Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3,
39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009, http://www.inf.unibz.it/krdb/

KRDB Research Centre Technical Report:

Checking Query Completeness over Incomplete Data

Simon Razniewski, Werner Nutt

Affiliation	KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Via della Mostra 4, 39100 Bolzano, Italy
Corresponding author	Simon Razniewski simon.razniewski@stud-inf.unibz.it
Keywords	data quality, metadata management, incomplete information
Number	KRDB11-02
Date	02-03-11
URL	http://www.inf.unibz.it/krdb/

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 that describe the completeness of parts of relations, and query completeness (QC) statements that describe the completeness of query answers. We show for conjunctive queries without comparisons that the problem of table completeness entailing query completeness (TC-QC) can be reduced to the TC-TC entailment problem. We also show that the latter is equivalent to query containment, thus making algorithms developed for this problem available for managing data completeness. For the related problem of QC-QC entailment, we discuss its connection to query determinacy. Furthermore, we show that additional completeness inferences are possible in the presence of finite domain constraints or if the concrete state of a database is taken into account. In both cases, however, completeness reasoning becomes computationally harder.

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 [Cod75]. The theoretical foundations for representing and querying incomplete information were laid by Imielinski and Lipski [IL84] 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, in particular in the context of information integration, has focussed on the concepts of certain and possible answers, which formalize the facts that certainly hold and that possibly hold over incomplete data [AKG87, Len02, FKMP02].

Data quality investigates how well data serves its purpose. Traditionally, aspects of data quality concern accuracy, consistency, correctness and similar issues. With growing amounts of data, also completeness is becoming a more and more important aspect. However so far, most emphasis is put on statistical methods [NFL04, BNQ06].

As an example where data completeness plays a crucial role, we discuss a problem arising in the management of school data in the province of South Tyrol (Italy), which motivated the technical work reported here. The IT department of the provincial school administration runs a distributed database for storing school data. The database contains information about student enrolments, classes, teachers, etc. This data is inserted and maintained in a decentralized manner, as each school is responsible for its own data. Because there are many schools in South Tyrol, the overall database is notoriously incomplete. However, periodically the statistics department of the province queries the school database to generating statistical reports. These statistics are the basis for important 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 how business processes have to be organized such that these guarantees can be provided.

The problem described above gives rise to several research questions:

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

Reasoning about data completeness has been investigated by Motro [Mot89] and Halevy [Lev96]. Motro describes how knowledge about the completeness of some query answers can allow one to conclude that other query answers are complete as well. Halevy tries to answer under which circumstances completeness of parts of database relations allows one to conclude that some query answer is 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.

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 statements that allow to describe partial completeness. In Section 4, we define the completeness inference tasks and develop reasoning techniques for them. In Section 5 we show how the practically important concept of finite domains influences completeness reasoning. In Section 6, we discuss reasoning that takes into account a concrete database instance. In Section 7, we analyse principles for giving completeness statements in practice. Section 8 summarizes our work and outlines open questions.

2 Related Work

Amihai Motro [Mot89] investigated query completeness as an aspect of query integrity, under which he subsumed both completeness and correctness. A partially incorrect and incomplete database is a database that may contain both facts that do not hold in the real world, and miss facts that hold in the real world. He described integrity of parts of the database in terms of query completeness (QC) or query correctness. The inference was such that some query could be derived to be correct or complete, respectively, if it could be rewritten as a conjunctive query using correct or complete queries. This inference is algorithmically correct but incomplete, i.e., the condition of conjunctive rewritability is a sufficient but not a necessary condition for integrity inference.

Alon Halevy introduced a different formalism for stating completeness of parts of a generally incomplete database, namely that of local completeness statements [Lev96], which, for a better distinction from the QC statements, we call table completeness (TC) statements. TC statements allow the assertion of completeness of parts of relations independently of the completeness of other relations. They are distinct from the query completeness statements introduced by Motro, i.e., the two concepts can not generally be reduced to each other.

The main question Halevy's work investigates is how to decide whether table completeness entails query completeness (TC-QC entailment). He proposed a reduction to the problem of Queries Independent of Updates (QIU) [Elk90, LS93]. He observed that a query is complete with respect to a set of TC statements exactly if the query is independent from database updates on parts that are not stated to be complete. That is, the query is complete if its result only depends on parts stated to be complete.

A second contribution of Halevy's work was the idea of deciding query completeness with respect to a concrete database instance. Using an interplay between information from a concrete database instance and functional dependencies, he showed how additional completeness statements may be derived in the presence of a concrete database instance.

The main shortcoming of Halevy's work is that his reduction of TC-QC problems leads to QIU problems for which no general decision procedures are known. The reduction introduces negation symbols into the table completeness statements which make solving the QIU problem difficult. For QIU, reductions to query containment exist. But containment of queries containing negation is decidable only in restricted cases. So with the reduction of TC-QC to QIU, only for some trivial problems in TC-QC the corresponding QIU problems are known to be decidable. Those trivial TC-QC problems are problems where either the query contains no projections, or the table completeness statements contain only selfjoins. For the other TC-QC problems, decidability remained unknown.

Etzioni et al. [EGW97] discussed completeness statements in the context of planning and presented an algorithm for querying partially complete data. Doherty et al. [DLS00] 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.

Recently, Denecker et al.[DCCBA10] 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 [FG09, FG10] 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. The inclusion dependencies set upper bounds for the possible query answers over the periphery databases. So, if one detects that a query over a periphery database contains already all tuples that are maximally possible due to the inclusion dependencies, one concludes that the query is complete.

Abiteboul et al. [ASV06] 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 [Mot89] for relational data, can be solved in PTIME w.r.t. data complexity.

