Best Answers over Incomplete Data : Complexity and First-Order Rewritings

Amélie Gheerbrant and **Cristina Sirangelo**
Université de Paris, IRIF, CNRS, F-75013 Paris, France
{amelie, cristina}@irif.fr

Abstract

Answering queries over incomplete data is ubiquitous in data management and in many AI applications that use query rewriting to take advantage of relational database technology. In these scenarios one lacks full information on the data but queries still need to be answered with certainty. The certainty aspect often makes query answering unfeasible except for restricted classes, such as unions of conjunctive queries. In addition often there are no, or very few, certain answers, thus expensive computation is in vain. Therefore we study a relaxation of certain answers called best answers. They are defined as those answers for which there is no better one (that is, no answer true in more possible worlds). When certain answers exist the two notions coincide. We compare different ways of casting query answering as a decision problem and characterise its complexity for first-order queries, showing significant differences in the behaviour of best and certain answers. We then restrict attention to best answers for unions of conjunctive queries and produce a practical algorithm for finding them based on query rewriting techniques.

1 Introduction

Answering queries over incomplete databases is crucial in many different scenarios such as data integration [Lenzerini, 2002], data exchange [Arenas *et al.*, 2014], inconsistency management [Bertossi, 2011], data cleaning [Geerts *et al.*, 2013], ontology-based data access (OBDA) [Bienvenu and Ortiz, 2015], and many others. The common thread running through all these applications lies in computing *certain answers* [Amendola and Libkin, 2018; Libkin, 2018a], which is the standard way of answering queries over incomplete databases. Intuitively this produces answers that can be obtained from all the possible complete databases a given incomplete database represents. However, computing such query answers then relies on sophisticated algorithms that are often difficult to implement. It is well known that restricting to unions of conjunctive queries allows to overcome the difficulty by using *naïve evaluation* [Imieliński and Lipski, 1984]. This amounts to evaluating queries over incomplete databases as if nulls were usual data values, thus merely using the standard database query engine to compute certain answers.

In general though it is a common occurrence that few if any certain answers can be found. If there are no certain answers, it is still useful to provide a user with some answers, with suitable guarantees. To address this need, a framework to measure how close an answer is to certainty has recently been proposed [Libkin, 2018b], setting the foundations to both a quantitative and a qualitative approach. We focus on the qualitative notion of *best answers*. Those are a refinement of certain answers based on comparing query answers; one that is supported by a larger set of complete interpretations is better. Best answers are those answers for which there is no better one. They always exist and when certain answers exist the two notions simply coincide.

Best answers is a natural notion, but we still know little about it. Identifying the set of best answers among some given family of sets of answers is known to be complete in data complexity for the class $\text{P}^{\text{NP}[\log n]}$, which is considered as "mildly" harder than both NP and coNP. However this very formulation of the decision problem is non standard. Traditionally, in databases we rather focus on problems stating that some given result belongs to the set of answers, or that some given set is the set of answers. Certain answers as a decision problem is typically formulated in the first way. So do these variations matter ? We fully answer the question for both best and certain answers of first-order queries, showing significant differences in their computational behaviour.

Despite the high complexity of finding best answers in general, one gains tractability when restricting to unions of conjunctive queries. This is a common class of queries, usually well behaved computationally, even for certain answers. Finding best answers for them was shown to be tractable in [Libkin, 2018b] via an adaptation of techniques used in the context of certain answers [Gheerbrant and Libkin, 2015]. Those are essentially resolution based algorithms for first-order formulas; this makes them hard to implement in the database context. To overcome this we develop new query rewriting techniques. In particular we show that best answers to any union of conjunctive queries can be computed by issuing a new first-order query directly on the incomplete database. Query rewritings are standard in the context of, e.g., consistent query answering, OBDA, query answering using views etc., i.e., all contexts where only partial information is available about the data to be queried [Calvanese *et al.*, 2000; Calvanese *et al.*, 2007; Calì *et al.*, 2013; Calì *et al.*, 2003b]. First-order rewritings are particularly useful, as they allow to use the power of standard database query engines. In fact when they exist, the rewritten queries can be implemented in any relational query engine by expressing them in SQL, with no need to implement ad-hoc algorithms.

2 Preliminaries

We represent missing information in relational databases in the standard way using nulls [Abiteboul *et al.*, 1995; Imieliński and Lipski, 1984; van der Meyden, 1998]. Databases are populated by *constants* and *nulls*, coming respectively from two countably infinite sets Const and Null. We denote nulls by $\perp$, sometimes with sub- or superscript. We also allow them to repeat, thus adopting the model of *marked* nulls, as customary in the context of applications such as OBDA or data integration and exchange. A relational schema, or vocabulary σ, is a set of relation names with associated arities. A database D over σ associates to each relation name of arity k in σ, a k-ary relation which is a finite subset of $(\mathsf{Const} \cup \mathsf{Null})^k$. Sets of constants and nulls occuring in D are denoted by $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$. The *active domain* of D is $\text{adom}(D) = \mathsf{Const}(D) \cup \mathsf{Null}(D)$. A *complete* database has no nulls.

A *valuation* $v : \mathsf{Null}(D) \to \mathsf{Const}$ on a database D is a map that assigns constant values to nulls occurring in D. By $v(D)$ [resp. $v(\bar{a})$] we denote the result of replacing each null $\perp$ by $v(\perp)$ in D [resp. in the tuple $\bar{a}$]. The semantics $[\![D]\!]$ of an incomplete database D is the set $\{v(D) \mid v \text{ is a valuation on } D\}$ of all complete databases it can represent. $\mathsf{V}(D)$ denotes the set of all valuations defined on D.

