. It is clear that query determinacy is a sufficient condition for QC-QC entailment, as expressed by the following proposition:

PROPOSITION 20. *Let $\mathcal{Q} \cup \{ Q \}$ be a set of queries. Then*

$$Compl(\mathcal{Q}) \models Compl(Q) \quad \textit{if} \quad \mathcal{Q} \twoheadrightarrow Q.$$

PROOF. The definitions of query determinacy and QC-QC entailment are exactly the same, except that query determinacy considers arbitrary database instances D_1, D_2, while QC-QC entailment considers only partial databases, that is pairs of instances (D_1, D_2) where $D_1 \supseteq D_2$. □

Whether the existence of a rewriting and thus query determinacy is also a necessary condition for QC-QC entailment is not known so far. We were able, however, to show this for conjunctive queries that are boolean and relational.

THEOREM 21. *Let $\mathcal{Q} \cup \{ Q \}$ be a set of boolean relational conjunctive queries. Then*

$$\mathcal{Q} \twoheadrightarrow Q \quad \textit{if} \quad Compl(\mathcal{Q}) \models Compl(Q).$$

PROOF. Both determinacy and QC-QC entailment hold exactly if there exists a rewriting of Q in terms of $\mathcal{Q}$. The sufficiency of this condition is trivial, for the necessity, observe that if Q cannot be rewritten in terms of $\mathcal{Q}$, then a counterexample of a partial database can be constructed where completeness of the queries in $\mathcal{Q}$ holds but completeness of Q not. This partial database instance then is also a counterexample that Q is not determined by $\mathcal{Q}$. □

Whether determinacy and QC-QC entailment coincide also in the general case, remains an open question.

7. PRACTICAL ISSUES

In this section we briefly discuss practical issues regarding completeness statements. Clearly, any completeness inference is only as correct as the statements it is derived from. It is therefore important to understand on which basis completeness statements can be given and how this can be alleviated.

Except of cases where the ideal database is formalized but hidden, e.g., for authoritative or performance reasons, given completeness statements cannot be verified. They can only be given on basis of information that is outside the available database:

1. Someone may know some part of the ideal world. As an example, a class teacher knows all the student in his class, and can therefore guarantee completeness for all students of his class if they are present in the available database.

2. The method of data collection may be known to be complete. E.g., if every student has to fill in an enrolment form online which is then stored in the database, then this policy assures that by the deadline of enrolment, the table containing the enrolment information must be complete. In contrast to 1., no one could assure this by inspecting the available data.
3. Cardinalities of parts of the ideal world may be known. E.g., if a number of 117 schools in the province is known and the available database contains 117 schools, then under the reasonable assumption that no one enters invalid schools, completeness of the schools can be concluded.

Schema constraints over the ideal database can be useful, e.g., foreign keys can allow to simplify (canonical) completeness statements, or finite domains can allow to replace TC statements by smaller, equivalent ones.

Finally, database instance information can have similar useful aspects, but which to explain is beyond the scope of this paper (for a very simple example, see section 5.2).

8. CONCLUSION

We outlined the importance of data completeness in the field of data quality and illustrated the research questions with the example of the management of school data in the province of Bolzano. We argued that a general approach to database completeness management is necessary.

In this paper, we developed a framework for describing completeness of databases and query answers, drawing upon earlier work by Motro [19] and Halevy [18]. We distinguished between the table completeness (TC) statements introduced by Halevy and the query completeness (QC) statements introduced by Motro.

We identified TC-QC entailment as the central problem. We showed that in certain cases weakest preconditions for TC-QC entailment can be identified, which then allow to reduce TC-QC entailment to TC-TC entailment, which is equivalent to query containment. For TC-QC problems where no characterization of preconditions was possible, we provided a reduction to a particular problem of query containment. We showed decidability of all these problems for conjunctive queries, closing a crucial gap in previous work by Halevy [18], and presented detailed complexity results.

For the problem of QC-QC entailment, we outlined the strong connection to the open problem of deciding conjunctive query determinacy.

In addition, we showed that by taking into account concrete database instances, more completeness statements can be derived. However, TC-QC entailment becomes computationally harder.

We also discussed practical issues regarding gathering of completeness assertions in organisations.

A limitation of previous work, which we have not yet addressed, is that databases are assumed to be null free. Furthermore, weakest preconditions also for queries containing comparisons remain open.

Acknowledgement

We are thankful to Zeno Moriggl and Martin Prosch from the school IT department of the province of Bolzano for introducing us to their problem of query completeness and to Dmitrijs Milajevs for having explored this problem in his BSc thesis. We thank Balder ten Cate and Leonid Libkin for pointing out important connections to our work. This work has been partially supported by the project ACSI, funded by the EU under FP7 grant agreement n. 257593.

9. REFERENCES

[1] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *Proc. SIGMOD*, pages 34–48, 1987.

[2] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and querying XML with incomplete information. *ACM TODS*, 31(1):208–254, 2006.

[3] J. Biswas, F. Naumann, and Q. Qiu. Assessing the completeness of sensor data. In *Proc. DASFAA*, pages 717–732, 2006.

[4] E. F. Codd. Understanding relations (installment #7). *FDT – Bulletin of ACM SIGMOD*, 7(3):23–28, 1975.