3 Formalization

3.1 Standard Definitions

We assume a fixed set of relation symbols Σ, and an infinite set of constants *dom*. A *database instance* I is a finite set of ground atoms with relation symbols from Σ and arguments from *dom*. For a relation symbol $R \in \Sigma$ we write $R(I)$ to denote the interpretation of R in I, that is, the set of atoms in I with relation symbol R.

A *condition* 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 $Q(\bar{x}) :- B(\bar{x}, \bar{y})$, where B is a safe condition. We often refer to the entire query by the symbol Q. As usual, we call $Q(\bar{x})$ the *head*, $B(\bar{x}, \bar{y})$ the *body*, the variables in the vector $\bar{x}$ the *distinguished variables*, and the variables in the vector $\bar{y}$ the *nondistinguished variables* of Q. If B contains no comparisons, then Q is a *relational conjunctive query*.

The result of evaluating a query Q over a database instance I is denoted as $Q(I)$. 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 South Tyrol 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 class level and code they are in. The table *person* contains records about persons (e.g., students, teachers, etc.), that is, their names and genders. The table *language_attendance* describes who is attending courses in which language. The table *class* contains classes described by level and code together with the primary language of that class (which in South Tyrol can be German, Italian or Ladin).

3.3 Completeness

The first and very basic concept is that of a partially complete database, which we also call a partial database. 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 instance* is a pair $D = (\check{D}, \hat{D})$ of two database instances $\check{D}$ and $\hat{D}$ such that $\check{D} \subseteq \hat{D}$. We call $\check{D}$ (read "D check") the *available* database instance, and $\hat{D}$ (read "D hat") the *ideal* database instance. The requirement that $\check{D}$ is included in $\hat{D}$ formalizes the intuition that the real database contains no more information than the ideal one.

Example 1. Consider a partial database instance D_S for a school with two students, John and Mary, and one teacher, Bob. The ideal database instance is

$$\begin{aligned}\hat{D}_S = \{\, & \mathit{student}(\mathit{John}, 3, A), \mathit{student}(\mathit{Mary}, 5, C),\\ & \mathit{person}(\mathit{John}, \mathit{male}), \mathit{person}(\mathit{Mary}, \mathit{female}),\\ & \mathit{person}(\mathit{Bob}, \mathit{male})\,\}.\end{aligned}$$

Suppose that in the available database the information is missing that Mary is a student and that Bob is a person. The available database, with missing information crossed out, is then

$$\begin{aligned}\check{D}_S = \{\, & \mathit{student}(\mathit{John}, 3, A), \cancel{\mathit{student}(\mathit{Mary}, 5, C)},\\ & \mathit{person}(\mathit{John}, \mathit{male}), \mathit{person}(\mathit{Mary}, \mathit{female}),\\ & \cancel{\mathit{person}(\mathit{Bob}, \mathit{male})}\,\}. \quad \square\end{aligned}$$

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

For a query Q, we use the *query completeness* statement $\mathit{Compl}(Q)$ to express that Q can be answered completely over the available database. Formally, $\mathit{Compl}(Q)$ is *satisfied* by a partial database instance D, denoted as $D \models \mathit{Compl}(Q)$, if $Q(\check{D}) = Q(\hat{D})$.

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

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

asking for all male students. This query will return the same result over the available database $\check{D}_S$ and the ideal database $\hat{D}_S$, namely it always returns *John* only. Thus, we can say that D_S satisfies the query completeness statement for the query Q_1, that is,

$$D_S \models \mathit{Compl}(Q_1).$$

$\square$

A table completeness statement allows one to express that a certain part of relation R is complete, without requiring completeness of other parts of the database. It has two components, the 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 already present in the available relation R.

Formally, let $R(\bar{x})$ be an R-atom whose arguments are distinct variables and let G be a condition such that $R(\bar{x}), G$ 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 $\mathit{Compl}(R(\bar{x}); G)$ is a *table completeness* statement. It has an *associated query*, which is defined as $Q_{R(\bar{x});G}(\bar{x}) :- R(\bar{x}), G$.

The statement is satisfied by a partial database D, written $D \models \mathit{Compl}(R(\bar{x}); G)$, if $Q_{R(\bar{x});G}(\hat{D}) \subseteq R(\check{D})$. 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 $R(\check{D})$.

Later on, we will denote a table completeness statement generically as C and refer to the associated query simply as Q_C.

Example 3. Consider again the partial database D_S defined above. There, we can observe that in the table *person*, only the teacher *Bob* is missing, thus it contains all students. We can say that the relation *person* is complete for all students. The table student still contains *John*, who is the only male student, thus, we can say that the relation *student* is complete for all male persons. Formally, this two observations can be written as table completeness statements

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

which are then satisfied by the partial database D_S. □

One may wonder whether table completeness can be expressed by query completeness. To show that this is not the case we go back to our example.

Example 4. Consider the table completeness statement C_1 that states that the relation *person* is complete for all students. The corresponding query Q_{C_1} that asks for all persons that are students is

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

Evaluating Q_{C_1} over $\hat{D}_S$ the result is $\{ \mathit{John}, \mathit{Mary} \}$. However, evaluating it over $\check{D}_S$ the result contains only *John*. Thus, D_S does not satisfy the query completeness for the query Q_{C_1} although it satisfies the table completeness statement C_1. □

Consider a query $Q(\bar{x}) :- A_1, \ldots, A_n, E_1, \ldots, E_m$, with relational atoms A_i and comparisons E_j. The *canonical completeness statement* for the atom A_i is the TC statement

$$C_i = \mathit{Compl}(A_i; A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n, E_1, \ldots, E_m).$$

Intuitively, C_i states that $\check{D}$ contains all the instances of A_i that contribute to an answer to Q over $\hat{D}$. We call $\mathcal{C}_Q = \{ C_1, \ldots, C_n \}$ the set of canonical completeness statements for Q.

