On Querying Incomplete Information in Databases under Bag Semantics

Marco Console , **Paolo Guagliardo** and **Leonid Libkin**
University of Edinburgh

Abstract

Querying incomplete data is an important task both in data management, and in many AI applications that use query rewriting to take advantage of relational database technology. Usually one looks for answers that are certain, i.e., true in every possible world represented by an incomplete database. For positive queries – expressed either in positive relational algebra or as unions of conjunctive queries – finding such answers can be done efficiently when databases and query answers are sets. Real-life databases however use bag, rather than set, semantics. For bags, instead of saying that a tuple is certainly in the answer, we have more detailed information: namely, the range of the numbers of occurrences of the tuple in query answers. We show that the behavior of positive queries is different under bag semantics: finding the minimum number of occurrences can still be done efficiently, but for maximum it becomes intractable. We use these results to investigate approximation schemes for computing certain answers to arbitrary first-order queries that have been proposed for set semantics. One of them cannot be adapted to bags, as it relies on the intractable maxima of occurrences, but another scheme only deals with minima, and we show how to adapt it to bag semantics without losing efficiency.

1 Introduction

In many problems lying at the intersection of AI and data management, one tries to recast key tasks as evaluation of database queries, in an attempt to leverage existing database technology. Examples include data integration and data exchange, where finding answers over integrated and exchanged data is often achieved by first rewriting a query and then running it against a constructed database or data source [Arenas *et al.*, 2014; Calì *et al.*, 2003b; Lenzerini, 2002]. Another example is ontology-based query answering, where one attempts to rewrite a query in a standard query language and run it on the original or modified database [Calvanese *et al.*, 2007; Gottlob *et al.*, 2014; Kontchakov *et al.*, 2011].

There are two features common to all these approaches. First, they try to take advantage of decades of research into efficient evaluation of SQL queries over relational databases. Second, the databases on which queries are executed are typically incomplete: for example, they often have null values added to them by applying a version of the chase procedure. Thus, one ends up using techniques for answering queries on databases with nulls. Such techniques usually produce correct answers for positive queries (i.e., unions of conjunctive queries) and quickly descend into intractability outside this class [Abiteboul *et al.*, 1995; Imielinski and Lipski, 1984], partly explaining the prevalence of positive queries in this line of research.

Despite many visible successes of applying traditional database technology in the above tasks, there is one notable mismatch between theoretical work and what happens in real life. Real SQL databases use *bag*, or multiset semantics. That is, database relations and query results may contain duplicates; query evaluation algorithms apply special rules that correctly count occurrences of tuples in their results. However, the standard notions of query answering over incomplete databases, such as certain answers, have been primarily worked out under the set semantics.

It is well known that the set/bag mismatch can have a profound effect on many results about query evaluation and reasoning about queries. For example, the standard problem of conjunctive query containment – which is essentially testing for the existence of a homomorphism, or a solution to a non-uniform CSP instance; cf. [Chandra and Merlin, 1977; Kolaitis, 2003] – is NP-complete under set semantics, but its exact complexity under bag semantics remains a major open problem [Chaudhuri and Vardi, 1993; Jayram *et al.*, 2006]. The complexity of basic query languages such as relational algebra remains tractable but moves up to a different complexity class, which requires new techniques for analyzing its expressiveness [Grumbach and Milo, 1996; Libkin and Wong, 1997]. And, when it comes to answering queries over bag-based databases with nulls, there is still no understanding of how the problem behaves even for positive queries.

Keeping track of the number of occurrences of a record is important in many real-life applications, in particular those where data from multiple independent sources is integrated or merged. But the need for bag semantics is more basic than

that, for at least two reasons. The first is efficiency; to leverage existing technology, the vast majority of the approaches to handling data rely on commercial DBMSs as a backend (this is true for data integration, data exchange, OBDA, etc.). As prescribed by the SQL Standard, these systems use bag semantics in query evaluation. Set semantics in SQL can be enforced, but at a high computational cost and thus it is normally avoided. The second reason is that query results are often used as input for further data analytics; in these cases, preserving duplicates is essential for correctly computing aggregates over them. Thus, if certain answers over incomplete databases are to be used in further processing, we ought to ensure that they retain information about record occurrences.

In light of these observations, our primary goal is to initiate the study of answering queries on incomplete databases under bag semantics, to move it closer to real-life applications that use SQL queries in relational DBMSs. At first it might seem that at least for positive queries there should be no problem in lifting results from sets to bags. We show that this is not so.

An incomplete database defines a set of possible worlds. When we evaluate a query over an incomplete database, for each candidate answer tuple we want to know how it occurs in query answers over possible worlds: for example, in all of them (certainty) or in some of them (possibility). When we deal with bags, we need to look at the *number* of occurrences. This gives us an interval between the minimum and the maximum numbers of occurrences. For sets, these numbers could be only zero or one, but for bags these could be arbitrary natural numbers. What we show is that, for positive queries, the minimum can be computed efficiently, by modifying set-based techniques appropriately, but finding the maximum becomes *intractable*. In fact it is already intractable in very simple settings.

In essence, our results say that for unions of conjunctive queries, which dominate many applications that require dealing with incomplete information, one can rely on query evaluation provided by commercial DBMSs and be certain about the minimum number of occurrences of tuples in the result. But getting additional information about such numbers is computationally intractable.

