Explainable Certain Answers

Giovanni Amendola[1], **Leonid Libkin**[2]
[1] University of Calabria
[2] University of Edinburgh
amendola@mat.unical.it, libkin@inf.ed.ac.uk

Abstract

When a dataset is not fully specified and can represent many possible worlds, one commonly answers queries by computing certain answers to them. A natural way of defining certainty is to say that an answer is certain if it is consistent with query answers in all possible worlds, and is furthermore the most informative answer with this property. However, the existence and complexity of such answers is not yet well understood even for relational databases. Thus in applications one tends to use different notions, essentially the intersection of query answers in possible worlds. However, justification of such notions has long been questioned. This leads to two problems: are certain answers based on informativeness feasible in applications? and can a clean justification be provided for intersection-based notions?

Our goal is to answer both. For the former, we show that such answers may not exist, or be very large, even in simple cases of querying incomplete data. For the latter, we add the concept of *explanations* to the notion of informativeness: it shows not only that one object is more informative than the other, but also says *why* this is so. This leads to a modified notion of certainty: explainable certain answers. We present a general framework for reasoning about them, and show that for open and closed world relational databases, they are precisely the common intersection-based notions of certainty.

1 Introduction

Problems where AI and data management techniques meet are characterized by the need to compute answers to queries over many possible worlds, such as instances of a global schema in data integration [Calì *et al.*, 2003; Lenzerini, 2002] or solutions in data exchange [Arenas *et al.*, 2014; Fagin *et al.*, 2005] or results of chasing a database with ontology constraints [Calvanese *et al.*, 2007; Calì *et al.*, 2012]. This is due to the fact that datasets one uses for query answering typically contain null values (i.e., unknown individuals), and thus they represent potentially many possible complete worlds (i.e., datasets without null values). Query answering that takes all of them into account is based on the notion of *certain answers*.

Intuitively, certain answers are those that are true in all possible worlds. However, defining this notion formally is not completely straightforward. There are two different approaches. One of them, applicable when databases are relations (sets or multisets of tuples), simply looks for tuples that are present in all answers [Imielinski and Lipski, 1984], i.e., the intersection of query answers in all possible worlds. We refer to this notion as *intersection-based*. In fact one often uses its slight generalization that also allows tuples with null values in the output [Lipski, 1984]. This is what one typically finds in the literature on querying incomplete information, or its applications such as data integration, data exchange, or ontology-based data access (OBDA).

While this notion looks rather natural, it comes with little justification, which led to an attempt [Libkin, 2016] to provide a systematic approach to defining certain answers; it generalized earlier approaches that utilized techniques from programming semantics [Buneman *et al.*, 1991; Rounds, 1991]. Suppose we have an incomplete object x, and let $[\![x]\!]$, the *semantics* of x, be the set of all of its possible worlds. To answer a query Q on x, we need to extract certain information from $Q([\![x]\!]) = \{Q(y) \mid y \in [\![x]\!]\}$, where $Q(y)$ is the answer to Q on y. The key element of this approach is the assumption that we have an ordering on query answers: $a \preccurlyeq a'$ means that a' is at least as informative as a (i.e., a' contains at least all information from a). This defines the notion of a *candidate answer*, which is an object a such that $a \preccurlyeq Q(y)$ for all $y \in [\![x]\!]$. In other words, a candidate answer is no more informative than any of query answers in possible worlds. A *certain answer* is the maximal candidate answer, i.e., the most informative answer consistent with answers in all possible worlds (formally, it is the greatest lower bound of the set $Q([\![x]\!])$). We refer to this notion as *informativeness-based*.

To compare this approach with the intersection-based approach for the relational data model, we need to explain what the incomplete objects are and how the informativeness ordering is interpreted. Most commonly, incompleteness is represented by *marked nulls* [Imielinski and Lipski, 1984] (this is in fact the model typically used in applications such as data integration and exchange and OBDA). In databases with marked nulls, some of the entries are nulls that represent currently unknown values. Possible worlds are obtained by as-

signing values to nulls; e.g., by means of a function v on nulls whose values are constants that can appear in complete databases.