Example 5. Consider the query

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

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

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

□

4 Completeness Reasoning

In this section we discuss the general completeness reasoning tasks. Before, we have seen the two formalism of table completeness (TC) and query completeness (QC). We consider TC statements to be the most useful formalism for giving assertions about database completeness, because they allow stating completeness of tables independent of the completeness status of other tables. We consider QC statements to be the most interesting kind of database completeness, i.e., one is usually interest whether given some completeness assertions, one can conclude that a query result will be complete or not.

Therefore, we identify the question of when table completeness entails query completeness (TC-QC) as the most interesting reasoning problem. We will reduce this problem to the problem of table completeness entailing table completeness (TC-TC) which we deal with in the next section. There, we show that the TC-TC entailment problem can be reduced to containment. We show that both TC-QC and TC-TC are decidable whenever the union containment problem of the underlying query language is. Finally, we will discuss the problem of QC-QC entailment and will show that the question of what necessary conditions are is an open question in the general case.

4.1 Table Completeness Entailing Table Completeness

table completeness statements describe parts of relations, which are stated to be complete. Hence, intuitively, a set of such statements entails another statement if the part described by the latter is contained in the parts described by the former.

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:

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

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:

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

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}. □

With the next lemma we show that, in fact, TC-TC entailment can naturally be reduced to query containment. We remind the reader that for a table completeness statement C the associated query is denoted as Q_C. Moreover, if C is a table completeness statement for relation R, then Q_C and R have the same arity.

Lemma 7. *Let C and $C_1, \ldots, C_n$ be table completeness statements for a relation R. Then*

$$C_1, \ldots, C_n \models C \quad \textit{iff} \quad Q_C \subseteq Q_{C_1} \cup \cdots \cup Q_{C_n}.$$

Proof. "⇒" When Q_C is not contained in $Q_{C_1} \cup \cdots \cup Q_{C_n}$, there exists a database instance I such that there is a tuple t in $Q_C(I)$ that is not in $Q_{C_1} \cup \cdots \cup Q_{C_n}$. We construct from I a partial database instance $D = (I \setminus \{ R(t) \}, I)$ where ideal and available database are exactly the same except that $R(t)$ is not in the available database. As t is not in $Q_{C_1} \cup \cdots \cup Q_{C_n}$, we conclude that $C_1, \ldots, C_n$ hold on this partial database instance, whereas C does not.

"⇐" Let $Q_C \subseteq Q_{C_1} \cup \cdots \cup Q_{C_n}$ and let D be a partial database instance satisfying $C_1, \ldots, C_n$. We show that D satisfies C as well. Let t be a tuple in $Q_C(\hat{D})$. We show that t is also in $R(\check{D})$.

As Q_C is contained in $Q_{C_1} \cup \cdots \cup Q_{C_n}$, the tuple t is also in $Q_{C_1}(\hat{D}) \cup \cdots \cup Q_{C_n}(\hat{D})$. Suppose, wlog, that $t \in Q_{C_i}(\hat{D})$. As C_i holds in D, we have that $Q_{C_i}(\hat{D}) \subseteq R(\check{D})$ and thus, t is also in $R(\check{D})$. □

Conversely, we can also reduce containment of unions of conjunctive queries to TC-TC entailment. Let $Q(\bar{x}) :- B$ be a query and R a new relation symbol, with the same arity as Q. The TC statement associated to Q and R is the statement $Compl(R(\bar{x}); B)$, which we denote as $C_{R,Q}$.

Lemma 8. *Let Q, $Q_1, \ldots, Q_n$ be conjunctive queries, all with the same arity, and let R be a new relation symbol, with the same arity as the queries. Then*

$$Q \subseteq Q_1 \cup \cdots \cup Q_n \quad \textit{iff} \quad C_{R,Q_1}, \ldots, C_{R,Q_n} \models C_{R,Q}.$$

Proof. "⇒" Let $Q \subseteq Q_1 \cup \ldots \cup Q_n$. We show that in every partial database instance D where $C_{R,Q_1}, \ldots, C_{R,Q_n}$ hold, also $C_{R,Q}$ holds. To this end we show that whenever a tuple t is in $Q_{C_{R,Q}}(\hat{D})$, then it is also in $R(\check{D})$.

Suppose that $t \in Q_{C_{R,Q}}(\hat{D})$. Since this query is defined as $Q_{C_{R,Q}}(\bar{x}) :- R(\bar{x}), B$, where B is the body of Q, it follows that $t \in R(\hat{D})$ and that $t \in Q(\hat{D})$. Due to the containment, we also have $t \in Q_1(\hat{D}) \cup \cdots \cup Q_n(\hat{D})$. Combining this with the fact that $t \in R(\hat{D})$, we conclude that $t \in Q_{R,Q_1}(\hat{D}) \cup \cdots \cup Q_{R,Q_n}(\hat{D})$. As $C_{R,Q_1}, \ldots, C_{R,Q_n}$ hold in D, it follows that $t \in R(\check{D})$.

"⇐" Assume $Q \not\subseteq Q_1 \cup \ldots \cup Q_n$. We have to show that $C_1, \ldots, C_n \not\models C$.

If $Q \not\subseteq Q_1 \cup \cdots \cup Q_n$, then there exists a database instance I such that there is a tuple t that is in $Q(I)$, but not in $Q_1(I) \cup \ldots \cup Q_n(I)$.