[5] S. Cohen, W. Nutt, and Y. Sagiv. Deciding equivalences among conjunctive aggregate queries. *J. ACM*, 54(2), 2007.

[6] R. Demolombe. Answering queries about validity and completeness of data: From modal logic to relational algebra. In *FQAS*, pages 265–276, 1996.

[7] R. Demolombe. Database validity and completeness: Another approach and its formalisation in modal logic. In *KRDB*, pages 11–13, 1999.

[8] M. Denecker, A. Cortés-Calabuig, M. Bruynooghe, and O. Arieli. Towards a logical reconstruction of a theory for locally closed databases. *ACM TODS*, 35(3), 2010.

[9] P. Doherty, W. Lukaszewicz, and A. Szalas. Efficient reasoning using the local closed-world assumption. In *AIMSA*, pages 49–58, 2000.

[10] C. Elkan. Independence of logic database queries and updates. In *Proc. PODS*, pages 154–160, 1990.

[11] O. Etzioni, K. Golden, and D. S. Weld. Sound and efficient closed-world reasoning for planning. *AI*, 89(1-2):113–148, 1997.

[12] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. ICDT*, pages 207–224, 2002.

[13] W. Fan and F. Geerts. Relative information completeness. In *PODS*, pages 97–106, 2009.

[14] W. Fan and F. Geerts. Capturing missing tuples and missing values. In *PODS*, pages 169–178, 2010.

[15] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31:761–791, 1984.

[16] A. C. Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.

[17] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS*, pages 233–246, 2002.

[18] A. Levy. Obtaining complete answers from incomplete databases. In *Proc. VLDB*, pages 402–412, 1996.

[19] A. Motro. Integrity = Validity + Completeness. *ACM TODS*, 14(4):480–502, 1989.

[20] F. Naumann, J.-C. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29:583–615, September 2004.

[21] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operation. In *VLDB*, pages 535–548, 1978.

[22] L. Segoufin and V. Vianu. Views and queries: Determinacy and rewriting. In *Proc. PODS*, pages 49–60, 2005.

[23] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *PODS*, pages 331–345, 1992.

APPENDIX

A. PROOF OF THEOREM 1

THEOREM 1. *Let Q be a conjunctive query. Then for all partial database instances $\mathcal{D}$,*

$$\mathcal{D} \models \mathit{Compl}(Q) \quad \textit{iff} \quad \mathcal{D} \models \mathcal{C}_Q,$$

provided one of the following conditions holds: (i) Q is evaluated under multiset semantics, or (ii) Q is a projection-free query.

PROOF. *(i)* "⇒" Indirect proof: Suppose, one of the completeness assertions in $\mathcal{C}_Q$ does not hold over $\mathcal{D}$, for instance, assertion C_1 for atom A_1. Suppose, R_1 is the relation symbol of A_1. Let C_1 stand for the TC statement $\mathit{Compl}(A_1;\, B_1)$ where $B_1 = B \setminus \{\, A_1 \,\}$ and B is the body of Q. Let Q_1 be the query associated to C_1.

Then $Q_1(\hat{D}) \not\subseteq R_1(\check{D})$. Let t be a tuple that is in $Q_1(\hat{D})$, and therefore in $R_1(\hat{D})$, but not in $R_1(\check{D})$. By the fact that Q_1 has the same body as Q, the valuation υ of Q_1 over $\hat{D}$ that yields t is also a satisfying valuation for Q over $\hat{D}$. So we find one occurence of some tuple $t' \in Q(\hat{D})$, where t' is υ applied to the distinguished variables of Q.

However, υ does not satisfy Q over $\check{D}$ because t is not in $R_1(\check{D})$. By the monotonicity of conjunctive queries, we cannot have another valuation yielding t' over $\check{D}$ but not over $\hat{D}$. Therefore, $Q(\check{D})$ contains at least one occurence of t' less than $Q(\hat{D})$, and hence Q is not complete over D.

(i) "⇐" Direct proof: We have to show that if t is n times in $Q(\hat{D})$ then t is also n times in $Q(\check{D})$.

For every occurence of t in $Q(\hat{D})$ we have a valuation of the variables of Q that is satisfying over $\hat{D}$. We show that if a valuation is satisfying for Q over $\hat{D}$, then it is also satisfying for Q over $\check{D}$. A valuation υ for a conjunctive condition G is satisfying over a database instance if we find all elements of the instantiation νG in that instance. If a valuation satisfies Q over $\hat{D}$, then we will find all instantiated atoms of νG also in $\check{D}$, because the canonical completeness conditions hold in D by assumption. Satisfaction of the canonical completeness conditions requires that for every satisfying valuation of υ of Q, for every atom A in the body of Q, the instantionation atom νA is in $\check{D}$. Therefore, each satisfying valuation for Q over $\hat{D}$ yielding a result tuple $t \in Q(\hat{D})$ is also a satisfying valuation over $\check{D}$ and hence Q is complete over $\mathcal{D}$.

(ii) Follows from *(i)*. When a query with projections is complete under multiset semantics, any variant of it that contains projections is complete as well. □

B. PROOF OF THEOREM 3

THEOREM 3. *Let Q be a conjunctive query with at least one non-distinguished variable. Then no set of table completeness statements is characterizing for $\mathit{Compl}(Q)$ under set semantics.*

PROOF. We show that for queries containing projections, no set of table completeness statements exists that can exactly characterize the query completeness. We present the principle for simple query first, and discuss then how it extends to arbitrary queries with projections.