These results also shed some light on handling queries that go beyond unions of conjunctive queries. Finding answers to such queries that come with correctness guarantees is known to be coNP-hard [Abiteboul *et al.*, 1991], so it is natural to look at *approximations*. The idea was already pursued in [Reiter, 1986; Vardi, 1986] in the context of databases represented as logical theories, but recently it was shown that it can be made to work in the context of standard relational DBMSs [Libkin, 2016; Guagliardo and Libkin, 2016]. The key idea was to devise approximation schemes that, unlike the native evaluation of SQL queries, come with correctness guarantees. Two such schemes were developed under set semantics; one of their features is that for positive queries, they can rely on the standard query evaluation. Here we look at these schemes under bags and show that one of them requires computing an intractable query, while the other remains tractable. This observation provides a theoretical justification for results empirically observed in [Guagliardo and Libkin, 2016].

Organization Section 2 introduces basic notations. In Section 3 we give the semantics of bag relational algebra. Section 4 proves that minimal numbers of occurrences can be found efficiently for positive queries under bag semantics, and Section 5 shows that for maximal numbers the problem is intractable. In Section 6 we discuss implications for approximating certain answers; Section 7 gives concluding remarks.

2 Preliminaries

Incomplete databases An incomplete database D is a way to represent many complete databases (i.e., possible worlds); the set of those is referred to as the *semantics* of D, denoted by $[\![D]\!]$. The model of incompleteness that we use here is the very common model of *marked* or *labeled nulls*. It is not only common in databases [Abiteboul *et al.*, 1995; Imielinski and Lipski, 1984] but naturally occurs in the applications we mentioned before. For instance, the chase procedure – which is heavily used in data integration, exchange, as well as ontology-based querying – populates databases with both known constants and marked nulls, cf. [Arenas *et al.*, 2014; Kontchakov *et al.*, 2011; Lenzerini, 2002].

In this model databases are populated by *constants* and *nulls*, coming from two disjoint countably infinite sets denoted by Const and Null, respectively. Nulls are denoted by $\bot$, sometimes with sub- or superscripts. A database D is then a set of relations over $\mathsf{Const} \cup \mathsf{Null}$; a k-ary relation is a finite subset of $(\mathsf{Const} \cup \mathsf{Null})^k$. We write $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$ for the set of constants and nulls that occur in D. A database is complete if $\mathsf{Null}(D) = \emptyset$.

The semantics $[\![D]\!]$ is defined via *valuations* v which are mappings $v : \mathsf{Null}(D) \to \mathsf{Const}$. Then

$$[\![D]\!] \;=\; \{v(D) \mid v \text{ is a valuation}\}\,,$$

where $v(D)$ is the complete database obtained by replacing each null $\bot$ with $v(\bot)$. This is usually referred to as the closed-world semantics of incompleteness [Reiter, 1977].

In the applications we mentioned, both closed and open world semantics are used [Arenas *et al.*, 2014; Lenzerini, 2002; Lutz *et al.*, 2015]. In the study of incompleteness in databases, closed world semantics is more common. It has better complexity for more expressive queries, and under set semantics makes no difference for answering positive queries [Imielinski and Lipski, 1984]; thus we use it in this study.

Query answering A relational query Q of arity k takes a complete database D and returns a set of k-tuples over $\mathsf{Const}(D)$. If such a query Q is asked on an incomplete database D, to answer it one has to analyze the behavior of Q on the elements of $[\![D]\!]$. Using the characteristic function of a tuple $\bar{c}$ in a relation R

$$\#(\bar{c}, R) \;=\; \begin{cases} 1 & \text{if } \bar{c} \in R \\ 0 & \text{if } \bar{c} \notin R \end{cases}$$

we define

$$\begin{aligned} \min_Q(D, \bar{c}) \;&=\; \min_{D' \in [\![D]\!]} \#(\bar{c}, Q(D'))\,, \\ \max_Q(D, \bar{c}) \;&=\; \max_{D' \in [\![D]\!]} \#(\bar{c}, Q(D'))\,. \end{aligned} \tag{1}$$

The cases of interest to us are:

- $\min_Q(D, \bar{c}) = 1$. Then we know that $\bar{c}$ is always in $Q(D')$ for $D' \in [\![D]\!]$ and thus it is a *certain answer*.
- $\max_Q(D, \bar{c}) = 1$. Then we know that $\bar{c}$ is in $Q(D')$ for some $D' \in [\![D]\!]$ and thus it is a *possible answer*.

For a large class of queries, namely *positive queries* (or, equivalently, unions of conjunctive queries) calculating $\min_Q$ and $\max_Q$ can be done efficiently. We define this class of queries using the positive operations of *relational algebra*:

- Selection $\sigma_{i=j}$, when applied to a k-ary relation R with $k \geq i, j$, returns the set of tuples $(a_1, \ldots, a_k) \in R$ such that $a_i = a_j$.
- Projection $\pi_{\bar{i}}$ (omitting the ith component), when applied to a k-ary relation R with $k \geq i$, returns the set of tuples $(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k)$ for $(a_1, \ldots, a_k) \in R$.
- Cartesian product $\times$, when applied to a k-ary relation R and an m-ary relation S produces a $k + m$-ary relation with tuples $(a_1, \ldots, a_k, b_1, \ldots, b_m)$ where $(a_1, \ldots, a_k) \in R$ and $(b_1, \ldots, b_m) \in S$.
- Union $R \cup S$ can be applied to two relations of the same arity.