Let R_I denote the set of all atoms with relation symbol R where the argument is in $Q(I) \cup Q_1(I) \cup \ldots \cup Q_n(I)$. We construct a partial database instance $D = (\check{D}, \hat{D})$ out of I, where $\hat{D} = I \cup R_I$ and $\check{D} = I \cup R_I \setminus \{ R(t) \}$. That is, both the ideal and the available database contain I, plus all R-atoms for tuples that are in the answer of one of the queries over I, except that $R(t)$ is not in $\check{D}$.

Over D, the completeness statements $C_{R,Q_1}, \ldots, C_{R,Q_n}$ hold because all tuples that are in $Q_1(I) \cup \cdots \cup Q_n(I)$ are also in $R(\check{D})$. However, the completeness statement $C_{R,Q}$ does not hold in D because t is not in $R(\check{D})$. □

Since the problems of table completeness entailment and query containment can be reduced to each other, we conclude that both problems have the same complexity in any query language where the reductions are possible.

Theorem 9. *Let $\mathcal{Q}$ 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 have the same complexity.*

Proof. Follows from Lemmas 7 and 8. □

4.2 Table Completeness Entailing Query Completeness

Regarding the question when table completeness entails query completeness, a first observation is that under certain conditions, completeness of a query can exactly be characterized by its canonical table completeness statements.

Theorem 10. *Let Q be a conjunctive query. Then for all partial database instances D,*

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

provided one of the conditions below holds:

1. *Q is evaluated under multiset semantics, or*
2. *Q is a projection-free query.*

Proof. 1. "⇒" Indirect proof: Suppose, one of the completeness assertions in $\mathcal{C}_Q$ does not hold over 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.

1. "⇐" 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 D.

2. Follows from 1. Under multiset semantics, violation of a necessary completeness assertion leads to a difference of at least one occurence of some tuple in the results over $\hat{D}$ and $\check{D}$. Under set semantics, multiplicities collapse so the query could be still complete if there existed another way to compute that tuple in the result. However, observe, that without disjunction and projection, under set semantics, there exists only exactly one valuation per tuple in the result. □

From the theorem above we conclude immediately that the canonical table completness statements of a query are sufficient conditions for the completeness of that query.

Corollary 11. *Let Q be a conjunctive query and $\mathcal{C}$ be a set of table completeness statements. Then*

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

Proof. Instead of Q, we consider its projection-free variant $\mathcal{Q}'$. Note that $\mathcal{C}_Q = \mathcal{C}_{Q'}$. Thus, by the preceding theorem, if $D \models \mathcal{C}_Q$, then $D \models \mathit{Compl}(Q')$, 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, $D \models \mathit{Compl}(Q)$. □

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 D it holds that $D \models \mathcal{C}$ if and only if $D \models \mathit{Compl}(Q)$. The lemma above shows that the canonical statements $\mathcal{C}_Q$ are characterizing under set semantics for projection free Q.

One can show that under set semantics, satisfaction of the canonical completeness statements is still a sufficient but not necessary condition for query completeness. One may wonder whether this means that no set of TC statements is characterizing. In fact this is what the next theorem shows.

Theorem 12. *Let Q be a conjunctive query where at least one variable in the body is not a distinguished variable. Then no set of table completeness statements exists that is characterizing for Q.*

Proof. We show the nonexistence of a characterizing set of table completeness statements for a simple query first, and describe afterwards, how this proof extends to arbitrary queries.

Consider the relation schema $\Sigma = \{ R/1 \}$ and the boolean query $Q() :- R(x)$. Furthermore, assume a characterizing set of table completeness conditions $\mathcal{C}$ for Q existed. Now consider the partial database instances D_1, D_2 and D_3 such that:

$$\begin{aligned} \hat{D}_1 &= \{ R(a), R(b) \} & \check{D}_1 &= \{ R(a) \} \\ \hat{D}_2 &= \{ R(a), R(b) \} & \check{D}_2 &= \{ R(b) \} \\ \hat{D}_3 &= \{ R(a), R(b) \} & \check{D}_3 &= \{ \}. \end{aligned}$$

Then, $\mathit{Compl}(Q)$ holds in D_1 and D_2 but not in D_3, and therefore all table completeness conditions in $\mathcal{C}$ have to hold in D_1 and D_2, but at least one of them must not hold in D_3. Let us call that condition C.

The statement C must be of the form $\mathit{Compl}(R(x), G)$. Then $G = \mathit{true}$ does not hold in D_1 and D_2 (because in both cases there is a tuple in $\hat{R}$ that is not in $\check{R}$). 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 D_1 or D_2. So the only form G can have such that $\mathit{Compl}(R(x), G)$ holds in D_1 and D_2 is $G = \mathit{false}$. However, $\mathit{Compl}(R(x), \mathit{false})$ holds in D_3 as well.

The proof for this specific query can be extended to any query with projection. The idea is the same, one constructs three partial database instances,

where the ideal database instances contain the frozen body of the query plus an isomorphic structure differing only in a nondistinguished variable's name. The three available database instances are once the frozen body, once the isomorphic structure differing in a nondistinguished variable's name, 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. □

Since in general characterizing sets of TC statements do not exist for a conjunctive query Q, we cannot replace the statement $\mathit{Compl}(Q)$ by a set of TC statements for arbitrary reasoning tasks. However, as the next theorem shows, the set of canonical TC statements $\mathcal{C}_Q$ can replace $\mathit{Compl}(Q)$ when checking TC-QC entailment, provided Q is a minimal relational query.

Theorem 13. *Let Q be a minimal relational conjunctive query and $\mathcal{C}$ be a set of table completeness statements. Then*

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

Proof. Observe first that the theorem holds trivially for unsatisfiable queries, as the table completeness queries of the canonical completeness statements of unsatisfiable queries are unsatisfiable as well, and therefore the set $\mathcal{C}_Q$ of canonical completeness statements holds in any partial database.