Consider the relation schema $\Sigma = \{\, R/1 \,\}$ and the boolean query $Q() :- R(x)$. Furthermore, assume a characterizing set of TC statements $\mathcal{C}$ for Q existed. Now consider the partial database instances $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ such that:

$$\hat{D}_1 = \{\, R(a),\, R(b) \,\} \qquad \check{D}_1 = \{\, R(a) \,\}$$
$$\hat{D}_2 = \{\, R(a),\, R(b) \,\} \qquad \check{D}_2 = \{\, R(b) \,\}$$
$$\hat{D}_3 = \{\, R(a),\, R(b) \,\} \qquad \check{D}_3 = \{\ \}.$$

Then, $\mathit{Compl}(Q)$ holds in $\mathcal{D}_1$ and $\mathcal{D}_2$ but not in $\mathcal{D}_3$, and therefore all table completeness statements in $\mathcal{C}$ have to hold in $\mathcal{D}_1$ and $\mathcal{D}_2$, but at least one of them must not hold in $\mathcal{D}_3$. Let us call that condition C.

The statement C must be of the form $\mathit{Compl}(R(x), G)$. Then $G = \textit{true}$ does not hold in $\mathcal{D}_1$ and $\mathcal{D}_2$ (because in both cases there is a tuple in $R(\hat{D}_i)$ that is not in $R(\check{D}_i)$). Other relation symbols to introduce do not exist and repeating R with a variable generates only equivalent conditions. Adding an equality atom for x with some constant generates a table completeness statement that does not hold either in $\mathcal{D}_1$ or $\mathcal{D}_2$. So the only form G can have such that $\mathit{Compl}(R(x), G)$ holds in $\mathcal{D}_1$ and $\mathcal{D}_2$ is $G = \textit{false}$. However, $\mathit{Compl}(R(x), \textit{false})$ holds in $\mathcal{D}_3$ as well.

The proof for this specific query can be extended to any query with a nondistinguished variable x: Following the same idea, one constructs three partial database instances, where the ideal database instances contain the frozen body of the query plus the frozen body where only x has been replaced by another symbol. The three available database instances are once the frozen body, once the frozen body with x changed, and once the empty set. If the completeness statements cannot detect that in the first two instances once the frozen body and once the isomorphic structure is missing, they will not detect that in the third instance both are missing. But over the third instance, the query is clearly incomplete. □

C. PROOF OF THEOREM 4

THEOREM 4. *Let Q be a minimal relational conjunctive query and $\mathcal{C}$ be a set of table completeness statements containing no comparisons. Then*

$$\mathcal{C} \models \mathit{Compl}(Q) \quad \textit{implies} \quad \mathcal{C} \models \mathcal{C}_Q$$

PROOF. By contradiction. Assume Q is minimal and $\mathcal{C}$ is such that $\mathcal{C} \models \mathit{Compl}(Q)$, but $\mathcal{C} \not\models \mathcal{C}_Q$. Then, because $\mathcal{C} \not\models \mathcal{C}_Q$, there exists some partial database $\mathcal{D}$ such that $\mathcal{D} \models \mathcal{C}$, but $\mathcal{D} \not\models \mathcal{C}_Q$. Since $\mathcal{D} \not\models \mathcal{C}_Q$, we find that one of the canonical completeness statements in $\mathcal{C}_Q$ does not hold in $\mathcal{D}$. Let B be the body of Q. Wlog, assume that $\mathcal{D} \not\models C_1$, where C_1 is the canoncial statement for $A_1 = R_1(\bar{t}_1)$, the first atom in B. Let Q_1 be the query associated to C_1. Thus, there exists some tuple $\bar{u}_1$ such that $\bar{u}_1 \in Q_1(\hat{D})$, but $\bar{u}_1 \notin R_1(\check{D})$.

Now we construct a second partial database $\mathcal{D}_0$. To this end let B' be the frozen version of B, that is, each variable in B is replaced by a fresh constant, and let $A'_1 = R_1(\bar{t}'_1)$ be the frozen version of A_1. Now, we define $\mathcal{D}_0 = (B', B' \setminus \{\, A'_1 \,\})$.

Claim: $\mathcal{D}_0$ satisfies $\mathcal{C}$ as well

To prove the claim, we note that the only difference between $\hat{D}_0$ and $\check{D}_0$ is that $A'_1 \notin \check{D}_0$, therefore all TC statements in $\mathcal{C}$ that describe table completeness of relations other than R_1 are satisfied immediately. To show that $\mathcal{D}_0$ satisfies also all statements in $\mathcal{C}$ that describe table completeness of R_1, we assume the contrary and show that this leads to a contradiction.

Assume $\mathcal{D}_0$ does not satisfy some statement $C \in \mathcal{C}$. Then $Q_C(\hat{D}_0) \setminus R_1(\check{D}_0) \neq \emptyset$, where $Q_C(\bar{x}_C)$ is the query associated with C. Since $Q_C(\hat{D}_0) \subseteq R_1(\hat{D}_0)$, it must be the case that $\bar{t}'_1 \in Q_C(\hat{D}_0) \setminus R_1(\check{D}_0)$. Let B_C be the body of Q_C.