An m-ary *query* of active domain $C \subseteq \mathsf{Const}$ is a map that associates with a database D a subset of $(\text{adom}(D) \cup C)^m$. The active domain of a query will be denoted as $\text{adom}(Q)$. To answer a query Q over an incomplete database D we follow [Lipski, 1984; Libkin, 2018b] and adopt a slight generalisation of the usual intersection based certain answers notion. We define the set of *certain answers* to Q over D as $\Box(Q, D) = \{\bar{a} \text{ over adom}(D) \cup \text{adom}(Q) \mid \forall v : v(\bar{a}) \in Q(v(D))\}$. The only difference with the usual notion is that we allow answers to contain nulls.

Following [Libkin, 2018b], given a query Q, a database D, and a tuple $\bar{a}$ over $\text{adom}(D) \cup \text{adom}(Q)$, we let the *support* of $\bar{a}$ be the set of all valuations that witness it:

$$\mathsf{Supp}(Q, D, \bar{a}) = \{v \in \mathsf{V}(D) \mid v(\bar{a}) \in Q(v(D))\}.$$

Supports thus measure how close a tuple is to certainty. We consider one answer to be *better* than another if it has more support. That is, given a database D, a k-ary query Q, and k-tuples $\bar{a}, \bar{b}$ over $\text{adom}(D) \cup \text{adom}(Q)$, we let

$$\bar{a} \lhd_{Q,D} \bar{b} \quad \Leftrightarrow \quad \mathsf{Supp}(Q, D, \bar{a}) \subset \mathsf{Supp}(Q, D, \bar{b}).$$

The set of *best answers* to Q over D is defined as the set of answers for which there is no better one : $\mathsf{Best}(Q, D) = \{\bar{a} \mid \neg\exists\bar{b} : \bar{a} \lhd_{Q,D} \bar{b}\}$.

We focus on *first-order* (FO) queries of vocabulary σ written here in the logical notation using Boolean connectives $\wedge, \vee, \neg$ and quantifiers $\exists, \forall$. We write $\varphi(\bar{x})$ for an FO-formula φ with free variables $\bar{x}$. With slight abuse of notation, $\bar{x}$ will denote both a tuple of variables and the set of variables occurring in it. The set of constants used by φ is as usual denoted by $\text{adom}(\varphi)$, and gives the active domain of the associated query. We interpret FO-formulas under active domain semantics, i.e. we consider D as a relational structure with universe $\text{adom}(D) \cup \text{adom}(\varphi)$. Thus an FO formula $\varphi(\bar{x})$ represents a query (of active domain $\text{adom}(\varphi)$) mapping each database D into the set of tuples $\{\bar{t} \text{ over adom}(D) \cup \text{adom}(\varphi) \mid D \models \varphi(\bar{t})\}$.

To evaluate FO-formulas with free variables we use assignments ν from variables to constants in the active domain. Note that with a little abuse of notation we write $D \models \varphi(\bar{t})$ for $D \models_\nu \varphi(\bar{x})$ under the assignment ν sending $\bar{x}$ to $\bar{t}$.

Here it is important to note that the query associated to φ is a mapping defined on all databases D, possibly with nulls. If D contains nulls, $D \models \varphi(\bar{t})$ is to be intended "naïvely", i.e. nulls of D are treated as new constants in the domain of D, distinct from each other, and distinct from all the other constants in D and φ. For example the query $\varphi(x, y) = \exists z\ (R(x, z) \wedge R(z, y))$, on the database $D = \{R(1, \perp_1), R(\perp_1, \perp_2), R(\perp_3, 2)\}$ selects only the tuple $(1, \perp_2)$.

We consider the $\exists, \wedge, \vee$-fragment of FO known as *unions of conjunctive queries* and its $\exists, \wedge$-fragment known as *conjunctive queries*.

Example 2.1. *Let $Q(x) = \exists y(R(y) \wedge S(y, x))$ and $D = \{R(\perp_1), R(1), S(\perp_2, \perp_2)\}$. We have $\mathsf{Supp}(Q, D, \perp_2) = \{v \in \mathsf{V}(D) \mid v(\perp_2) = 1 \text{ or } v(\perp_1) = v(\perp_2)\}$, $\mathsf{Supp}(Q, D, 1) = \{v \in \mathsf{V}(D) \mid v(\perp_2) = 1\}$ and $\mathsf{Supp}(Q, D, \perp_1) = \{v \in \mathsf{V}(D) \mid v(\perp_1) = v(\perp_2)\}$. It follows that $\Box(Q, D) = \emptyset$ and $\mathsf{Best}(Q, D) = \{(\perp_2)\}$.*

In order to study the complexity of best answer computation we shall need two classes in the second level of the polynomial hierarchy. Both of these contain NP and coNP, and are contained in $\Sigma_2^p \cap \Pi_2^p$. The class DP consists of languages $L_1 \cap L_2$ where $L_1 \in$ NP and $L_2 \in$ coNP. This class has appeared in database applications [Fagin *et al.*, 2005; Barceló *et al.*, 2014]. The class $\text{P}^{\text{NP}[\log n]}$ consists of problems that can be solved in polynomial time with a logarithmic number of calls to an NP oracle [Eiter and Gottlob, 1997]. Equivalently, it can be described as the class of problems solved in P with an NP oracle where calls to the oracle are done in parallel, i.e., independent of each other. This class has appeared in the context of AI, modal logic, OBDA [Gottlob, 1995; Eiter and Gottlob, 1997; Calvanese *et al.*, 2006; Bienvenu and Bourgaux, 2016], data exchange [Arenas *et al.*, 2013].

3 Complexity of Best and Certain Answers

A natural way to cast computing best answers as a decision problem would be to proceed along the lines of certain answers [Imieliński and Lipski, 1984] and ask whether a given tuple is a best answer :

PROBLEM:	BESTANSWER
INPUT:	A query Q, a database D, a tuple $\bar{a}$
QUESTION:	Is $\bar{a} \in \mathsf{Best}(Q, D)$?