The proof is 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 D such that $D \models \mathcal{C}$, but $D \not\models \mathcal{C}_Q$. Since $D \not\models \mathcal{C}_Q$, we find that D violates some canonical completeness statement in $\mathcal{C}_Q$. Let B be the body of Q. Wlog, assume that $D \not\models C_1$, where C_1 is the canoncial statement for $A_1 = R_1(\bar{t}_1)$, the first atom in B. Let Q_{C_1} be the query associated to C_1. Thus, there exists some tuple $\bar{u}_1$ such that $\bar{u}_1 \in Q_{C_1}(\hat{D})$, but $\bar{u}_1 \notin R_1(\check{D})$.

Now we construct a second partial database D_0. As Q is satisfiable, there exist satisfying valuations for its body B. As the domain of our constants is dense, we can pick a valuation that we call σ, where each variable in B that is not restricted to a certain constant, is mapped to a constant that does not appear in B nor another variable is mapped to by σ. Let B' be σB and $A'_1 = R(\sigma\bar{t}_1)$. Now, we define $D_0 = (B', B' \setminus \{ A'_1 \})$.

Claim: 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 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 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 $\bar{u}_1 \in Q_{C_1}(\hat{D})$, and Q_{C_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 D_0 satisfies $\mathcal{C}$.

Since D_0 satisfies $\mathcal{C}$ and $\mathcal{C} \models \mathit{Compl}(Q)$, it follows that Q is complete over D_0. As $\hat{D}_0 = B'$, the frozen body of Q, we find that $\bar{x}' \in \hat{Q}(D_0)$, with $\bar{x}'$ being the frozen version of the distinguished variables $\bar{x}$ of Q. As Q is complete over 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. □

With the next example, we show that query minimality is really a necessary condition for the theorem given above to hold.

Example 14. Consider $Q(x) :- R(x,x), R(x,y)$, which is not minimal, because there exists a homomorphism, mapping y to x, allowing one to drop the second literal. Its set of canonical completeness statements is $\mathcal{C}_Q = \{ C_1, C_2 \}$, where

$$C_1 = \mathit{Compl}(R(x,x);\, R(x,y))$$

$$C_2 = \mathit{Compl}(R(x,y);\, R(x,x)).$$

Statement C_1 is satisfied by a partial database $(\hat{D}, \check{D})$ if $\check{D}$ contains all atoms $R(a,a) \in \hat{D}$ for which there is some matching atom $R(a,b) \in \hat{D}$, which is equivalent to the condition that $\check{D}$ contain all atoms $R(a,a) \in \hat{D}$. Statement C_2 is satisfied by $(\hat{D}, \check{D})$ if $\check{D}$ contains all atoms $R(a,b) \in \hat{D}$ for which there is a matching symmetric atom $R(a,a) \in \hat{D}$. Clearly, C_2 entails C_1, but C_1 does not entail C_2.

However, Q is complete already whenever C_1 holds. This follows from the fact that whenever we have a symmetric tuple that satisfies the first literal in Q, the same tuple satisfies also the second literal without influencing on the query result. Thus, with $\mathcal{C} = \{ C_1 \}$ we have that $\mathcal{C} \models \mathit{Compl}(Q)$, but $\mathcal{C} \not\models \mathcal{C}_Q$. □

Table 1 gives an overview over the complexity of TC-TC entailment with respect to different languages used for entailing and entailed TC statements. The complexity results shown there follow from Theorem 9 about the equivalence of query containment and TC-TC entailment. Query containment can be characterized by the existence of query homomorphisms. The table shows both the hardness and the tractability for different classes of TC statements.

As long as the containee queries contain no repeated relation symbols, the containment problems are in coNP because there is no nondeterminism in constructing query homomorphisms (each atom in the container queries can only be mapped to exactly one atom in the containee query). To show that containment does not hold, it suffices to guess one linearisation of the containee query for which the containment does not hold, and show that no query homomorphism exists, which then can be done in PTIME. The coNP-hardness can be shown by a reduction of propositional validity, a result that, to the best of our knowledge, has not appeared in the literature so far.

As long as the container queries contain no comparisons, the containment problems are in NP because containment of some query in a union of queries without comparisons holds exactly if the query is contained in one of those queries [SY78].

Results about the complexity of containment for queries containing comparisons can be found in [vdM92].

		Language 2	
		CQs w/o rep.	CQs, CQs w/ c.
Language 1	CQs	polynomial	NP-complete
	CQs w/ c.	coNP-complete	Π_2^P-complete

Table 1: Complexity of deciding whether a set of TC statements in language 1 entails a set of TC statements in language 2. CQs w/o rep. denotes conjunctive queries without repeated relation symbols. CQs w/ c. denotes conjunctive queries with comparisons

4.3 Query Completeness Entailing Query Completeness

To find out whether completeness of a set queries entails completeness of a given query, Motro [Mot89] 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 answer sets of the complete queries, which are complete.