Then, $\bar{t}'_1 \in Q_C(\hat{D}_0)$ implies that there is a valuation δ such that $\delta B_C \subseteq B'$ and $\delta \bar{x}_C = \bar{t}'_1$, where $\bar{x}_C$ are the distinguished variables of C. As $\bar{u}_1 \in Q_1(\hat{D})$, and Q_1 has the same body as Q, there exists another valuation θ such that $\theta B \subseteq \hat{D}$ and $\theta \bar{t}_1 = \bar{u}_1$, where $\bar{t}_1$ are the arguments of the atom A_1.

Composing θ and δ, while ignoring the difference between B and its frozen version B', we find that $\theta\delta B_C \subseteq \theta B' = \theta B \subseteq \hat{D}$ and $\theta\delta\bar{x}_C = \theta\bar{t}'_1 = \theta\bar{t}_1 = \bar{u}_1$. In other words, $\theta\delta$ is a satisfying valuation for Q_C over $\hat{D}$ and thus $\bar{u}_1 = \theta\delta\bar{x}_C \in Q_C(\hat{D})$. However, $\bar{u}_1 \notin R_1(\check{D})$, hence, D would not satisfy C. This contradicts our initial assumption. Hence, we conclude that also $\mathcal{D}_0$ satisfies $\mathcal{C}$.

Since $\mathcal{D}_0$ satisfies $\mathcal{C}$ and $\mathcal{C} \models \mathit{Compl}(Q)$, it follows that Q is complete over $\mathcal{D}_0$. As $\hat{D}_0 = B'$, the frozen body of Q, we find that $\bar{x}' \in \hat{Q}(\mathcal{D}_0)$, with $\bar{x}'$ being the frozen version of the distinguished variables $\bar{x}$ of Q. As Q is complete over $\mathcal{D}_0$, we should also have that $\bar{x}' \in Q(\check{D}_0)$. However, as $\hat{D}_0 = B' \setminus \{ A'_1 \}$, this would require a satisfying valuation from B to $B' \setminus \{ A'_1 \}$ that maps $\bar{x}$ to $\bar{x}'$. This valuation would correspond to a non-surjective homomorphism from Q to Q and hence Q would not be minimal. □

D. PROOF OF LEMMA 7

LEMMA 7. *Let $\mathcal{C}$ be a set of TC statements. Then*

1. *$f_{\mathcal{C}}(D) \subseteq D$, for all database instances D;*
2. *$(\hat{D}, \check{D}) \models \mathcal{C}$ iff $f_{\mathcal{C}}(\hat{D}) \subseteq \check{D}$, for all $\check{D} \subseteq \hat{D}$;*
3. *$Q^{\mathcal{C}}(D) = Q(f_{\mathcal{C}}(D))$, for all conjunctive queries Q and database instances D.*

PROOF. *(i)* Holds because of the specific form of the queries associated with $\mathcal{C}$. *(ii)* Follows from the definition of when a partial database satisfies a set of TC statements. *(iii)* Holds because unfolding Q using the queries in $\mathcal{C}$ and evaluating the unfolding over the original database D amounts to the same as computing a new database $f_{\mathcal{C}}(D)$ using the queries in $\mathcal{C}$ and evaluating Q over the result. □

E. PROOF OF LEMMA 8

LEMMA 8. *Let $Q(\bar{s}) :- L, M$ be a conjunctive query, let $\mathcal{C}$ be a set of TC statements, and let Θ be a set of assignments that is representative for the variables in Q and the constants in L and $\mathcal{C}$ relative to M. Then:*

1. *If $Q \in \mathcal{L}_{\mathrm{LCQ}}$, and $\mathcal{C} \subseteq \mathcal{L}_{\mathrm{RQ}}$, then*

$$Q \subseteq Q^{\mathcal{C}} \quad \text{iff} \quad L = f_C(L).$$

2. *If $Q \in \mathcal{L}_{\mathrm{LCQ}}$ and $\mathcal{C} \subseteq \mathcal{L}_{\mathrm{CQ}}$, then*

$$Q \subseteq Q^{\mathcal{C}} \quad \text{iff} \quad \theta L = f_{\mathcal{C}}(\theta L) \text{ for all } \theta \in \Theta.$$

PROOF. *(i)* "⇒" Suppose $f_{\mathcal{C}}(L) \not\subseteq L$. Then there is an atom A such that $A \in L \setminus f_C(L)$. We consider a satisfying assignment θ for Q and create the database $D = \theta L$. Then $Q(D) \neq \emptyset$ and, due to containment, $Q^{\mathcal{C}}(D) \neq \emptyset$. At the same time, $Q^{\mathcal{C}}(D) = Q(f_{\mathcal{C}}(D)) = Q(f_{\mathcal{C}}(\theta L))$. However, since $A \notin f_C(L)$, there is no atom in $f_C(D)$ with the same relation symbol as A and therefore $Q(f_C(D)) = \emptyset$.