Queries over relational databases produce relations as well. Let R and R' be two relations that can appear as a query answers, and assume that R' has no nulls, i.e., it is a query answer on a complete database. Then R' is at least as informative as R if $v(R) \subseteq R'$ for some valuation v of nulls in R. That is, R' provides additional information by instantiating nulls in R and adding some tuples. With these notions in place, we can now see how the two approaches to defining certainty compare.

Example 1. Consider an incomplete database $D = \{R(1,1), R(2,1), R(2,\bot), S(\bot,1)\}$ with three tuples in relation R and one tuple in relation S, where $\bot$ denotes a null value. Let $Q(x,y) = R(x,y) \wedge \neg S(x,y)$ be the query computing the difference of R and S. Possible worlds of D could be of three kinds:

- $D_1 = \{R(1,1), R(2,1), S(1,1)\}$, if $\bot$ is mapped to 1;
- $D_2 = \{R(1,1), R(2,1), R(2,2), S(2,1)\}$, if $\bot$ is mapped to 2, and
- $D_c = \{R(1,1), R(2,1), R(2,c), S(c,1)\}$, if $\bot$ is mapped to some other value $c \neq 1, 2$.

This gives us the following sets of answers

- $Q(D_1) = \{(2,1)\}$,
- $Q(D_2) = \{(1,1), (2,2)\}$, and
- $Q(D_c) = \{(1,1), (2,1), (2,c)\}$, for $c \neq 1, 2$.

There are no tuples present in all these answers, and thus the standard intersection-based approach will return the empty set. Its generalization that permits nulls in answers will return the single tuple $(2,\bot)$. Indeed, if $\bot$ is mapped to 1, then $(2,1) \in Q(D_1)$; if $\bot$ is mapped to 2, then $(2,2) \in Q(D_2)$; and if $\bot$ is mapped to $c \neq 1, 2$, then $(2,c) \in Q(D_c)$.

The informativeness-based notion of certain answer results in the set $A = \{(\bot_1, 1), (2, \bot_2)\}$. Indeed, one can see that for each c, there is v such that $v(A) \subseteq Q(D_c)$, and it is not hard to show that A is the greatest lower bound of all $Q(D_c)$s. □

Thus, the informativeness-based notion of certainty may be different from the intersection-based notion. We do not yet understand the informativeness-based notion well, however. For example, we do not know whether the certain answer defined as the greatest lower bound always exists, nor do we know how large it can be when it does exist. The latter question was partially answered in [Arenas *et al.*, 2017]: even for unions of conjunctive queries with inequalities, it can be of super-polynomial size under the open-world interpretation of incomplete databases. We complete the picture. First we show that for open-world, even for first-order queries, such certain answers need not exist. For closed-world semantics, certain answers exist for queries expressible in any reasonable logical language, but could still be of super-polynomial size. So suddenly the informativeness-based approach to defining certain answers faces a very serious obstacle of either non-existence or infeasible sizes of answers.

The reason for this bad behavior lies in the excessive permissiveness of the definition of certain answers with respect to information content. If $y \in [\![x]\!]$, there must be a specific *explanation* as to why y is a possible world. However, we only require $a \preccurlyeq Q(y)$ if a is candidate answer: that is, such an answer is less informative than $Q(y)$ but for reason that may have nothing to do with the fact that y is a possible world for x. This may lead to bizarre candidate answers. Indeed, returning to Example 1, consider the tuple $(1, \bot_1)$. It matches $Q(D_1)$, obtained by interpreting $\bot$ as 1, when $\bot_1$ is mapped to 2, and it matches $Q(D_2)$, obtained by interpreting $\bot$ as 2, when $\bot_1$ is mapped to 1. Thus, there is no connection between how a possible world is obtained, and why the tuple is in the answer to Q in that possible world.