Example 15. Consider the query $Q(x) :- R(x), S(x), T(x)$. Furthermore, assume that the queries Q_1 and Q_2 below have been asserted to be complete:

$$Q_1(x) :- R(x), S(x),$$
$$Q_2(x) :- T(x).$$

Observe that Q can be rewritten in terms of Q_1 and Q_2 as

$$Q(x) :- Q_1(x), Q_2(x).$$

That is, the result of the query Q is the intersection of the results of the queries Q_1 and Q_2. Thus, completeness of Q_1 and Q_2 entails completeness of Q. □

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 I_1 and I_2, we have that $Q'(I_1) = Q'(I_2)$ for all $Q' \in \mathcal{Q}$ implies $Q(I_1) = Q(I_2)$. The decidability of query determinacy for conjunctive queries is an open question so far. But as shown by Segoufin and Vianu [SV05], 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 16. *Let $\mathcal{Q} \cup \{ Q \}$ be a set of queries. Then*

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

Proof. The definitions of query determinacy and QC-QC entailment are exactly the same, except that query determinacy considers arbitrary database instances I_1, I_2, while QC-QC entailment considers only partial databases, that is pairs of instances (I_1, I_2) where $I_1 \supseteq I_2$. □

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 only able to show this for conjunctive queries that are boolean and relational.

Theorem 17. *Let $\mathcal{Q} \cup \{ Q \}$ be a set of boolean relational conjunctive queries. Then*

$$\mathcal{Q} \twoheadrightarrow Q \text{ if } \mathit{Compl}(\mathcal{Q}) \models \mathit{Compl}(Q).$$

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 we omit details here but say that whenever Q cannot be rewritten in terms of $\mathcal{Q}$, then a counterexample of a partial database instance 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.

5 Reasoning with Finite Domains

A question of practical interest is completeness reasoning with finite domains. If we know that an attribute of a relation has a finite domain of possible values, we can derive more completeness statements.

Example 18. Consider the query

$$Q(n, g) :- \mathit{person}(n, g),$$

asking for all persons, and the table completeness statements

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

which state that all male and female persons are present in the *person* table.

Observe first that, in general, completeness of the *person* table for all male and female persons does not imply completeness of the entire table. For instance, there could be persons with the value *'unkown'* as gender.

However, if there was a formal constraint allowing the values of the gender attribute to be only *'male'* or *'female'*, then C_1 and C_2 would entail $\mathit{Compl}(\mathit{person}(n, g);\ \mathit{true})$, that is, completeness of the whole *person* table, and completeness of the query Q could be derived. □

As both TC-TC entailment and TC-QC entailment for relational conjunctive queries can be reduced to query containment, a possible approach to completeness reasoning with finite domains is to extend containment reasoning to finite domains.

Formally, we say that a *finite domain constraint* is a triple $F = Dom(R, A, T)$ containing a relation name R, a set A of positions of R, and a finite set T of A-tuples. A database instance I *satisfies* a finite domain constraint F, if the projection on the positions A of the extension of R is contained in T, that is, $\pi_A(R(I)) \subseteq T$.

Given a set of finite domain constraints $\mathcal{F}$ and a set of conjunctive queries $\mathcal{Q} \cup \{ Q \}$, we say that Q is *contained in* $\mathcal{Q}$ *w.r.t.* $\mathcal{F}$, written

$$Q \subseteq_{\mathcal{F}} \mathcal{Q},$$

if $Q(I) \subseteq \bigcup_{Q' \in \mathcal{Q}} Q'(I)$ for all instances I satisfying $\mathcal{F}$.

In a very naive way, one can test whether finite domain containment holds by instantiating the query Q in all possible ways using the constraints in $\mathcal{F}$, and checking for each such instantiation θQ, whether θQ is contained in the union of the queries in $\mathcal{Q}$. Clearly, the number of instantiations to consider is finite, but exponential in the size of $\mathcal{F}$. In this way we can translate one problem "$Q \subseteq_{\mathcal{F}} \mathcal{Q}$" w.r.t. finite domain constraints into many problems "$\theta Q \subseteq \mathcal{Q}$" without constraints. Thus, whenever containment without constraints can be decided in Π_2^P, so can containment wrt finite domain constraints.

The question arises whether adding finite domain constraints increases the difficulty of containment for classes of queries where containment is not Π_2^P-hard. In fact, as can be shown by a reduction of the validity problem for universally quantified 3-SAT formulae, the difficulty is raised for relational conjunctive queries.

Theorem 19. *Finite domain containment of relational conjunctive queries is* Π_2^P*-complete.*

Proof. The reduction for the hardness proof is included in the appendix. The upper bound follows from the algorithm sketched above. □

6 Reasoning with Database Instances

A second question of practical interest is how to take advantage of the data in an concrete instance for completeness reasoning. In such a situation, completeness statements explicitly make available parts of the ideal database allow one to derive further conclusions about table and query completeness.

Example 20. As a very simple example, consider the query

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

that asks for the names of all students that attend language courses in Greek.

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 we know that in the available instance of our database, which is the one maintained by the school administration, the table *language_attendance* contains no records about Greek. So the result of Q evaluated over this available database will always be empty.

As the *language_attendance* table is asserted to be complete, it cannot have any records about Greek in the ideal instance either. So Q evaluated over the

ideal database must be empty as well, and thus, Q is necessarily complete. Without considering this concrete available database instance the conclusion would not hold. □

Formally, the question of *TC-QC entailment w.r.t. a concrete 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

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

For relational conjunctive queries and TC statements, a naive algorithm to check whether $\mathit{Compl}(Q)$ is entailed by $\mathcal{C}$ with respect to $\check{D}$ works as follows: First, one evaluates Q over $\check{D}$. Then, one tries to construct an ideal database instance $\hat{D}$ such that the evaluation of Q over $\hat{D}$ returns a tuple that is not returned over $\check{D}$. If this construction succeeds, then the entailment does not hold. If such a construction is not possible, then completeness of the query Q is entailed by $\mathcal{C}$ and $\check{D}$.

For constructing the test databases, it suffices to add to $\check{D}$ instantiations of the body of Q and to use for these instantiations only the constants appearing in $\check{D}$ plus a fresh constants for every variable of Q. The algorithm has a complexity of Π_2^P, because to show that the entailment does not hold it suffices to guess one such instantiation of the body of Q, to add it to $\check{D}$ and evaluate Q over the extended database.