Instead, [Libkin, 2018b] considers the following variant of the problem, asking whether the set of best answers belongs to a specified family of sets :

PROBLEM:	BESTANSWER$^\in$
INPUT:	A query Q, a database D, a family $\mathcal{X}$ of sets of tuples
QUESTION:	Is $\mathsf{Best}(Q, D) \in \mathcal{X}$?

This suggests yet another alternative formulation of the problem, asking if a given set is the best answer :

PROBLEM:	BESTANSWER$^=$
INPUT:	A query Q, a database D, a set X of tuples,
QUESTION:	Is $\mathsf{Best}(Q, D) = X$?

BESTANSWER$^\in$ was shown in [Libkin, 2018b] to be $\mathrm{P}^{\mathrm{NP}[\log n]}$-complete in data complexity. We show that the other alternatives are computationally equivalent. Interestingly, the situation is very different with certain answers, as we show next.

Theorem 3.1. *For* FO *queries the problems* BESTANSWER *and* BESTANSWER$^=$ *are* $\mathrm{P}^{\mathrm{NP}[\log n]}$*-complete in data complexity.*

Proof (sketch). The upper bound for BESTANSWER$^=$ immediately follows from the upper bound for BESTANSWER$^\in$ (take the family $\mathcal{X}$ to be a singleton $\{X\}$). As for BESTANSWER we only need a slight modification of the upper bound proof in [Libkin, 2018b]. To check whether $\bar{a} \in \mathsf{Best}(Q, D)$ we proceed as follows. Since the query is fixed, and has therefore fixed arity k, in polynomial time we can enumerate all the k-tuples of $\mathrm{adom}(D)$. Then, using parallel calls to the NP oracle, we can check for each such tuple $\bar{b}$ whether $\mathsf{Supp}(Q, D, \bar{a}) \subseteq \mathsf{Supp}(Q, D, \bar{b})$ and whether $\mathsf{Supp}(Q, D, \bar{b}) \subseteq \mathsf{Supp}(Q, D, \bar{a})$. With this information, in polynomial time we know whether $\bar{a} \lhd_{Q,D} \bar{b}$ for some $\bar{b}$.

We prove the two remaining lower bounds, reducing from the same $\mathrm{P}^{\mathrm{NP}[\log n]}$-complete problem [Wagner, 1990]: given an undirected graph G, is its chromatic number $\chi(G)$ odd? With each undirected graph $G = \langle N, E\rangle$ with nodes N and edges E, we associate a database D_G over binary relations L, E and unary relations C, O as follows. We use a null $\bot_n$ in D_G for each node n of G. For each edge $\{n, n'\}$ of G, we have pairs $(\bot_n, \bot_{n'})$ and $(\bot_{n'}, \bot_n)$ in the relation E of D_G. In relation C we have m constants $\{c_1, \ldots, c_m\}$ (intuitively representing possible colors), where m is the number of nodes of G. Relation O of D_G is defined as $\{c_i \mid i \text{ is odd}\}$ and L is a linear ordering on them, i.e., $(c_i, c_j) \in L$ iff $i \leq j$, for $i, j \leq m$.

Remark that any valuation v of D_G that maps each null into a constant of C represents an assignment of colours in $\{c_1, \ldots, c_m\}$ to nodes of G. Then we define a query

$$\varphi(x) \;=\; \begin{array}{ll} & C(x) \\ \wedge & \forall y, z\ \big(E(y,z) \rightarrow L(y,x)\big) \\ \wedge & \forall y\ \big(L(y,x) \rightarrow \exists z\ E(y,z)\big) \\ \wedge & \neg\exists y\ E(y,y)\,. \end{array}$$

For any valuation v, $\varphi(c)$ holds in $v(D_G)$ iff 1) $c = c_j$ for some $j = 1..m$ (ensured by the first conjunct). 2) For such a c_j, the valuation v maps each null into $\{c_1, \ldots, c_j\}$ (second conjunct), i.e. v represents an assignment of colours to nodes of G, using at most the first j colours. 3) Each color $\{c_1, \ldots, c_j\}$ is used by v, i.e. v represents an assignment of colours to nodes of G, using precisely the first j colours (third conjunct). 4) There are no loops in E (fourth conjunct).

Thus, for a valuation v, the formula $\varphi(c_j)$ is true in $v(D_G)$ iff v represents a colouring of G using precisely the first j colours $\{c_1, \ldots, c_j\}$ (which in the sequel we refer to as an *exact j-colouring* of G).

Next we define :

$$Q(x) = C(x) \wedge (\quad \begin{array}{l} \varphi(x) \vee \\ \exists y\ (O(y) \wedge L(x,y) \wedge \varphi(y)) \quad) \end{array}$$

For a valuation v, we have that $Q(c_i)$ holds in $v(D_G)$ iff either v represents an exact i-coloring of G; or v represents an exact j-coloring of G with j odd, and $i \leq j$. In other words valuations representing exact j-colorings, with j even, support only the maximal color c_j; while valuations representing exact j-colorings, with j odd, support all colors $\{c_1 \ldots c_j\}$.

With this in place we can conclude the reduction for the BESTANSWER problem :

Claim. $c_1 \in \mathsf{Best}(Q, D_G)$ iff the chromatic number of G is odd.

An adaptation of this encoding can be used to reduce the odd chromatic number problem to BESTANSWER$^=$ as well.

□

Now that we showed that all three formulations of best answers actually collapse computationally, another natural question arises. Does a similar result hold for certain answers ? We obtain the decision problems CERTAINANSWER, CERTAINANSWER$^=$ and CERTAINANSWER$^\in$ by replacing everywhere in the statements of the above decision problems BESTANSWER by CERTAINANSWER and $\mathsf{Best}(Q, D)$ by $\Box(Q, D)$. It is well known that data complexity of CERTAINANSWER is CONP-complete for FO-queries [Abiteboul *et al.*, 1991]. We complete the picture as follows.