Queries of positive relational algebra (RA^+) are built from relations and these operation, e.g., $(\sigma_{2=3}(R) \times \pi_{\bar{3}}(S)) \cup T$. Note that often RA^+ expressions are presented by allowing selections with a conjunction of equalities or projections on a group of attributes, but these are easily expressed in the version we have.

Full relational algebra (RA) adds the difference operation: $R - S$ contains tuples in R but not in S. In terms of expressiveness, RA has precisely the power of first-order logic (FO), while RA^+ corresponds to existential positive FO, i.e., formulae built from atoms using $\exists, \wedge, \vee$. These have the same power as unions of conjunctive queries (which are the $\exists, \wedge$-fragment of FO).

The following is well known [Abiteboul *et al.*, 1991; Imielinski and Lipski, 1984].

Fact 1. *For every RA^+ query Q, checking each of the conditions $\min_Q(D, \bar{c}) = 1$ and $\max_Q(D, \bar{c}) = 1$ can be done in polynomial time in the size of D.*

For full RA, these problems are in coNP and NP respectively, and there are RA queries for which they are coNP-complete and NP-complete in terms of data complexity. Data complexity [Vardi, 1982] refers to the computational complexity of the problem with query Q fixed, and $D, \bar{c}$ given as the input. In what follows all the complexity results will be stated in terms of data complexity.

3 Bag Semantics

Real-life databases are not based on sets but rather on bags (multisets): each tuple can occur multiple times in a database relation and consequently in query results [Date and Darwen, 1996]. We extend the notion of $\#(\bar{a}, R)$ from sets to bags: now for a tuple $\bar{a}$ and a relation R, the expression $\#(\bar{a}, R)$ denotes the number of occurrences of $\bar{a}$ in R; if $\bar{a}$ does not occur in R, we set $\#(\bar{a}, R) = 0$.

For RA queries, the syntax of basic operations does not change, but they are given bag semantics, and some operations that were derivable under set semantics are added. The interpretations of selection, projection, product, and union are changed as follows:

- Selection $\sigma_{i=j}$: for each tuple $\bar{a}$,

$$\#(\bar{a}, \sigma_{i=j}(R)) \;=\; \begin{cases} \#(\bar{a}, R) & \text{if } a_i = a_j \\ 0 & \text{otherwise} \end{cases}$$

- Projection $\pi_{\bar{i}}$:

$$\#((a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k), \pi_{\bar{i}}(R)) \\ = \sum \#((a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_k), R)$$

  with summation over all elements a that occur in R.
- Cartesian product: $\#((\bar{a}, \bar{b}), R \times S) = \#(\bar{a}, R) \cdot \#(\bar{b}, S)$.
- Union: $\#(\bar{a}, R \cup S) = \#(\bar{a}, R) + \#(\bar{a}, S)$

For relational algebra on sets, intersection is easily expressible with σ, π, and $\times$. This is not the case for bags [Libkin and Wong, 1997] so we shall add this operation explicitly, under the standard semantics:

- Intersection: $\#(\bar{a}, R \cap S) = \min(\#(\bar{a}, R),\ \#(\bar{a}, S))$.

We further add the operation ε of *duplicate elimination* (which in the case of sets is simply the identity):

- Duplicate elimination: $\#(\bar{a}, \varepsilon(R)) = \min(\#(\bar{a}, R), 1)$.

These operations, $\sigma, \pi, \times, \cup, \cap$, and ε, constitute RA^+ for bags. Full RA adds the operation $-$ (difference) with the semantics $\#(\bar{a}, R - S) = \max(\#(\bar{a}, R) - \#(\bar{a}, S), 0)$.

When it comes to query answering, the definition (1) still applies: $\min_Q(\bar{c}, D)$ is the minimum number of occurrences of $\bar{c}$ in $Q(D')$ for $D' \in [\![D]\!]$, and $\max_Q(\bar{c}, D)$ is the maximum such number. That is, we know with certainty that every query answer must contain at least $\min_Q(\bar{c}, D)$ occurrences of $\bar{c}$, and no answer will contain more than $\max_Q(\bar{c}, D)$.

In what follows, we look at *data complexity* of query answering: that is, for a fixed query Q, we want to compute $\min_Q(D, \bar{c})$ or $\max_Q(D, \bar{c})$ on the input that consists of a database D and a tuple $\bar{c}$. When we talk about complexity in terms of complexity classes, especially hardness results, we view these as decision problems, i.e., comparing $\min_Q(D, \bar{c})$ and $\max_Q(D, \bar{c})$ with a given number m.

Firstly, we note that these problems are in the first level of the polynomial hierarchy.

Proposition 1. *For every RA query Q interpreted under bag semantics, and for every $m \in \mathbb{N}$, checking whether $\min_Q(D, \bar{c}) > m$ or whether $\max_Q(D, \bar{c}) < m$ can be done in* coNP *with respect to data complexity.*

Furthermore, there are queries Q and numbers $m \in \mathbb{N}$ such that the above problems are coNP*-complete in data complexity.*

Of course this means that complements, $\min_Q(D, \bar{c}) \leq m$ and $\max_Q(D, \bar{c}) \geq m$, are in NP and can be NP-complete for some Q and m. These bounds follow from the fact that, like in the set case, it suffices to consider valuations v on D whose

range consists of $\mathsf{Const}(D)$ and a new distinct constant for each null in $\mathsf{Null}(D)$, and that relational algebra, under bag semantics, can be evaluated in DLOGSPACE in data complexity. For hardness, the same proofs as in [Abiteboul *et al.*, 1991] apply, since RA under set semantics can be simulated.