For the lower complexity bound of TC-QC entailment w.r.t. a database instance, we found again that the problem of validity of universally quantified 3-SAT formulas can be reduced to it. The reduction is enclosed in the appendix.

Theorem 21. *The problem, given a relational conjunctive query Q, a set of relational conjunctive TC statements $\mathcal{C}$, and a database instance $\check{D}$, to check whether*

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

is Π_2^P-complete.

Proof. The reduction for the hardness proof is included in the appendix. The upper bound follows from the algorithm sketched above. □

Most notably again, we observe that the problem is strictly more difficult than TC-QC entailment without taking into account an available database instance.

7 Where Completeness Statements Come From

So far we have seen how completeness statements can be derived from given completeness statements. So obviously, the correctness of the derived completeness statements relies on the correctness of the given completeness statements. But where can given completeness statements can come from? Why someone can give completeness statements? Only if the incomplete database is derived from another complete one but does not contain all the facts of the complete one e.g.,

for performance reasons (mirror) or authorization reasons, completeness statements can be derived automatically. Otherwise, there must be other reasons to give completeness statements. We identified three general basic principles:

1. Someone may perfectly know some part of the ideal world. Then, he/she can manually compare whether all facts that should be in some part of the available database are really there, and if so, state completeness of that part. This method seems only appropriate for small data sets.

Example 22. A class teacher knows all his/her students. Having a printout of all students stored in the database that are enrolled in his class, he/she can manually compare whether all are there. □

2. Someone may know that the method of data collection is complete. If so, he/she can state that after the data collection has finished the data is complete, without inspecting the data.

Example 23. Enrolment forms have to be sent or handed in to the secretariat by the deadline of the enrolment. So by the deadline of enrolment the set of all valid enrolment forms is complete, as any forms that are not there yet are not valid. So one can state completeness, although no one could manually inspect the set of enrolment forms and check whether they are complete, since hardly anyone knows all the students that apply for enrolment. □

3. Someone may know that the data is correct and know the number of tuples that should be there in some part. He/she then can compare whether the number of tuples that should be there matches the number of tuples that are there. If that holds, he/she can state completeness for that part.

Example 24. Under the reasonable assumption that no one enters a nonexisting school into the database, whenever one knows the number of schools in South Tyrol, e.g., from another data source, and one finds this number of schools in the database, then one can state that the table containing all the schools is complete. This works, although no one could manually inspect that table and check whether any school is missing. □

8 Conclusion

We outlined the importance of data completeness in the field of data quality and illustrated the main research questions by the example of the management of school data in the province of South Tyrol. 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 [Mot89] and Halevy [Lev96]. We distinguished between the table completeness (TC) statements introduced by Halevy, and the query completeness (QC) statements introduced by Motro. We identified three central completeness reasoning tasks, (i) entailment of QC statements by TC statements or TC-QC entailment for short, which is the most relevant for applications, as well as (ii) TC-TC entailment and (iii) QC-QC entailment. For the problem of TC-TC

entailment, we showed the equivalence to query containment. For the problem of TC-QC entailment for relational conjunctive queries, we presented a reduction to the TC-TC entailment problem. This closes a crucial gap in previous work by Halevy [Lev96]. 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 TC-QC entailment in the presence of a database instance and TC-TC entailment in the presence of finite domains become harder, and we presented the problem of query containment with respect to finite domains. We identified three basic principles for giving completeness assertions. In addition, we showed that TC-TC entailment becomes harder if one takes into account finite domain constraints and that TC-QC entailment becomes harder if one reasons w.r.t. a concrete database instance. We also discussed how completeness assertions could be gathered in an organisation.

A limitation of previous work, which we have not yet addressed, is that databases are assumed to be null free. This is the next topic that we plan to investigate.

Acknowledgement

We are thankful to Zeno Moriggl and Martin Prosch from the school IT department of the province of South Tyrol for introducing us to their practical problem.

References

[AKG87] S. Abiteboul, P.C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *Proc. SIGMOD*, pages 34–48, 1987.

[ASV06] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and querying XML with incomplete information. *ACM TODS*, 31(1):208–254, 2006.

[BNQ06] J. Biswas, F. Naumann, and Q. Qiu. Assessing the completeness of sensor data. In *Proc. DASFAA*, pages 717–732, 2006.

[Cod75] E. F. Codd. Understanding relations (installment #7). *FDT – Bulletin of ACM SIGMOD*, 7(3):23–28, 1975.

[DCCBA10] 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.

[DLS00] P. Doherty, W. Lukaszewicz, and A. Szalas. Efficient reasoning using the local closed-world assumption. In *AIMSA*, pages 49–58, 2000.

[EGW97] O. Etzioni, K. Golden, and D. S. Weld. Sound and efficient closed-world reasoning for planning. *AI*, 89(1-2):113–148, 1997.

[Elk90] Ch. Elkan. Independence of logic database queries and updates. In *Proc. PODS*, pages 154–160, 1990.

[FG09] W. Fan and F. Geerts. Relative information completeness. In *PODS*, pages 97–106, 2009.

[FG10] W. Fan and F. Geerts. Capturing missing tuples and missing values. In *PODS*, pages 169–178, 2010.

[FKMP02] R. Fagin, Ph. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. ICDT*, pages 207–224, 2002.

[IL84] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31:761–791, 1984.

[Len02] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS*, pages 233–246, 2002.

[Lev96] A.Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. VLDB*, pages 402–412, 1996.

[LS93] A.Y. Levy and Y. Sagiv. Queries independent of updates. In *Proc. VLDB*, pages 171–181, 1993.