Instead of the very permissive definition of candidate and certain answers, it seems much more natural to require, for a candidate answer a, that $a \preccurlyeq Q(y)$ for *the same* reason that y is in $[\![x]\!]$: that is, whatever explanation is given to $y \in [\![x]\!]$, it should also explain why $a \preccurlyeq Q(y)$. This idea gives us the notion of *explainable* candidate answers, and maximal such answer is the *explainable certain answer*.

Our main contribution is to extend the theory of informativeness-based notion of certainty to include explanations of why tuples appear in the answer, and to relate explainable certain answers to the classical intersection-based approach. As the first step, we develop a general, datamodel-independent, framework to reason about explanations applied to incomplete objects, and to query answers. Essentially, explanations are viewed as operations that can be applied to objects to make them more informative; such explanations compose, and naturally lead to notions of explainable candidate and certain answers.

We then see how these notions look in the familiar relational model, under both open- and closed-world semantics of incompleteness. Our main result in this case is that for both semantics, explainable certain answers always exist, are of at most polynomial size in the size of the input, and are, essentially, the intersection-based answers (to be more precise, their slight extension that allows nulls in the output: the one that produced the answer $(2,\bot)$ in Example 1). Thus, the standard notion so common in data management and reasoning literature is well justified after all, by applying the concept of explanations to both database objects and query answers.

Organization. Basic definitions are in Section 2. Existence and size of answers as greatest lower bounds is studied in Section 3. Section 4 compares the intersection- and informativeness-based notions of certainty. Section 5 introduces the framework of explanations, and Section 6 shows what explainable certain answers are for common semantics.

2 Preliminaries

Incomplete databases. An incomplete database D represents many possible worlds, or complete databases, known as its *semantics* $[\![D]\!]$. As a concrete model of incompleteness we look at relational databases with *marked nulls* that dominate applications such as data integration, OBDA, and data exchange [Arenas *et al.*, 2014; Lenzerini, 2002; Bienvenu and Ortiz, 2015]. Entries in databases come from the sets of *constants* and *nulls*, denoted by Const and Null. We use the symbol $\bot$ for nulls. A database D is a set of relations over $\mathsf{Const} \cup \mathsf{Null}$;

a k-ary relation is a finite subset of $(\mathsf{Const} \cup \mathsf{Null})^k$. Sets of constants and nulls that occur in D are denoted by $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$. Its *domain* is $dom(D) = \mathsf{Const}(D) \cup \mathsf{Null}(D)$. A *complete* database has no nulls.

A *homomorphism* $h : D \to D'$ is a mapping from $dom(D)$ to $dom(D')$ such that $h(c) = c$ for $c \in \mathsf{Const}(D)$, and for each tuple $\bar{a}$ in relation R of D, the tuple $h(\bar{a})$ is in relation R of D'. The image of h is denoted by $h(D)$.

A homomorphism v is a *valuation* if its range contains only constants, i.e., $v(\bot) \in \mathsf{Const}$ for each null (we shall be using v for valuations, and h for arbitrary homomorphisms). Two most common semantics of incompleteness, under open- and closed-world assumptions (OWA and CWA) are given as follows [Abiteboul *et al.*, 1995; Imielinski and Lipski, 1984]:

$$\begin{array}{rcl} [\![D]\!]_{\text{OWA}} & = & \{D' \text{ complete} \mid \exists \text{ valuation } v : D \to D'\} \\ [\![D]\!]_{\text{CWA}} & = & \{D' \text{ complete} \mid \exists \text{ valuation } v : D' = v(D)\} \end{array}$$

That is, under CWA, one just replaces nulls with constants, and under OWA, one can add extra tuples, cf. [Reiter, 1977].