Theorem 3.2. *For* FO *queries,* CERTAINANSWER$^=$ *is DP-complete and* CERTAINANSWER$^\in$ *is* $\mathrm{P}^{\mathrm{NP}[\log n]}$*-complete in data complexity.*

Proof. To prove membership of CERTAINANSWER$^=$ in DP, notice that for a query Q, this problem is the intersection of two languages $L_1 \cap L_2$ where $L_1 = \{(D, X) \mid X \subseteq \Box(Q, D)\}$ and $L_2 = \{(D, X) \mid \overline{X} \subseteq \overline{\Box(Q, D)}\}$. L_1 is known to be in CONP : we guess a tuple $\bar{a} \in X$ and a valuation $v \in V(D)$ with $v(\bar{a}) \notin Q(v(D))$. Similarly, L_2 is in NP : we guess a tuple $\bar{b} \in \overline{X}$ and a valuation $v' \in V(D)$ with $v'(\bar{b}) \in Q(v(D))$.

Type of problem	$\bar{a} \in$ Answer	$X =$ Answer	Answer $\in \mathcal{X}$
Certain Answer	coNP-complete [Abiteboul *et al.*, 1991]	DP complete	$\mathrm{P}^{\mathrm{NP}[\log n]}$-complete
Best Answer	$\mathrm{P}^{\mathrm{NP}[\log n]}$-complete	$\mathrm{P}^{\mathrm{NP}[\log n]}$-complete	$\mathrm{P}^{\mathrm{NP}[\log n]}$-complete [Libkin, 2018b]

Figure 1: Summary of data complexity results for FO queries

To prove membership of CERTAINANSWER$^{\in}$ in $\mathrm{P}^{\mathrm{NP}[\log n]}$, suppose the query Q is k-ary, and we are given a family of sets of k-ary tuples $\mathcal{X} = \{X_1, \ldots, X_n\}$ and a database D. For each $X_i \in \mathcal{X}$, we use the NP oracle to decide in parallel whether $X_i = \square(Q, D)$ (for each X_i the two calls to the oracle do not depend on each other and they can also be done in parallel).

For DP-hardness, we reduce from the problem of checking whether $\chi(G)$, the chromatic number of an undirected graph G, equals 4 [Rothe, 2003] and for $\mathrm{P}^{\mathrm{NP}[\log n]}$ -hardness, we reduce from the related problem of checking whether $\chi(G)$ is odd. With such a graph G, we associate the same database D_G as in the proof of Theorem 3.1. Using the exact coloring formula φ in the proof of Theorem 3.1, we define a query

$$Q(x) := C(x) \wedge \forall y\ (\varphi(y) \to L(x, y))$$

We claim that $\square(Q, D) = \{c_1, \ldots, c_n\}$ iff $\chi(G) = n$, which entails $\square(Q, D) = \{c_1, \ldots, c_4\}$ iff $\chi(G) = 4$ and $\square(Q, D) \in \{\{c_1, \ldots, c_j\} \mid j \text{ is odd and } 1 \leq j \leq |G|\}$ iff $\chi(G)$ is odd. Recall that $v(D_G) \models \varphi(c_i)$ iff c_i is a color in $\{c_1, ..., c_{|G|}\}$ and v represents an exact i-coloring of the graph. Now $v(D_G) \models Q(c_j)$ iff c_j is a color and there is no $i < j$ such that v represents an exact i-coloring of the graph, which holds exactly whenever $c_j \in \{c_1, ..., c_{\chi(G)}\}$.

$\square$

4 First-Order Rewritings for Best Answers

Considering arbitrary FO-queries brought us an intrinsic intractability result for all variants of best answers. This motivates restricting to unions of conjunctive queries, for which a polynomial time evaluation algorithm (in data complexity) already exists [Libkin, 2018b]. The resolution based procedure is however in sharp contrast with naïve evaluation, which allows to compute certain answers to unions of conjunctive queries via usual model checking. We thus initiate a descriptive complexity analysis of the best answers problem, showing that for unions of conjunctive queries, it can essentially be reduced - modulo a preprocessing of the query - to (naïve) evaluation of an FO-formula.

Given a union of conjunctive queries Q, our starting point towards an FO-rewriting for best answers is finding an FO-formula $Q_{\subseteq}(\bar{x}, \bar{y})$ encoding the inclusion of supports, i.e. selecting tuples $\bar{s}, \bar{t}$ over $\mathrm{adom}(D) \cup \mathrm{adom}(Q)$ iff $\mathsf{Supp}(Q, D, \bar{s}) \subseteq \mathsf{Supp}(Q, D, \bar{t})$. From $Q_{\subseteq}$ one can easily define an FO-formula selecting precisely all best answers to Q on D:

$$best_Q(\bar{x}) := \forall \bar{y} (Q_{\subseteq}(\bar{x}, \bar{y}) \to Q_{\subseteq}(\bar{y}, \bar{x})) \qquad (1)$$

We start by putting each conjunctive query in a normal form which eliminates repetition of variables, by introducing new equality atoms.