[Mot89] A. Motro. Integrity = Validity + Completeness. *ACM TODS*, 14(4):480–502, 1989.

[NFL04] F. Naumann, J.-Chr. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29:583–615, September 2004.

[SV05] L. Segoufin and V. Vianu. Views and queries: Determinacy and rewriting. In *Proc. PODS*, pages 49–60, 2005.

[SY78] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operation. In *VLDB*, pages 535–548, 1978.

[vdM92] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *PODS*, pages 331–345, 1992.

A Reduction of ∀3-SAT to Finite Domain Containment

To show the Π_2^P-hardness of finite domain containment, we give a reduction of the problem of deciding validity of universally quantified 3-satisfiability to finite domain containment.

A universally quantified 3-SAT formula is a formula of the form

$$\forall X_1, \ldots, X_m \exists Y_1, \ldots, Y_n : C_1 \wedge \ldots \wedge C_k,$$

where clause C_i is of the form $L_{i1} \vee L_{i2} \vee L_{i3}$ with L_{i1}, L_{i2}, L_{i3} being literals using propositions from $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$.

Let ϕ be a formula of the above form. We create an instance of a finite domain containment problem such that finite domain containment holds exactly if ϕ is valid.

We use the database schema $\Sigma = \{ R_1/2, \ldots, R_m/2, S/2, C'_1/3, \ldots C'_k/3 \}$. For a clause C_i with literals L_{i1}, L_{i2} and L_{i3} in ϕ, we define conjunctive conditions

$$G_i = R_i(a, W_i), R_i(W_i, b), S(W_i, 0), S(b, 1)$$
$$G'_i = R_i(a, b), S(b, X_i).$$

Furthermore, the set $C_i'^{(7)}$ denotes the set of the 7 ground instances of predicate C'_i over the domain $\{ 0, 1 \}$, such that constant 1 at position j in C'_i corresponds to the variable Z_{ij} being mapped to true, and the 7 ground instances are the ones where C_i evaluates to true under the variable mapping.

Let F be the finite domain constraint with

$$F = Dom(S, 1, \{a, b\}).$$

Let Q_1 and Q_2 be the following queries:

$$Q_1() :- G_1, \ldots, G_m, C_1'^{(7)}, \ldots, C_k'^{(7)}$$
$$Q_2() :- G'_1, \ldots, G'_m, C'_1, \ldots, C'_k$$

Lemma 25. *Let ϕ be a ∀3-SAT formula as shown above and let Q_1, Q_2 and F be constructed as above. Then*

$$\phi \text{ is valid exactly if } Q_1 \subseteq_F Q_2.$$

Proof. For containment to hold, both each conjunctive condition C'_i has to be contained in the conjunctive condition $C_i^{(7)}$, and the conjunctive condition G'_j in the conjunctive conditions G_j.

The key of that containment is that the variables W_i in G_i are not restricted further than to have the constant values a or b. The value of W_i determines which value X_i has, thus, X_i can be both 0 or 1. Thus, the indeterminacy of W_i leads to containment having to hold both in the case of X_i being 0 or 1, representing the universal quantification of the X variables in ϕ.

The remaining part of the containment problem is standard, for every possible assignment of the X variables, there must exist a valuation of the Y variables

such that each C' clause becomes a ground instance for which the clause evaluates to true.

The reduction is correct, because whenever containment holds, an assignment for the Y variables for each combination of X variables has to exist that enables the containment of the C' in the $C^{(7)}$. It is complete because whenever ϕ is valid, for every combination of the X variables, some combination of the Y variables has to exist that makes all the clauses true. □

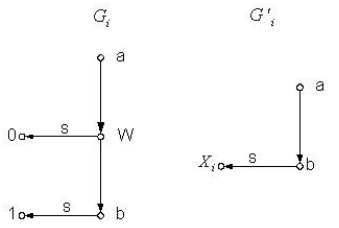

Figure 1: Structure of G_i and G'_i. Depending on the value assigned to W, X_i becomes either 0 or 1.

B Reduction of ∀3-SAT to TC-QC Entailment w.r.t. a Concrete Database Instance

To show the Π_2^P-hardness of TC-QC entailment w.r.t. a concrete database instance, we give a reduction of the same problem as in the previous chapter, validity of universally quantified 3-satisfiability.

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

$$\forall X_1, \ldots, X_m \exists Y_1, \ldots, Y_n : C_1 \wedge \ldots \wedge C_k.$$

We define the query completeness problem

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

as follows. Let the relation schema Σ be $\{\, B/1, R_1/1, \ldots, R_m/1, C'_1/3, \ldots, C'_k/3 \,\}$. Let Q be a query defined as

$$Q() :- B(X_1), R_1(X_1), \ldots, B(X_m), R_m(X_m).$$

Let $\check{D}$ be such that $B(\check{D}) = \{\, 0, 1 \,\}$, 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 C_i becomes true when the variables in C_i become the truth values assigned that correspond to 0 and 1.

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

$$\begin{aligned} &\mathit{Compl}(B(x), \mathit{true}) \\ &\quad \mathit{Compl}(R_1(X_1); R_2(X_2), \ldots, R_m(X_m), \\ &\qquad\qquad C'_1(\bar{Z}_1), \ldots, C'_k(\bar{Z}_k)), \end{aligned}$$

where the $\bar{Z}_i$ are 3-tuples of variables in $\{ X_1, \ldots, X_m \} \cup \{ Y_1, \ldots, Y_n \}$ as in ϕ.

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

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

Proof. Observe first, that validity of ϕ implies that for every possible combination of the X variables, there exist Y variables such that C_1 to C_k 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 addings may be allowed. That is, every such adding 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 addings 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. □