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 under semantics $[\![\,]\!]$, one has to analyze $Q([\![D]\!]) = \{Q(D') \mid D' \in [\![D]\!]\}$. The definition one sees most commonly in the literature is simply $\bigcap Q([\![D]\!]) = \bigcap\{Q(D') \mid D' \in [\![D]\!]\}$. We shall consider a slightly more permissive definition:

$$\begin{array}{rcl} \Box_{\text{OWA}}(Q, D) & = & \{\bar{a} \mid \forall v : D \to D' \quad v(\bar{a}) \in Q(D')\} \\ \Box_{\text{CWA}}(Q, D) & = & \{\bar{a} \mid \forall v : \quad v(\bar{a}) \in Q(v(D))\} \end{array}$$

where v ranges over valuations on D, and $\bar{a}$ is a tuple over constants and nulls. The latter comes from [Lipski, 1984], and has often been used as a substitute for intersection-based definition since null-free tuples in $\Box_{\text{CWA}}(Q, D)$ and $\bigcap Q([\![D]\!]_{\text{CWA}})$ are exactly the same (and likewise for OWA). For Boolean (true/false) queries, where the standard representations of true and false are $\{()\}$ and $\varnothing$ (i.e., the set containing the empty tuple $()$, and the empty set), $\Box(Q, D)$ and $\bigcap Q([\![D]\!])$ are the same for both OWA and CWA semantics. Otherwise $\Box_{\text{OWA}}(Q, D)$ and $\Box_{\text{CWA}}(Q, D)$ may produce informative tuples with nulls. For instance, if Q returns a relation R in D, then $\Box_{\text{OWA}}(Q, D) = \Box_{\text{CWA}}(Q, D) = R$, while both $\bigcap Q([\![D]\!]_{\text{OWA}})$ and $\bigcap Q([\![D]\!]_{\text{CWA}})$ contain only null-free tuples in R.

Query languages. As our basic query language, we shall use *first-order predicate logic* (FO) over the relational vocabulary. Its formulae are built up from relational atoms $R(\bar{x})$ and equality atoms $x = y$, using Boolean connectives $\wedge, \vee, \neg$, and quantifiers $\exists, \forall$. The $\exists, \wedge$-fragment of FO is known as *conjunctive queries*. Their unions are denoted by UCQ (unions of conjunctive queries); in terms of their expressiveness, they correspond to the $\exists, \wedge, \vee$-fragment of FO. If inequality atoms $x \neq y$ are allowed in them, we have the class of unions of conjunctive queries with inequalities, denoted by $\text{UCQ}^{\neq}$.

3 Query answers as greatest lower bounds

We now outline a general approach to viewing certain query answers as greatest lower bounds in orderings induced by information content, and then see what it yields for answering relational queries, under both OWA and CWA semantics.

A general framework

We review the basics of the framework of [Libkin, 2016]. A *database domain* is a triple $\langle \mathbb{D}, \mathbb{C}, [\![\,]\!] \rangle$ where $\mathbb{D}$ is a set of database objects (e.g., all databases of a given schema), $\mathbb{C} \subseteq \mathbb{D}$ is a set of complete objects, and $[\![\,]\!]$ is the semantic function that assigns complete objects (possible worlds) to an incomplete object, i.e., $[\![x]\!] \subseteq \mathbb{C}$. With such a domain, we associate its *information ordering* given by $x \preccurlyeq y$ iff $[\![y]\!] \subseteq [\![x]\!]$. The intuition is that the more possible worlds there are for x, the less information about x we have.