Definition 4.1 (NRV normal form)**.** *A conjunctive query Q is in non repeating variable normal form (NRV normal form) if it is of the form $Q(\bar{x}) = \exists \bar{y}, \bar{z}\ (q(\bar{y}, \bar{z}) \wedge e(\bar{y}, \bar{z}) \wedge \bar{x} = \bar{z})$ where variables in $\bar{x}\bar{y}\bar{z}$ are pairwise distinct, $\bar{x}$ and $\bar{z}$ have the same length, and:*

- *$q(\bar{y}, \bar{z})$ is a conjunction of relational atoms, where each free variable in $\bar{y}, \bar{z}$ has at most one occurrence in q,*
- *$e(\bar{y}, \bar{z})$ is a conjunction of equality atoms, possibly using constants.*

We say that $q(\bar{y}, \bar{z})$ is the relational subquery *of Q, and $e(\bar{y}, \bar{z}) \wedge \bar{x} = \bar{z}$ is the* equality subquery *of Q.*

Example 4.2. *The query $Q(x)$ from Example 2.1 is equivalent to $\exists y_1 y_2 z (R(y_1) \wedge S(y_2, z) \wedge y_1 = y_2 \wedge z = x)$, which is in NRV normal form.*

Clearly every conjunctive query Q is equivalent to a query in NRV normal form; moreover Q can be easily rewritten in NRV normal form (in linear time in the size of the query). Thus in what follows we assume w.l.o.g. that conjunctive queries are given in NRV normal form. Intuitively the NRV normal form allows us to separate the two ingredients of a conjunctive query : the existence of facts in some relations of the database on the one side, and a set of equality conditions on data values occurring in these facts, on the other side. The existence of facts does not depend on the valuation of nulls, and thus can be directly tested on the incomplete database. Instead equality atoms in an NRV normal form imply conditions that valuations need to satisfy in order for the query to hold. We can thus first concentrate on the support of equality subqueries. This will be encoded in FO and then integrated in the rewriting of the whole conjunctive query.

We introduce a notion of equivalence of database elements w.r.t. to a set of equalities. Intuitively equivalent elements of a tuple $\bar{t}$ are the ones which should be collapsed into a single value in order for a valuation of $\bar{t}$ to satisfy all the given equalities.

Definition 4.3. *Given a database D, a conjunction of equality atoms $\gamma(\bar{y})$ and an assignment $\nu : \bar{y} \cup adom(\gamma) \to adom(D) \cup adom(\gamma)$ preserving constants, we say that $u, u' \in adom(D) \cup adom(\gamma)$ are equivalent w.r.t. γ and ν and write $u \equiv^{\nu}_{\gamma} u'$, if either $u = u'$ or (u, u') belongs to the reflexive symmetric transitive closure of $\{(\nu(x), \nu(w)) \mid x = w \in \gamma\}$.*

The relation $\equiv^{\nu}_{\gamma}$ is clearly an equivalence relation over $\mathrm{adom}(D) \cup \mathrm{adom}(\gamma)$, where each element outside the range of ν forms a singleton equivalence class.

Example 4.4. *Let γ be $x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_6 = 1$. Let ν assign $\bot_i$ to x_i for $i \leq 5$, and $\bot_5$ to x_6. The equivalence classes of $\equiv^{\nu}_{\gamma}$ are $\{\bot_i \mid i \leq 3\}$ and $\{1, \bot_4, \bot_5\}$, plus one singleton for each other domain element.*

In what follows we denote by $\sim_\gamma$ the reflexive symmetric transitive closure of $\{(x,w) \mid x = w \in \gamma\}$. Note that this is an equivalence relation among variables and constants of γ.

We will be using the following formula to provide a syntactic characterisation of $\equiv^\nu_\gamma$, where m is the number of equivalence classes of $\sim_\gamma$: [1]

$$equiv_\gamma(\bar{y}, z, z') \ := \ z = z' \ \vee \bigvee_{\substack{u_1, v_1 \dots u_m, v_m \in \ \bar{y} \ \cup \ \text{adom}(\gamma) \ \mid \\ u_i \sim_\gamma v_i \text{ for all } 1 \le i \le m}} (z = u_1 \wedge z' = v_m \wedge \bigwedge_{1 \le i < m} v_i = u_{i+1})$$

Proposition 4.5. *Given an incomplete database D, a conjunction of equality atoms $\gamma(\bar{y})$ and an assignment $\nu(\bar{y}) = \bar{t}$ over $adom(D) \cup adom(\gamma)$, given s, s' in $\bar{t} \cup adom(\gamma)$, we have that $D \models equiv_\gamma(\bar{t}, s, s')$ if and only if $s \equiv^\nu_\gamma s'$.*

Intuitively this holds because each disjunct of $equiv_\gamma(\bar{t}, s, s')$ corresponds to a possible derivation of (s, s') in the reflexive symmetric transitive closure of $\{(\nu(x), \nu(w)) \mid x = w \in \gamma\}$, and one can prove that there is a bound only depending on γ on the number of steps of this derivation.

Example 4.6. *Let $\gamma := y_1 = y_2 \wedge z = x$ be the equality subquery of the query $Q(x)$ in Example 4.2. Up to logical equivalence, $equiv_\gamma(y_1, y_2, z, x, w, w')$ contains precisely the disjuncts $w = w'$, $w = y_1 \wedge w' = y_2$, $w = z \wedge w' = x$, $w = y_1 \wedge w' = x \wedge y_2 = z$, plus all disjuncts obtained from them by applying one or more of the following transformations : switch w and w', switch y_1 and y_2, switch x and z. Let D be the database from Example 2.1, then we have for instance $D \models equiv_\gamma(1, \bot_2, \bot_2, 1, a, a')$ and $D \models equiv_\gamma(1, \bot_2, \bot_2, \bot_2, a, a')$ for all $a, a' \in \{1, \bot_2\}$. Similarly $D \models equiv_\gamma(\bot_1, \bot_2, \bot_2, 1, a, a')$ for all $a, a' \in \{1, \bot_1, \bot_2\}$.*

As a consequence of Proposition 4.5, for fixed γ and $\bar{t}$, the relation $\{(s, s') \mid D \models equiv_\gamma(\bar{t}, s, s')\}$ is an equivalence relation over adom$(D) \cup$ adom(γ) where each element of adom(D) neither in $\bar{t}$ nor in adom(γ) forms a singleton equivalence class.