4 Bag Semantics: Positive Results

While Proposition 1 tells us that computing $\min_Q(D, \bar{c})$ and $\max_Q(D, \bar{c})$ may require superpolynomial time (assuming that NP $\neq$ PTIME), under set semantics these are computable efficiently for positive queries Q in RA$^+$. We now see what happens under bag semantics. In this section we show the positive result: $\min_Q(D, \bar{c})$ can be computed efficiently.

Theorem 1. *Under bag semantics, for each fixed RA$^+$ query Q, computing $\min_Q(D, \bar{c})$ can be done in* DLOGSPACE *with respect to data complexity.*

Proof sketch. The proof is based on the following observation. Let $Q(D)$ mean the naive evaluation of Q on D (under bag semantics). That is, selection conditions evaluate as follows: if $\mathsf{Null}(D) = \{\bot_1, \ldots, \bot_k\}$, then $\bot_i = \bot_j$ is true iff $i = j$, and $\bot_i \neq c$ for every $c \in \mathsf{Const}$.

We call a valuation v on D canonical if $v(\bot_i) = c_i$, for $1 \leq i \leq k$, where $c_1, \ldots, c_k$ are distinct constants and none of them belongs to $\mathsf{Const}(D)$. Then we prove the following by induction on RA$^+$ queries Q. For every tuple $\bar{a} \in Q(D)$:

1. $\#(\bar{a}, Q(D)) \leq \#(v(\bar{a}), Q(v(D)))$ for every valuation v on D;
2. $\#(\bar{a}, Q(D)) = \#(v_0(\bar{a}), Q(v_0(D)))$ for every canonical valuation v_0.

For a constant tuple $\bar{c}$ this implies that $\min_Q(\bar{c}, D) = \#(\bar{c}, Q(D))$, and the latter can be computed in DLOGSPACE due to known bounds on the complexity of RA under bag semantics [Grumbach and Milo, 1996; Libkin and Wong, 1997].

Before proving 1 and 2 above, first observe that if v is a canonical valuation on D, then for every two tuples $\bar{a}$ and $\bar{b}$ over $\mathsf{Const}(D) \cup \mathsf{Null}(D)$, the condition $v(\bar{a}) = v(\bar{b})$ implies $\bar{a} = \bar{b}$. Now proceed with the proof by induction on expressions. The base case is when Q is R, a database relation. If $\bar{a} \in R$, then $v(\bar{a}) \in v(R)$ and thus 1 holds. If v_0 is canonical and $v_0(\bar{a}) \in v_0(R)$, then $v_0(\bar{a}) = v_0(\bar{b})$ for some $\bar{b} \in R$, which implies $\bar{a} = \bar{b}$ and thus $\#(\bar{a}, R) = \#(v_0(\bar{a}), v_0(R))$, proving the base case.

For the induction case we need to look at queries obtained by applying operations $\sigma, \pi, \times, \cup, \cap$, and ε. As an example, we consider the case of selection. Assume $Q' = \sigma_{i=j}(Q)$. If $0 < \#(\bar{a}, Q'(D)) = m$, then $\#(\bar{a}, Q(D)) = m$ and $a_i = a_j$. By the hypothesis, $\#(v(\bar{a}), Q(v(D))) \geq m$ for every v, and since $v(a_i) = v(a_j)$ we have $\#(v(\bar{a}), Q'(v(D))) \geq m$, proving 1. Next assume that v is canonical and $m = \#(\bar{a}, Q'(D))$. If $a_i \neq a_j$, then $m = 0$, but since v is canonical, $v(a_i) \neq v(a_j)$ and thus $\#(v(\bar{a}), Q'(v(D))) = 0$ as well. So assume $a_i = a_j$. Then $m = \#(\bar{a}, Q(D))$ and by the induction hypothesis $m = \#(v(\bar{a}), v(Q(D))) = \#(v(\bar{a}), v(Q'(D)))$ since $v(\bar{a})$ satisfies the selection condition. This proves 2.

The remaining cases of $\pi, \times, \cup, \cap$, and ε are similarly obtained by induction. □

Theorem 1 also provides information about tuples containing nulls. It is often important to keep them in the answers, as they provide valuable information. For instance, if we have a relation $R = \{(1, \bot)\}$ and a query Q returning R itself, then there are no constant tuples with $\min_Q(R, \bar{c}) > 0$; this however loses certain information that R contains a tuple whose first component is 1. An alternative to overcome this, known as certain answers with nulls [Lipski, 1984], suggests, in the set case, looking for tuples $\bar{a}$ such that $v(\bar{a}) \in Q(v(D))$ for every valuation v. For constant tuples this is the same as saying $\min_Q(D, \bar{a}) = 1$.

For RA$^+$ queries under set semantics, it is well known [Lipski, 1984; Libkin, 2016] that $\bar{a} \in Q(D)$ if and only if $\#(v(\bar{a}), Q(v(D))) = 1$ for every valuation v; or, equivalently, $\#(\bar{a}, Q(D)) = \min_v \#(v(\bar{a}), Q(v(D)))$. We now see that the situation is identical for bag semantics.