"⇐" Let $\bar{c} \in Q(D)$. We show that $\bar{c} \in Q^{\mathcal{C}}(D)$. There exists an assignment θ such that $\theta \models M$, $\theta L \subseteq D$, and $\theta\bar{s} = \bar{c}$. Since $L = f_{\mathcal{C}}(L)$, we conclude that $\theta L = f_{\mathcal{C}}(\theta L) \subseteq f_C(D)$. Hence, θ satisfies Q over $f_{\mathcal{C}}(D)$. Thus $\bar{c} = \theta\bar{s} \in Q(f_{\mathcal{C}}(D)) = Q^{\mathcal{C}}(D)$. □

(ii) Straightforward generalization of the proof for *(i)*.

F. PROOF OF LEMMA 11

LEMMA 11.

1. $\mathsf{ContU}(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is* coNP*-complete.*
2. $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LRQ}})$ *is* NP*-complete.*
3. $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ *is* Π_2^P*-complete.*

The upper bounds are straightforward. For the lower bounds, consider the following reductions.

F.1 $\mathsf{ContU}(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{LCQ}})$ is coNP-hard

3-UNSAT is a coNP-complete problem. A 3-SAT formula is unsatisfiable exactly if its negation is valid.

Let ϕ be a 3-SAT formula in disjunctive normal form as follows:

$$\phi = \gamma_1 \vee \ldots \vee \gamma_k,$$

where each clause γ_i is a conjunction of literals l_{i1}, l_{il2} and l_{i3}, and each literal is a positive or negated propositional variable p_{i1}, p_{i2} or p_{i3}, respectively.

We define queries $Q, Q'_1, \ldots, Q'_k$ as follows:

$$Q() :- C_1(p_{11}, p_{12}, p_{13}), \ldots, C_k(p_{k1}, p_{k2}, p_{k3}),$$

$$Q'_i() :- C_i(x_1, x_2, x_3), x_1 \circ_1 0, x_2 \circ_2 0, x_3 \circ_3 0,$$

where $\circ_j = \text{“}\geq\text{”}$ if l_{ij} is a positive proposition and $\circ_j = \text{“}<\text{”}$ otherwise.

Clearly, Q is a linear relational query and the Q'_i are linear conjunctive queries.

LEMMA 22. *Let ϕ be a 3-SAT formula in disjunctive normal form and Q and Q_1 to Q_k be constructed as above. Then*

$$\phi \text{ is valid} \quad \text{iff} \quad Q \subseteq \bigcup_{i=1..k} Q'_i.$$

PROOF. Observe first that the comparisons in the Q'_i correspond to the disambiguation between positive and negated propositions, that is, whenever a variable is interpreted as a constant greater or equal zero, this corresponds to the truth value assignment *true*, while less zero corresponds to false *false*.

"⇒" If ϕ is valid, then for every possible truth value assignment of the propositional variables p, one of the clauses C_i evaluates to true. Whenever Q returns true over some database instance, the query Q'_i that corresponds to the clause C_i that evaluates to true under that assignment, returns true as well.

"⇐" If the containment holds, then for every instantiation of Q we find a Q'_i that evaluates to true as well. This Q'_i corresponds to the clause C_i of ϕ that evaluates to true under that variable assignment. □

F.2 $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LRQ}})$ is NP-hard

Let ϕ be a 3-SAT formula in conjunctive normal form as follows:

$$\phi = \gamma_1 \wedge \ldots \wedge \gamma_k,$$

where each clause γ_i is a conjunction of literals l_{i1}, l_{i2} and l_{i3}, and each literal is a positive or negated propositional variable p_{i1}, p_{i2} or p_{i3}, respectively.

We define queries Q and Q' as follows:

$$Q() :- F_1^{(7)}, \ldots, F_k^{(7)},$$

where $F_i^{(7)}$ stands for the 7 ground instances of the predicate C_i over $\{ 0, 1 \}$, under which, when 0 is considered as the truth value false and 1 as the truth value true, the clause γ_i evaluates to true, and

$$Q'() :- C_1(p_{11}, p_{12}, p_{13}), \ldots, C_k(p_{k1}, p_{k2}, p_{k3}).$$

Clearly, Q is a relational query and Q' a linear relational query.

LEMMA 23. *Let ϕ be a 3-SAT formula in conjunctive normal form and let Q and Q' be constructed as shown above. Then*

$$\phi \text{ is satisfiable} \quad \textit{iff} \quad Q \subseteq Q'.$$

PROOF. "$\Rightarrow$" If ϕ is satisfiable, there exists an assignment of truth values to the propositions, such that each clause evaluates to true. This assignment can be used to show that whenever Q returns a result, every C_i in Q' can be mapped to one ground instance of that predicate in Q. "$\Leftarrow$" If the containment holds, Q' must be satisfiable over a database instance that contains only the ground facts in Q. The mapping from the variables in Q' to the constant $\{0, 1\}$ gives a satisfying assignment for the truth values of the propositions in ϕ. □

F.3 $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ is Π_2^P-hard

Checking validity of a universally-quantified 3-SAT formula is Π_2^P-complete problem. A universally-quantified 3-SAT formula ϕ is a formula of the form

$$\forall x_1, \ldots, x_m \exists y_1, \ldots, y_n : \gamma_1 \wedge \ldots \wedge \gamma_k,$$

where each γ_i is a disjunction of three literals over propositions p_{i1}, p_{i2} and p_{i3}, and $\{x_1, \ldots, x_m\} \cup \{y_1, \ldots, y_n\}$ are propositions.