The formula $equiv_\gamma$ is a key ingredient towards a rewriting of a conjunctive query; in fact, as formalized in the following lemma, it selects precisely the pairs of elements of a tuple that a valuation needs to collapse to satisfy a set of equalities.

Lemma 4.7. *Let $\gamma(\bar{y})$ be a conjunction of equality atoms, D a database and $\nu(\bar{y}) = \bar{t}$ an assignment over $adom(D) \cup adom(\gamma)$. Assume v is a valuation of nulls. Then $v(D) \models \gamma(v(\bar{t}))$ if and only if $v(s) = v(s')$ for all s, s' such that $D \models equiv_\gamma(\bar{t}, s, s')$.*

Example 4.8. *Let γ and ν be as in Example 4.4, then Lemma 4.7 implies that a valuation $v(D) \models \gamma(v(\bar{t}))$ iff $v(\bot_i) = v(\bot_j)$ for all $i, j = 1..3$, and $v(\bot_i) = 1$ for all $i = 4, 5$.*

Formulas we write in the remainder of this section are over signature $\sigma \cup Null$, where σ is the database schema. In any incomplete database D over $\sigma \cup Null$, $Null$ is always interpreted by the set of nulls occurring in D (in accordance with the semantics of the SQL construct `IS NULL`). I.e. we allow rewritings to test whether a database element is null or not.

For $\gamma(\bar{y})$ a conjunction of equality atoms, using $equiv_\gamma$ we define a new formula $comp_\gamma(\bar{y})$ stating the existence of a valuation that collapses all equivalent elements of a tuple:

$$comp_\gamma(\bar{y}) \ := \ \forall z z' (equiv_\gamma(\bar{y}, z, z') \wedge \neg Null(z) \wedge \neg Null(z') \rightarrow z = z')$$

Notice that if $D \models comp_\gamma(\bar{t})$ then for each $s \in adom(D) \cup$ adom(γ) there exists at most one constant c such that $D \models equiv_\gamma(\bar{t}, s, c)$. In fact if for constants c_1 and c_2, $D \models equiv_\gamma(\bar{t}, s, c_1)$ and $D \models equiv_\gamma(\bar{t}, s, c_2)$, by transitivity $D \models equiv_\gamma(\bar{t}, c_1, c_2)$, implying $c_1 = c_2$.

Example 4.9. *Let D and γ be as in Example 4.6. Consider $comp_\gamma(y_1, y_2, z, x)$. Given the tuples selected by $equiv_\gamma$ in Example 4.6, we can conclude that $D \models comp_\gamma(1, \bot_2, \bot_2, 1)$.*

Proposition 4.10. *Let $\gamma(\bar{y})$ be a conjunction of equality atoms, D a database and $\nu(\bar{y}) = \bar{t}$ an assignment over $adom(D) \cup adom(\gamma)$, then $D \models comp_\gamma(\bar{t})$ if and only if there exists a valuation v of nulls such that $v(D) \models \gamma(v(\bar{t}))$.*

We are now ready to define a formula capturing the inclusion of supports between two conjunctions of equality atoms, which will be a crucial ingredient in our rewriting. Let $\gamma(\bar{x})$ and $\gamma'(\bar{y})$ be conjunctions of equality atoms with adom$(\gamma) =$ adom(γ'). We define :

$$imply_{\gamma,\gamma'}(\bar{x}, \bar{y}) \ := \ \forall z z' \ (equiv_{\gamma'}(\bar{y}, z, z') \rightarrow equiv_\gamma(\bar{x}, z, z'))$$

Example 4.11. *Let γ and D be as in Example 4.6. Let $\gamma' := y'_1 = y'_2 \wedge z' = x'$, then it follows from Example 4.6 that $D \models imply_{\gamma\gamma'}(\bot_1\bot_2\bot_2 1, 1\bot_2\bot_2\bot_2)$ and $D \models imply_{\gamma\gamma'}(1\bot_2\bot_2 1, 1\bot_2\bot_2\bot_2)$.*

Using Proposition 4.10 and Lemma 4.7 we obtain :

Proposition 4.12. *Let $\gamma(\bar{x})$, $\gamma'(\bar{y})$ be conjunctions of equality atoms with $adom(\gamma) = adom(\gamma')$, D a database and $\nu(\bar{y}) = \bar{t}$, $\nu'(\bar{y}) = \bar{t}'$ assignments over $adom(D) \cup adom(\gamma)$. Then $D \models imply_{\gamma,\gamma'}(\bar{t}, \bar{t}') \vee \neg comp_\gamma(\bar{t})$ iff for all valuations v, $v(D) \models \gamma(v(\bar{t}))$ implies $v(D) \models \gamma'(v(\bar{t}'))$.*

By combining Propositions 4.12 and 4.10 we also get :

Corollary 4.13. *Let $\gamma(\bar{y})$, $\gamma'(\bar{y})$ be conjunctions of equality atoms with $adom(\gamma) = adom(\gamma')$, D a database and $\nu(\bar{y}) = \bar{t}$, $\nu'(\bar{y}) = \bar{t}'$ assignments over $adom(D) \cup adom(\gamma)$. If $D \models comp_\gamma(\bar{t}) \wedge imply_{\gamma,\gamma'}(\bar{t}, \bar{t}')$, then $D \models comp_{\gamma'}(\bar{t}')$.*

We now go back to an arbitrary union of conjunctive queries of vocabulary σ in NRV-normal form :

$$Q(\bar{x}) := \bigvee_{1 \le i \le n} Q_i(\bar{x})$$

where each Q_i is in NRV-normal form with relational subquery $q_i(\bar{y}_i, \bar{z}_i)$ and equality subquery $eq_i(\bar{x}, \bar{y}_i, \bar{z}_i)$.