Corollary 1. *For every RA$^+$ query Q interpreted under bag semantics, every database D and an arbitrary tuple $\bar{a}$, we have $\#(\bar{a}, Q(D)) = \min_v \#(v(\bar{a}), Q(v(D)))$.*

5 Bag Semantics: Negative Results

We saw that, under bag semantics, $\min_Q(D, \bar{c})$ can be computed efficiently. We now show that the situation is drastically different for max: computing $\max_Q(D, \bar{c})$ is intractable, in data complexity, even for very simple queries that just return a relation from the database (that is, SQL queries of the form `SELECT * FROM R`).

To state the result, we need to cast this problem as a decision problem. For this, we define the problem LEASTOCCUR that takes as input a k-ary relation R over $\mathsf{Const} \cup \mathsf{Null}$, a k-ary tuple $\bar{c}$ over Const, and an integer $m > 0$, and asks whether there exists a valuation v on $\mathsf{Null}(R)$ such that $\bar{c}$ occurs at least m times in $v(R)$, i.e., $\#(\bar{c}, v(R)) \geq m$. Clearly the problem is in NP: it suffices to guess a valuation that only uses elements of $\mathsf{Const}(R)$ and $\bar{c}$. The problem is also NP-hard.

Theorem 2. *The problem* LEASTOCCUR *is* NP*-complete.*

Corollary 2. *There is an RA$^+$ query Q (in fact a query returning a relation in the database) such that checking, for given $D, \bar{c}$, and m whether $\max_Q(D, \bar{c}) < m$ is* coNP*-complete.*

Proof sketch. To show that LEASTOCCUR is NP-hard we reduce to it from *Maximum 2-Satisfiability* (MAX2SAT), which is NP-complete [Garey and Johnson, 1979]. Given a set Σ of propositional clauses, each consisting of exactly two literals, and a positive integer $m \leq |\Sigma|$, MAX2SAT is the problem of deciding whether there exists a truth assignment that satisfies at least m clauses in Σ.

Let Σ be a set of clauses of two literals. With each propositional variable p occurring in Σ we associate a distinct null value $\bot_p$ and the following two pairs: $\bar{u}_p^{\mathbf{t}} = (0, \bot_p)$ and $\bar{u}_p^{\mathbf{f}} = (\bot_p, 1)$. Observe that $(0, 1)$ unifies with each of these pairs, but there is no valuation v for which $v(\bar{u}_p^{\mathbf{t}}) = v(\bar{u}_p^{\mathbf{f}})$.

With each clause $\sigma \in \Sigma$ we then associate a quaternary relation R_σ defined as follows:

$$R_\sigma = \{\, \bar{u}_p^{\mathbf{t}}\bar{u}_q^{\mathbf{t}}\,,\ \bar{u}_p^{\mathbf{t}}\bar{u}_q^{\mathbf{f}}\,,\ \bar{u}_p^{\mathbf{f}}\bar{u}_q^{\mathbf{t}} \,\} \qquad \text{if } \sigma = \{\ \ p,\ \ q\}\,;$$
$$R_\sigma = \{\, \bar{u}_p^{\mathbf{t}}\bar{u}_q^{\mathbf{f}}\,,\ \bar{u}_p^{\mathbf{t}}\bar{u}_q^{\mathbf{t}}\,,\ \bar{u}_p^{\mathbf{f}}\bar{u}_q^{\mathbf{f}} \,\} \qquad \text{if } \sigma = \{\ \ p, \neg q\}\,;$$
$$R_\sigma = \{\, \bar{u}_p^{\mathbf{f}}\bar{u}_q^{\mathbf{f}}\,,\ \bar{u}_p^{\mathbf{f}}\bar{u}_q^{\mathbf{t}}\,,\ \bar{u}_p^{\mathbf{t}}\bar{u}_q^{\mathbf{f}} \,\} \qquad \text{if } \sigma = \{\neg p, \neg q\}\,;$$

where p and q range over the propositional variables mentioned in Σ. Intuitively, the tuples in R_σ represent the truth assignments that satisfy σ. Note that $(0,1,0,1)$ unifies with each of the tuples in R_σ, but there is no valuation v for which it would occur more than once in $v(R_\sigma)$. This is an immediate consequence of the fact that $v(\bar{u}_p^{\mathbf{t}}) \neq v(\bar{u}_p^{\mathbf{f}})$ for every propositional variable p and every valuation v.

Now, let $R_\Sigma = \bigcup_{\sigma\in\Sigma} R_\sigma$. Clearly, by construction, there is a bijective correspondence between the nulls in R_Σ and the propositional variables in Σ. Thus, for each truth assignment α we can define a valuation v_α with $v_\alpha(\bot_p) = 1$ if $\alpha(p) = \mathbf{t}$, and $v_\alpha(\bot_p) = 0$ otherwise; and for each valuation v we can define a truth assignment α_v with $\alpha_v(p) = \mathbf{t}$ if $v(\bot_p) = 1$, and $\alpha_v(p) = \mathbf{f}$ otherwise.