Let the C_i be again ternary relations and let R_i and S_i be binary relations. We first define conjunctive conditions G_j and G'_j as follows:

$$G_j = R_j(0, w_j), R_j(w_j, 1), S_j(w_j, 0), S_j(1, 1),$$
$$G'_j = R_j(y_j, z_j), S_j(z_j, x_j), y_j \leq 0, z_j > 0.$$

Now we define queries Q and Q' as follows:

$$Q() :- G_1, \ldots, G_k, F_1^{(7)}, \ldots, F_m^{(7)},$$

where $F_i^{(7)}$ stands for the 7 ground instances of the predicate C_i over $\{0, 1\}$, under which, when 0 is considered as the truth value false and 1 as the truth value true, the clause γ_i evaluates to true, and

$$Q'() :- G'_1, \ldots, G'_m, C_1(p_{11}, p_{12}, p_{13}), \ldots, C_k(p_{k1}, p_{k2}, p_{k3}).$$

Clearly, Q is a relational query and Q' is a linear conjunctive query.

LEMMA 24. *Let ϕ be a universally quantified 3-SAT formula as shown above and let Q and Q' be constructed as above. Then*

$$\phi \text{ is valid} \quad \textit{iff} \quad Q \subseteq Q'.$$

PROOF. Observe first the function of the conditions G and G': Each condition G_j is contained in the condition G'_j, as whenever a structure corresponding to G_j is found in a database instance, G'_j is also found there. However, there is no homomorphism from G'_j to G_j as x_j will either be mapped to 0 or 1, depending on the instantiation of w_j (see also figure 1).

"$\Rightarrow$" If ϕ is valid, then for every possible assignment of truth values to the universally quantified propositions, a satisfying assignment for the existentially quantified ones exists.

Whenever a database instance D satisfies Q, each condition G_j must be satisfied there, and w_j will have a concrete value, that determines which value x_j in G'_j can take. As ϕ is valid, however, it does not matter which values the universally quantified variables x take, there always exists a satisfying assignment for the other variables, such that each atom C_j can be mapped to one of the ground instances $F_j^{(7)}$ that are in D since Q is satisfied over D. Then, Q' will be satisfied over D as well and hence $Q \subseteq Q'$ holds.

"$\Leftarrow$" If Q is contained in Q', for every database D that instantiates Q, we find that Q' is satisfied over it. Especially, no matter whether we instantiate the w_j by a positive or a negative number, and hence whether the x_j will be mapped to 0 or 1, there exists an assignment for the existentially quantified variables such that each C_j is mapped to a ground instance from $F_j^{(7)}$. This directly corresponds to the validity of ϕ, where for every possible assignment of truth values to the universally quantified variables, a satisfying assignment for the existential quantified variables exists. □

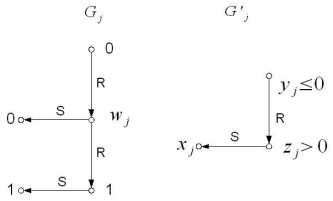

Figure 1: Structure of G_j and G'_j. Depending on the value assigned to w_j, x_j becomes either 0 or 1.

G. PROOF OF LEMMA 12

LEMMA 12. *There is a* PTIME *many-one reduction from $\forall\exists$3-SAT to* TC-QC$(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{CQ}})$.

In Section F.3, we have seen that $Cont(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ is Π_2^P-hard, because validity of $\forall\exists$3-SAT formulas can be translated into a $Cont(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ instance.

We now show Π_2^P-hardness of TC-QC$(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{CQ}})$ by translating those $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ instances into TC-QC$(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{CQ}})$ instances.

Recall that the $\mathsf{Cont}(\mathcal{L}_{\mathrm{RQ}}, \mathcal{L}_{\mathrm{LCQ}})$ problems were of the form "$Q \overset{?}{\subseteq} Q'$?", where Q and Q' were

$$Q() :- G_1, \ldots, G_m, F_1^{(7)}, \ldots, F_k^{(7)},$$

$$Q'() :- G'_1, \ldots, G'_m, C_1(p_{11}, p_{12}, p_{13}), \ldots, C_k(p_{k1}, p_{k2}, p_{k3}),$$

and G_j and G'_j were

$$G_j = R_j(0, w_j), R_j(w_j, 1), S_j(w_j, 0), S_j(1, 1),$$
$$G'_j = R_j(y_j, z_j), S_j(z_j, x_j), y_j \leq 0, z_j > 0.$$

Now consider a set $\mathcal{C}$ of completeness statements containing for every $1 \leq j \leq m$ the statements

$$\mathit{Compl}(R_j(0, \_); \mathit{true}),$$
$$\mathit{Compl}(R_j(\_, 1); \mathit{true}),$$
$$\mathit{Compl}(S_j(\_, 0); \mathit{true}),$$
$$\mathit{Compl}(S_j(\_, 1); \mathit{true}),$$

and containing for every $1 \leq i \leq k$ the statements

$$\mathit{Compl}(C_i(1, \_, \_); \mathit{true}),$$
$$\mathit{Compl}(C_i(0, \_, \_); \mathit{true}),$$

where, for convenience, "_" stands for arbitrary variables.