[1]Queries we write in the sequel can be domain dependent. So it is important to recall that we always use active domain semantics.

Lemma 4.14. *Let D be a database, v a valuation of D and $Q(\bar{x})$ a union of conjunctive queries in NRV-normal form, then $v \in Supp(Q, D, \bar{r})$ if and only there exists $i, \bar{s}$ and $\bar{t}$ such that $D \models q_i(\bar{s}, \bar{t}) \wedge comp_{eq_i}(\bar{r}\bar{s}\bar{t})$ and $v(D) \models eq_i(v(\bar{r}\bar{s}\bar{t}))$.*

We are now ready to define the FO-formula encoding the inclusion of supports.

$$Q_{\subseteq}(\bar{x}, \bar{x'}) := \bigwedge_{1 \le i \le n} (\forall \bar{y}\bar{z}((q_i(\bar{y}, \bar{z}) \wedge comp_{eq_i}(\bar{x}, \bar{y}, \bar{z})) \rightarrow$$
$$\bigvee_{1 \le j \le n} \exists \bar{y}'\bar{z}'(q_j(\bar{y}', \bar{z}') \wedge imply_{eq_i, eq_j}(\bar{x}\bar{y}\bar{z}, \bar{x}'\bar{y}'\bar{z}')) \;) \;)$$

Combining Lemmas 4.7, 4.14, Propositions 4.10, 4.12 and Corollary 4.13 we get :

Proposition 4.15. *$D \models Q_{\subseteq}(\bar{s}, \bar{t})$ iff $Supp(Q, D, \bar{s}) \subseteq Supp(Q, D, \bar{t})$.*

Recall that from $Q_{\subseteq}$ one can easily define a first order rewriting $best_Q(\bar{x})$ for best answers as in (1).

Theorem 4.16. *Given Q a union of conjunctive queries over schema σ and an incomplete database D, $\bar{t} \in \mathsf{Best}(Q, D)$ iff $D \models best_Q(\bar{t})$.*

Example 4.17. *For Q, D, γ, γ' as in Example 2.1 and 4.11 :*

$$Q_{\subseteq}(x, x') := \forall y_1 y_2 z((R(y_1) \wedge S(y_2, z) \wedge comp_{\gamma}(y_1, y_2, z, x)) \rightarrow$$
$$\exists y_1' y_2' z'(R_1(y_1') \wedge S(y_2', z') \wedge imply_{\gamma, \gamma'}(y_1 y_2 z x, y_1' y_2' z' x'))).$$

This allows to derive for instance $\mathsf{Supp}(Q, D, 1) \subseteq \mathsf{Supp}(Q, D, \bot_2)$ (as observed in Example 2.1). In fact the subquery $R(y_1) \wedge S(y_2, z) \wedge comp_{\gamma}(y_1, y_2, z, x)$ with free variables y_1, y_2, z, x selects on D tuples $(1, \bot_2, \bot_2, 1), (\bot_1, \bot_2, \bot_2, 1)$, and no other tuple with last element 1. Moreover as shown in Example 4.11

$$D \models imply_{\gamma\gamma'}(\bot_1 \bot_2 \bot_2 1, 1 \bot_2 \bot_2 \bot_2)$$
$$D \models imply_{\gamma\gamma'}(1 \bot_2 \bot_2 1, 1 \bot_2 \bot_2 \bot_2)$$

Thus $D \models Q_{\subseteq}(1, \bot_2)$. Similarly one can show $D \models Q_{\subseteq}(\bot_1, \bot_2)$ and therefore $D \models best_Q(\bot_2)$.

As a corollary of Theorem 4.16, for a union of conjunctive queries Q one can compute $\mathsf{Best}(Q, D)$ by first computing the formula $best_Q(\bar{x})$ from Q, then evaluating $best_Q$ on D. Since data complexity of FO query evaluation is DLOGSPACE (and in particular AC^0), this gives the following corollary :

Corollary 4.18. *For each fixed union of conjunctive queries Q, the data complexity of* BESTANSWER *is* DLOGSPACE.

Note that it was known from [Libkin, 2018b] that the data complexity of computing best answers for unions of conjunctive queries is polynomial time. In terms of combined complexity (i.e. when either Q, D and $\bar{a}$ are in the input), the rewriting approach (i.e. the procedure of computing $best_Q$ from Q and then evaluating $best_Q$ on D), can be easily shown to be in PSPACE. In fact it is well known that a first order query φ can be evaluated on a database D in space at most $qr(\varphi) \log |D| + log|\varphi|$, where $qr(\varphi)$ is the quantifier rank of φ. Note that although $best_Q$ has size exponential in Q, the quantifier rank of $best_Q$ is linear in the size of Q. Thus whether $\bar{a} \in best(Q, D)$ can be checked using space $O(|Q|, |D|)$. Moreover one can easily check that $best_Q$ can be computed from Q in space polynomial in the size of $|Q|$. Since space bounded computations can be composed without storing the intermediate output, computing $best_Q$ from Q and then evaluating $best_Q$ on D can be done overall in PSPACE in the size of $|Q|$ and $|D|$. The rewriting approach thus implies a PSPACE upper bound for the combined complexity of BESTANSWER for unions of conjunctive queries. However we can show that the problem actually stands in the third level of the polynomial hierarchy.

Theorem 4.19. *For unions of conjunctive queries, combined complexity of* BESTANSWER *is Π_3^p-complete. Hardness already holds for conjunctive queries.*

Therefore under standard complexity theoretic assumptions, our rewriting approach is not optimal in terms of combined complexity, as it is often the case with generic approaches. However it has the advantage of exploiting standard FO query evaluation, which despite the PSPACE combined complexity, is highly optimised in database systems and works well in practice.