Claim 1. *Let Σ be a set of clauses of two literals, let α be a truth assignment to the propositional variables of Σ, and let $\sigma \in \Sigma$. Then, α satisfies σ if and only if $(0,1,0,1)$ occurs exactly once in $v_\alpha(R_\sigma)$.*

We illustrate one case. Let $\bar{c} = (0,1,0,1)$, and let p and q be the propositional variables occurring in σ. Assume p occurs positively and q occurs negatively. Then,

$$R_\sigma = \{(0,\bot_p,\bot_q,1),(0,\bot_p,0,\bot_q),(\bot_p,1,\bot_q,1)\}$$

$\alpha \models \sigma$ iff exactly one of the following holds:

$$\alpha(p) = \mathbf{t} \ \text{ and } \ \alpha(q) = \mathbf{f}\ ;$$
$$\alpha(p) = \mathbf{t} \ \text{ and } \ \alpha(q) = \mathbf{t}\ ;$$
$$\alpha(p) = \mathbf{f} \ \text{ and } \ \alpha(q) = \mathbf{f}\ .$$

$\#\big(\bar{c}, v_\alpha(R_\sigma)\big) = 1$ iff exactly one of the following holds:

$$v_\alpha(\bot_p) = 1 \ \text{ and } \ v_\alpha(\bot_q) = 0\ ;$$
$$v_\alpha(\bot_p) = 1 \ \text{ and } \ v_\alpha(\bot_q) = 1\ ;$$
$$v_\alpha(\bot_p) = 0 \ \text{ and } \ v_\alpha(\bot_q) = 0\ .$$

The claim then follows by the definition of v_α. The remaining cases, when both variables occur positively, or both occur negatively, are analogous.

Similarly, but using the definition of α_v, we also obtain:

Claim 2. *Let Σ be a set of clauses of two literals, let v be a valuation of the nulls in R_Σ, and let $\sigma \in \Sigma$. Then, $(0,1,0,1)$ occurs exactly once in $v(R_\sigma)$ if and only if α_v satisfies σ.*

The above claims now apply to show the following:

Claim 3. *Let (Σ, m) be instance of* MAX2SAT. *There exists a truth assignment α that satisfies at least m clauses in Σ if and only if there exists a valuation v such that $(0,1,0,1)$ occurs at least m times in $v(R_\Sigma)$.*

Given Σ, the construction of R_Σ is polytime and gives the desired reduction from MAX2SAT to LEASTOCCUR. □

In some restricted cases $\max_Q(D,\bar{c})$ can be calculated efficiently. Consider, for example, queries of the form $\varepsilon(Q)$, with duplicate elimination applied as the outermost operation. They correspond to SQL queries `SELECT DISTINCT ...`, where the remainder of the query is arbitrary and is interpreted under bag semantics.

Proposition 2. *For RA$^+$ queries of the form $\varepsilon(Q)$, where Q is interpreted under bag semantics, $\max_Q(D,\bar{c})$ can be computed in polynomial time with respect to data complexity.*

Proof sketch. For a query Q, let $Q^b(D)$ and $Q^s(D)$ denote their outputs under bag and set semantics on complete databases D; also $\varepsilon(D)$ refers to D in which duplicate elimination was applied to every relation. We then show by induction that for queries Q in RA$^+$, we have $\varepsilon(Q^b(D)) = Q^s(\varepsilon(D))$ for every D (of course ε is the identity on set-based databases). Then $\max_{\varepsilon(Q^b)}(D,\bar{c}) = \max_{Q^s}(\varepsilon(D),\bar{c})$ and the latter can be computed in polynomial time by [Abiteboul *et al.*, 1991]. □

6 Approximations under Bag Semantics

Our results shed some light on *approximating* answers to queries over incomplete databases: we can explain why some approximation schemes work in real-life DBMSs while others do not. As is well known (see Fact 1), computing certain and possible answers to relational algebra (and thus first-order) queries over incomplete databases is an intractable problem. It is thus natural to try to approximate such answers by tractable queries. The idea is not new: first solutions were proposed long ago [Reiter, 1986; Vardi, 1986] but in the context of databases viewed as logical theories which made them unimplementable in real DBMSs.

The first approximation scheme designed to work with the standard relational database technology appeared in [Libkin, 2016]. It still used set semantics; its idea was to modify an RA query Q into a query $Q^{\mathbf{t}}$ that returns a subset of certain answers. More precisely, for a tuple $\bar{a}$ that can contain constants and nulls, define

$$\begin{array}{rcl} \min_Q(D,\bar{a}) & = & \min_v \#(v(\bar{a}), Q(v(D))) \\ \max_Q(D,\bar{a}) & = & \max_v \#(v(\bar{a}), Q(v(D))) \end{array} \tag{4}$$

where v ranges over valuations. Note that if $\bar{a}$ contains only constants, this is the same as (1), and for arbitrary tuples, $\min_Q(D,\bar{a})$ is efficiently computable for RA$^+$ queries, by Corollary 1.

The scheme showed how to transform Q into queries $Q^{\mathbf{t}}, Q^{\mathbf{f}}$ such that

$$\begin{array}{rcl} Q^{\mathbf{t}}(D) & \subseteq & \{\bar{a} \mid \min_Q(D,\bar{a}) = 1\} \\ Q^{\mathbf{f}}(D) & \subseteq & \{\bar{a} \mid \min_{\bar{Q}}(D,\bar{a}) = 1\} \end{array} \tag{5}$$

where $\bar{Q}$ computes the complement of Q. These queries, giving us certainly true and certainly false answers, were defined by mutual recursion (i.e., $Q^{\mathbf{f}}$ was necessary to define $Q^{\mathbf{t}}$).