Clearly, $\mathcal{C}$ contains only statements that are in $\mathcal{L}_{\mathrm{LRQ}}$ and $Q \cap Q'$ is in $\mathcal{L}_{\mathrm{CQ}}$.

LEMMA 25. *Let Q and Q' be queries constructed from the reduction of a $\forall\exists$ 3-SAT instance, and let $\mathcal{C}$ be constructed as above. Then*

$$\mathcal{C} \models \mathit{Compl}(Q \cap Q') \text{ iff } Q \subseteq Q'.$$

PROOF. "$\Leftarrow$" Assume $Q \subseteq Q'$. We have to show that $\mathcal{C} \models \mathit{Compl}(Q \cap Q')$. Because of the containment, $Q \cap Q'$ is equivalent to Q, and hence it suffices to show that $\mathcal{C} \models \mathit{Compl}(Q)$.

Consider a partial database $\mathcal{D}$ such that $\mathcal{D} \models \mathcal{C}$ and $\hat{D} \models Q$. Because of the way in which $\mathcal{C}$ is constructed, all tuples in $\hat{D}$ that made Q satisfied are also in $\check{D}$, and hence $\check{D} \models Q$ as well.

"$\Rightarrow$" Assume $Q \not\subseteq Q'$. We have to show that $\mathcal{C} \not\models \mathit{Compl}(Q \cap Q')$.

Since the containment does not hold, there exists a database D_0 that satisfies Q but not Q'. We construct a partial database $\mathcal{D}$ with

$$\hat{D} = D_0 \cup \sigma B_{Q'}$$
$$\check{D} = D_0,$$

where $\sigma B_{Q'}$ is an instantiation of the body of Q' that uses only the constants -3 and 3.

By that, the tuples from $\sigma B_{Q'}$, missing in $\check{D}$ do not violate $\mathcal{C}$, that always has constants 0 or 1 in the heads of its statements, so $\mathcal{C}$ is satisfied by $\mathcal{D}$. But as $\hat{D}$ satisfies $Q \cap Q'$ and $\check{D}$ does not, this shows that $\mathcal{C} \not\models \mathit{Compl}(Q \cap Q')$. □

H. PROOF OF THEOREM 14

THEOREM 14. *TC-QC entailment w.r.t. a database instance has polynomial data complexity and is Π_2^P-complete in combined complexity for all combinations of languages among $\mathcal{L}_{\mathrm{LRQ}}$, $\mathcal{L}_{\mathrm{LCQ}}$, $\mathcal{L}_{\mathrm{RQ}}$, and $\mathcal{L}_{\mathrm{CQ}}$.*

To show the Π_2^P-hardness of $\mathsf{TC\text{-}QC}(\mathcal{L}_{\mathrm{LRQ}}, \mathcal{L}_{\mathrm{LRQ}})$ entailment w.r.t. a concrete database instance, we give a reduction of the previously seen problem of validity of an universally quantified 3-SAT formula.

So consider ϕ to be an allquantified 3-SAT formula of the form

$$\forall x_1, \ldots, x_m \exists y_1, \ldots, y_n : \gamma_1 \wedge \ldots \wedge \gamma_k.$$

where each γ_i is a disjunction of three literals over propositions p_{i1}, p_{i2} and p_{i3}, and $\{ x_1, \ldots, x_m \} \cup \{ y_1, \ldots, y_n \}$ are propositions.

We define the query completeness problem

$$\Gamma_\phi = (\ \check{D}, \mathcal{C} \stackrel{?}{\models} \mathit{Compl}(Q)\ )$$

as follows. Let the relation schema Σ be $\{ B_1/1, \ldots, B_m/1, R_1/1, \ldots, R_m/1, C_1/3, \ldots, C_k/3 \}$. Let Q be a query defined as

$$Q() :- B_1(x_1), R_1(x_1), \ldots, B_m(x_m), R_m(x_m).$$

Let $\check{D}$ be such that for all B_i, $B_i(\check{D}) = \{ 0, 1 \}$, and for all $i = 1, \ldots, m$ let $R_i(\check{D}) = \{\}$ and let $C_i(\check{D})$ contain all the 7 triples over $\{ 0, 1 \}$ such that γ_i is mapped to true if the variables in γ_i become the truth values *true* for 1 and *false* for 0 assigned.

Let $\mathcal{C}$ be the the set containing the following TC statements

$$\mathit{Compl}(B_1(x), \mathit{true}), \ldots, \mathit{Compl}(B_m(x), \mathit{true})$$
$$\mathit{Compl}(R_1(x_1);\ R_2(x_2), \ldots, R_m(x_m), C_1(p_{11}, p_{12}, p_{13}), \ldots, C_k(p_{k1}, p_{k2}, p_{k3})),$$

where the $\bar{z}_i$ are the variables from γ_i in ϕ.

LEMMA 26. *Let ϕ be a $\forall\exists$3-SAT formula as shown above and let Q, C and $\check{D}$ be constructed as above. Then*

$$\phi \text{ is valid} \quad \text{iff} \quad \check{D}, \mathcal{C} \models \mathit{Compl}(Q).$$

PROOF. Observe first, that validity of ϕ implies that for every possible instantiation of the x variables, there exist an instantiation of the y variables such that C_1 to C_k in the second TC statement in $\mathcal{C}$ evaluate to true.