5 Future Work

Constraints (e.g., keys and functional dependencies) are known to raise the complexity of finding certain answers [Calì *et al.*, 2003a]. They appear in another model of incompleteness - conditional tables [Imieliński and Lipski, 1984] - that in general leads to higher complexity of query evaluation [Abiteboul *et al.*, 1995] but are nonetheless useful in several applications [Arenas *et al.*, 2013]. We would like to explore how our rewriting techniques interact with integrity constraints.

In another direction, while we focused on FO-rewritings, we could consider more expressive rewriting languages such as Datalog or fixed point logics, as it is common in the context of OBDA, query answering using views, or consistent query answering [Bienvenu and Ortiz, 2015; Francis *et al.*, 2015; Bertossi, 2011]. These more expressive logics are likely to be able to express rewritings of larger classes of queries than unions of conjunctive queries.

We would also like to investigate how our techniques can be extended to different semantics of incompleteness. We used here the closed-world semantics [Abiteboul *et al.*, 1995; Imieliński and Lipski, 1984; van der Meyden, 1998], in which data values are the only missing information, but there are other possible semantics, e.g. needed in order to cope with data inconsistencies [Calì *et al.*, 2003a], where query rewritings could still be found.

Finally, we would like to investigate how techniques developed in this paper could be extended to study rewritings of certain answers.

Acknowledgements

We thank the anonymous referees and Leonid Libkin for their useful feedback. This work was supported by contract ANR-18-CE40-0031 QUID.

References

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

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and VictorVianu. *Foundations of Databases*. Addison-Wesley, Boston, MA, USA, 1995.

[Amendola and Libkin, 2018] Giovanni Amendola and Leonid Libkin. Explainable certain answers. In *IJCAI*, pages 1683–1690, Stockholm, Sweden, August 2018.

[Arenas and Ugarte, 2017] Marcelo Arenas and Martín Ugarte. Designing a query language for RDF: marrying open and closed worlds. *ACM TODS*, 42(4):21:1–21:46, 2017.

[Arenas *et al.*, 2013] Marcelo Arenas, Jorge Pérez, and Juan L. Reutter. Data exchange beyond complete data. *J. ACM*, 60(4):28:1–28:59, 2013.

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

[Barceló *et al.*, 2014] Pablo Barceló, Leonid Libkin, and Miguel Romero. Efficient approximations of conjunctive queries. *SIAM J. Comput.*, 43(3):1085–1130, 2014.

[Bertossi, 2011] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Bienvenu and Bourgaux, 2016] Meghyn Bienvenu and Camille Bourgaux. Inconsistency-tolerant querying of description logic knowledge bases. In *Tutorial Lectures of RW Summer School*, pages 156–202. LNCS, 2016.

[Bienvenu and Ortiz, 2015] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web. Web Logic Rules - 11th International Summer School*, pages 218–307, Berlin, Germany, August 2015. Springer.

[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, June 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, Acapulco, Mexico, August 2003. Morgan Kaufmann.

[Calì *et al.*, 2013] Andrea Calì, Diego Calvanese, and Maurizio Lenzerini. Rewrite and conquer: Dealing with integrity constraints in data integrations. In *Seminal Contributions to Information Systems Engineering*, pages 353–359, 2013.

[Calvanese *et al.*, 2000] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is view-based query rewriting? In *KRDB*, pages 17–27, Berlin, Germany, August 2000.

[Calvanese *et al.*, 2006] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Epistemic first-order queries over description logic knowledge bases. In *DL*, Windermere, Lake District, UK, May-June 2006.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. *TCS*, 371(3):169–182, March 2007.

[Eiter and Gottlob, 1997] Thomas Eiter and Georg Gottlob. The complexity class theta$_2$$^{\text{p}}$: Recent results and applications in AI and modal logic. In *FCT*, pages 1–18, Kraków, Poland, September 1997. Springer.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM TODS*, 30(1):174–210, 2005.

[Francis *et al.*, 2015] Nadime Francis, Luc Segoufin, and Cristina Sirangelo. Datalog rewritings of regular path queries using views. *Logical Methods in Computer Science*, 11(4), 2015.

[Geerts *et al.*, 2013] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The LLUNATIC data-cleaning framework. *PVLDB*, 6(9):625–636, 2013.

[Gheerbrant and Libkin, 2015] Amélie Gheerbrant and Leonid Libkin. Certain answers over incomplete xml documents: Extending tractability boundary. *ACM ToCS*, 57(4):892–926, July 2015.

[Gottlob *et al.*, 2015] Georg Gottlob, Marco Manna, and Andreas Pieris. Polynomial rewritings for linear existential rules. In *IJCAI*, pages 2992–2998, Buenos Aires, Argentina, August 2015. AAAI Press.

[Gottlob, 1995] Georg Gottlob. NP trees and Carnap's modal logic. *Journal of the ACM*, 42(2):421–457, March 1995.

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

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *ACM PODS*, pages 233–246, Madison, USA, June 2002.

[Libkin, 2018a] Leonid Libkin. Certain answers as objects of knowledge. *Artificial Intellligence*, 232:195–207, 2018.

[Libkin, 2018b] Leonid Libkin. Certain answers meet zero-one laws. In *ACM PODS*, pages 1–19, Houston, USA, June 2018.

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

[Rothe, 2003] Jörg Rothe. Exact complexity of exact-four-colorability. *Inf. Process. Lett.*, 87(1):7–12, 2003.

[van der Meyden, 1998] Ron van der Meyden. Logical approaches to incomplete information: a survey. In *Logics for databases and information systems*, pages 307–356, Norwell, MA, USA, 1998. Kluwer Academic Publishers.

[Wagner, 1990] Klaus W. Wagner. Bounded query classes. *SIAM J. Comput.*, 19(5):833–846, 1990.