How does one extend such definitions to bags? To see this, we restate (5) as follows. Let $\oplus$ be the addition in the two-element field $\mathbb{F}_2$.

Proposition 3. *Under set semantics of queries, conditions (5) are equivalent to*

$$\begin{array}{rcl} \#(\bar{a}, Q^{\mathbf{t}}(D)) & \leq & \min_Q(D, \bar{a}) \\ \#(\bar{a}, Q^{\mathbf{f}}(D)) & \leq & 1 \oplus \max_Q(D, \bar{a}) \end{array} \tag{6}$$

for every tuple $\bar{a}$.

This suggests a natural extension of the translation scheme $(Q^{\mathbf{t}}, Q^{\mathbf{f}})$ to bags: one uses (6) but replaces $\oplus$ with the usual addition, as now occurrences can be arbitrary natural numbers. But this is suddenly very problematic, as $\max_Q(D, \bar{a})$ is hard computationally, for *all* queries, since we cannot even compute it efficiently for base relations! Thus, implementing this approximation scheme in a real-life RDBMS (which is bag-based) is infeasible.

It was observed in [Guagliardo and Libkin, 2016] that the translation $Q \mapsto (Q^{\mathbf{t}}, Q^{\mathbf{f}})$, while enjoying good theoretical complexity, is unlikely to work well in a real-world setting because rewritten queries require very large joins even in the set case. The results of the previous section show that even good theoretical complexity is lost if one uses bag semantics.

Based on the size of the rewritten queries that the translation $Q \mapsto (Q^{\mathbf{t}}, Q^{\mathbf{f}})$ produces, [Guagliardo and Libkin, 2016] proposed a different translation, again for the set case. It transforms a query Q into two queries Q^+ and $Q^?$ such that

$$v(Q^+(D)) \subseteq Q(v(D)) \subseteq v(Q^?(D)) \tag{7}$$

for all valuations v. This, over sets, implies that $Q^+(D) \subseteq \{\bar{a} \mid \min_Q(D, \bar{a}) = 1\} \subseteq Q^?(D)$. In other words, we have lower and upper approximations of certain answers.

Again, all the development of [Guagliardo and Libkin, 2016] was done under the set semantics of queries, but (7) makes it easy to extend the framework to bags. We take (7) to be the requirement for queries Q^+ and $Q^?$, where $\subseteq$ now has bag-theoretic meaning: for two bags B_1, B_2, we write $B_1 \subseteq B_2$ iff $\#(b, B_1) \leq \#(b, B_2)$ for every element b.

Proposition 4. *Assume that for an RA query Q, we have two queries Q^+ and $Q^?$ that satisfy (7), under bag semantics. Then $\#(\bar{a}, Q^+(D)) \leq \min_Q(\bar{a}, D) \leq \#(\bar{a}, Q^?(D))$, for every database D and every tuple $\bar{a}$ of elements of D.*

In fact Proposition 4 is true for any query that is invariant under isomorphisms, which includes queries expressible in logics such as first-order and its extensions. It recasts the approximation approach of [Guagliardo and Libkin, 2016] in a way that makes no reference to $\max_Q$. This gives a strong indication that it can be adapted to bag semantics.

This is indeed so. To show this, we use the translation $Q \mapsto (Q^+, Q^?)$ below:

$$\begin{array}{cc} R^+ = R & R^? = R \\ (Q_1 \times Q_2)^+ = Q_1^+ \times Q_2^+ & (Q_1 \times Q_2)^? = Q_1^? \times Q_2^? \\ (Q_1 \cup Q_2)^+ = Q_1^+ \cup Q_2^+ & (Q_1 \cup Q_2)^? = Q_1^? \cup Q_2^? \\ (Q_1 \cap Q_2)^+ = Q_1^+ \cap Q_2^+ & (Q_1 \cap Q_2)^? = Q_1^? \ltimes_{\Uparrow} Q_2^? \\ (Q_1 - Q_2)^+ = Q_1^+ - Q_1^+ \ltimes_{\Uparrow} Q_2^? & (Q_1 - Q_2)^? = Q_1^? - Q_2^+ \\ (\sigma_{i=j}(Q))^+ = \sigma_{i=j}(Q^+) & (\sigma_{i=j}(Q))^? = \sigma_{i\sim j}(Q^?) \\ (\pi_{\bar{i}}(Q))^+ = \pi_{\bar{i}}(Q^+) & (\pi_{\bar{i}}(Q))^? = \pi_{\bar{i}}(Q^?) \\ (\varepsilon(Q))^+ = \varepsilon(Q^+) & (\varepsilon(Q))^? = \varepsilon(Q^?) \end{array}$$

The selection condition $i \sim j$ used in $(\sigma_{i=j}(Q))^?$ is a disjunction $(i = j) \vee \mathsf{null}(i) \vee \mathsf{null}(j)$, checking that either the ith and the jth component of a tuple are the same, or one of them belongs to Null. The translations $(Q_1 \cap Q_2)^?$ and $(Q_1 - Q_2)^+$ use the semijoin operation $\ltimes_{\Uparrow}$ defined as follows. We write $\bar{a} \Uparrow \bar{b}$ if there exists a valuation v such that $v(\bar{a}) = v(\bar{b})$. Then, for two relations of the same arity, $R \ltimes_{\Uparrow} S$ contains tuples $\bar{a}$ in R such that $\bar{a} \Uparrow \bar{b}$ for some $\bar{b} \in S$, and $\#(\bar{a}, R \ltimes_{\Uparrow} S) = \#(\bar{a}, R)$.