Completeness of Q follows from $\mathcal{C}$ and $\check{D}$, if Q returns the same result over $\check{D}$ and any ideal database instance $\hat{D}$ that subsumes $\check{D}$ and $\mathcal{C}$ holds over $(\hat{D}, \check{D})$.

Q returns nothing over $\check{D}$. To make Q return the empty tuple over $\hat{D}$, one value from $\{ 0, 1 \}$ has to be inserted into each ideal relation instance $\hat{R}_i$, because every predicate R_i appears in Q, and every extension is empty in $\check{D}$. This step of adding any value from $\{ 0, 1 \}$ to the extensions of the R-predicates in $\hat{D}$ corresponds to the universal quantification of the variables X.

Now observe, that for the query to be complete, none of these combinations of additions may be allowed. That is, every such addition has to violate the table completeness constraint $\mathcal{C}$. As the extension of R_1 is empty in $\check{D}$ as well, $\mathcal{C}$ becomes violated whenever adding the values for the R-predicates leads to the existence of a satisfying valuation of the body of $\mathcal{C}$. For the existence of a satisfying valuation, the mapping of the variables y is not restricted, which corresponds to the existential quantification of the y-variables.

The reduction is correct, because whenever $\mathcal{C}, \check{D} \models \mathit{Compl}(Q)$ holds, for all possible additions of $\{ 0, 1 \}$ values to the extensions of the R-predicates in $\hat{D}$ (all combinations of x), there existed a valuation of the y-variables which yielded a mapping from the C-atoms in $\mathcal{C}$ to the ground atoms of C in $\check{D}$, that satisfied the existential quantified formula in ϕ.

It is complete, because whenever ϕ is valid, then for all valuations of the x-variables, there exists an valuation for the y-variables that satisfies the formula ϕ, and hence for all such extensions of the R-predicates in $\hat{D}$, the same valuation satisfied the body of C_0, thus disallowing the extension. □

I. PROOF OF THEOREM 17

THEOREM 17. *Let Q^{sum} be a reduced nonnegative* sum*-query and $\mathcal{C}$ be a set of relational TC statements. Then*

$$\mathcal{C} \models \mathit{Compl}(Q^{\mathsf{sum}}) \text{ if and only if } \mathcal{C} \models \mathcal{C}_Q.$$

PROOF. The direction $\mathcal{C} \models \mathcal{C}_Q$ implies $\mathcal{C} \models \mathit{Compl}(Q^{\mathsf{sum}})$ holds trivially. It remains to show that $\mathcal{C} \models \mathit{Compl}(Q^{\mathsf{sum}})$ implies $\mathcal{C} \models \mathcal{C}_Q$.

Assume this does not hold. Then $\mathcal{C} \models \mathit{Compl}(Q^{\mathsf{sum}})$ and there exists some $\mathcal{D} = (\hat{D}, \check{D})$ such that $\mathcal{D} \models \mathcal{C}$, but $\mathcal{D} \not\models \mathcal{C}_Q$. W.l.o.g. assume that condition C_1 of $\mathcal{C}_Q$, which corresponds to the first relational atom, say A_1, of the body of Q, is not satisfied by $\mathcal{D}$. Then there is an assignment θ such that $M \models \theta$ and $\theta L \subseteq \check{D}$, but $\theta A_1 \notin \check{D}$. If $\theta y \neq 0$, then we are done, because θ contributes a positive value to the overall sum for the group $\theta\bar{x}$. Otherwise, we can find an assignment θ' such that *(i)* $\theta' \models M$, *(ii)* $\theta' y > 0$, *(iii)* if $\theta' z \neq \theta z$, then $\theta' z$ is a fresh constant not occurring in $\mathcal{D}$, and *(iv)* for all terms s, t, it holds that $\theta' s = \theta' t$ only if $\theta s = \theta t$. Such a θ' exists because M is reduced and the order over which our comparisons range is dense. Due to *(iii)*, in general we do not have that $\theta' L \subseteq \hat{D}$.

We now define a new partial database $\mathcal{D}' = (\hat{D}', \check{D}')$ by adding $\theta' L \setminus \{ \theta' A \}$ both to $\hat{D}$ and $\check{D}$. Thus, we have that *(i)* $\theta' L \subseteq \hat{D}'$, *(ii)* $\theta' L \not\subseteq \check{D}'$, and *(iii)* $\mathcal{D}' \models \mathcal{C}$. The latter claim holds because any violation of $\mathcal{C}$ by $\mathcal{D}'$ could be translated into a violation of $\mathcal{C}$ by $\mathcal{D}$, using the fact that $\mathcal{C}$ is relational. Hence, θ' contributes the positive value $\theta' y$ to the sum for the group $\theta' \bar{x}$ over $\mathcal{D}'$, but not over $\mathcal{D}$. Consequently, the sums for $\theta' \bar{x}$ over $\hat{D}'$ and $\check{D}'$ are different (or there is no such sum over $\check{D}'$), which contradicts our assumption that $\mathcal{C} \models \mathit{Compl}(Q^{\mathsf{sum}})$. □