This is essentially the same translation of [Guagliardo and Libkin, 2016] but now all operations, including semijoin, are interpreted under bag semantics. It turns out that it provides an approximation of certain answers in the following sense.

Theorem 3. *The translation $Q \mapsto (Q^+, Q^?)$ satisfies (7) when queries are interpreted under bag semantics. Furthermore, queries Q^+ and $Q^?$ can be evaluated in* DLOGSPACE *with respect to data complexity.*

It was previously empirically observed that the translation $Q \mapsto (Q^+, Q^?)$ is efficient on real-life benchmark queries, while $Q \mapsto (Q^{\mathbf{t}}, Q^{\mathbf{f}})$ is not. Now using our complexity results for answering queries on incomplete databases under bag semantics we provided a theoretical justification for this observation, based on the fact that one scheme can be expressed in terms of $\min_Q$ alone, while the other needs $\max_Q$, which cannot even be computed efficiently for base relations.

7 Conclusions

Much of the theoretical work on query answering in databases, including handling incomplete information, assumes set-based semantics [Abiteboul *et al.*, 1995], leading to a mismatch with what happens in real-life DBMSs [Date and Darwen, 1996]. This also applies to many scenarios that reduce problems combining reasoning and data to answering relational database queries.

In this paper we showed that even for the well-behaved class of positive relational algebra queries (or unions of conjunctive queries), bag semantics complicates query answering with incomplete information considerably, as some of the problems easily solvable under set semantics become intractable. This is not just bad news however: this observation provided a theoretical justification for an approximation of query answers on incomplete databases that was known to behave well in practice.

Our results open up a new line of work on understanding incompleteness in real-life databases, and on using it in applications that combine data with knowledge and meta-information such as ontologies or schema mappings. One needs to devise and optimize approximation schemes, look into interaction with constraints [Calì *et al.*, 2003a], extend results to open-world assumption, and look into new applications such as querying inconsistent data, where reliance on database technology [Bertossi, 2011] and the use of approximations [Bienvenu and Rosati, 2013] is common.

Acknowledgments

We thank the anonymous referees for their comments. Work supported by EPSRC grants N023056 and M025268.

References

[Abiteboul *et al.*, 1991] Serge Abiteboul, Paris Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, 78(1):158–187, 1991.

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Arenas *et al.*, 2014] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014.

[Bertossi, 2011] Leopoldo Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.

[Bienvenu and Rosati, 2013] Meghyn Bienvenu and Riccardo Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI*, pages 775–781, 2013.

[Calì *et al.*, 2003a] Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*, pages 260–271, 2003.

[Calì *et al.*, 2003b] Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI*, pages 16–21, 2003.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[Chandra and Merlin, 1977] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[Chaudhuri and Vardi, 1993] Surajit Chaudhuri and Moshe Y. Vardi. Optimization of *real* conjunctive queries. In *PODS*, pages 59–70, 1993.

[Date and Darwen, 1996] C. J. Date and Hugh Darwen. *A Guide to the SQL Standard*. Addison-Wesley, 1996.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman, 1979.

[Gottlob *et al.*, 2014] Georg Gottlob, Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, Thomas Schwentick, and Michael Zakharyaschev. The price of query rewriting in ontology-based data access. *Artificial Intelligence*, 213:42–59, 2014.

[Grumbach and Milo, 1996] Stéphane Grumbach and Tova Milo. Towards tractable algebras for bags. *JCSS*, 52(3):570–588, 1996.

[Guagliardo and Libkin, 2016] Paolo Guagliardo and Leonid Libkin. Making SQL queries correct on incomplete databases: A feasibility study. In *PODS*, pages 211–223, 2016.

[Imielinski and Lipski, 1984] Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

[Jayram *et al.*, 2006] T. S. Jayram, Phokion G. Kolaitis, and Erik Vee. The containment problem for real conjunctive queries with inequalities. In *PODS*, pages 80–89, 2006.

[Kolaitis, 2003] Phokion G. Kolaitis. Constraint satisfaction, databases, and logic. In *IJCAI*, pages 1587–1595, 2003.

[Kontchakov *et al.*, 2011] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *IJCAI*, pages 2656–2661, 2011.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: a theoretical perspective. In *PODS*, pages 233–246, 2002.

[Libkin and Wong, 1997] Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *JCSS*, 55(2):241–272, 1997.

[Libkin, 2016] Leonid Libkin. SQL's three-valued logic and certain answers. *ACM Trans. Database Syst.*, 41(1):1:1–1:28, 2016.

[Lipski, 1984] Witold Lipski. On relational algebra with marked nulls. In *PODS*, pages 201–203, 1984.

[Lutz *et al.*, 2015] Carsten Lutz, Inanç Seylan, and Frank Wolter. Ontology-mediated queries with closed predicates. In *IJCAI*, pages 3120–3126, 2015.

[Reiter, 1977] Raymond Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.

[Reiter, 1986] Raymond Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM*, 33(2):349–347, 1986.

[Vardi, 1982] Moshe Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing, STOC'82*, pages 137–146, 1982.

[Vardi, 1986] Moshe Y. Vardi. Querying logical databases. *JCSS*, 33(2):142–160, 1986.