Given two database domains $\langle \mathbb{D}', \mathbb{C}', [\![\,]\!]' \rangle$ and $\langle \mathbb{D}, \mathbb{C}, [\![\,]\!] \rangle$, a query is a mapping $Q : \mathbb{C}' \to \mathbb{C}$, i.e., a mapping from complete objects (query inputs) to complete objects (query answers). This is consistent with the standard query languages. To extend Q to arbitrary elements of $\mathbb{D}$ in a correct way, we need to extract certain information from $Q([\![x]\!])$. We say that a is a *candidate answer* to Q on x if $a \preccurlyeq Q(c)$ for every $c \in [\![x]\!]$, and a is a *certain answer* to Q on x if it is maximal among candidate answers. In other words, the certain answer is the *greatest lower bound* of $Q([\![x]\!])$ in ordering $\preccurlyeq$, denoted by $\mathsf{glb}_{\preccurlyeq} Q([\![x]\!])$. Depending on a concrete ordering $\preccurlyeq$ on $\mathbb{D}$, certain answers may or may not exist. If certain answers do exist, then $[\![y]\!]' \subseteq [\![x]\!]'$ implies $\mathsf{glb}_{\preccurlyeq} Q([\![x]\!]) \preccurlyeq \mathsf{glb}_{\preccurlyeq} Q([\![y]\!])$. That is, more informative query inputs yield more informative query answers.

The framework for relational databases

When elements of database domains are relational databases, for query inputs we use either OWA or CWA semantics. For the semantics of *answers*, it is common to use OWA. Indeed, a partial query answer can be improved in two ways: either by finding more tuples, or by instantiating nulls with values. This is precisely what the OWA semantics does. The ordering it generates, $D \sqsubseteq D'$ iff $[\![D']\!]_{\text{OWA}} \subseteq [\![D]\!]_{\text{OWA}}$, has a nice description: $D \sqsubseteq D'$ iff there is a homomorphism $h : D \to D'$, see [Gheerbrant *et al.*, 2014].

This is a well-known and studied relation [Hell and Nešetřil, 2004], and we now recall some basic notions related to it. It is a preorder, that is, reflexive and transitive. If both $D \sqsubseteq D'$ and $D' \sqsubseteq D$ hold, we say that D and D' are homomorphically equivalent and write $D \approx D'$; note that $\approx$ is an equivalence relation. A relational structure is a *core* if it has no proper substructure that is homomorphically equivalent to it. A substructure D' of D is called a core of D if it is a core and $D \approx D'$. It is known that up to isomorphism, there is only one core of D, and thus one speaks of *the core* of D, denoted by $\text{core}(D)$. Any two homomorphically equivalent structures will have the same core (up to isomorphism).

The certain answer to a relational query Q on an incomplete database D is $\mathsf{glb}_{\sqsubseteq} Q([\![D]\!])$, where $[\![\,]\!]$ is either $[\![\,]\!]_{\text{OWA}}$ or $[\![\,]\!]_{\text{CWA}}$, and the greatest lower bound is taken with respect to $\sqsubseteq$. Since $\sqsubseteq$ is a preorder, $\mathsf{glb}_{\sqsubseteq}$ is defined up to homomorphic equivalence, and one typically takes the core of these homomorphically equivalent structures to represent $\mathsf{glb}_{\sqsubseteq}$.

For structures D_1 and D_2, their greatest lower bound is any structure homomorphically equivalent to $D_1 \times D_2$ [Hell and Nešetřil, 2004]. Thus, there is the smallest such structure,

namely core$(D_1 \times D_2)$. If homomorphisms are required to preserve constants, the definition of the product $D_1 \times D_2$ needs a small adjustment: if $a_i \in dom(D_i)$, for $i = 1, 2$, then we replace the pair (a_1, a_2) in the domain of $D_1 \times D_2$ by a_1 if $a_1 = a_2 \in \mathsf{Const}$ and by a fresh null otherwise [ten Cate and Dalmau, 2015; Libkin, 2011]).

While $\mathsf{glb}_{\sqsubseteq}$ exists for finite sets, our goal is to find it for *infinite* sets $Q(\llbracket D \rrbracket)$. That is what we analyze now.

Certain answers for UCQs.

In the case of unions of conjunctive queries, there is an easy description of these greatest lower bounds. By $Q(D)$ we mean the *naïve* evaluation of Q on D that treats nulls simply as new constant values. Then, if $Q \in \mathrm{UCQ}$,

$$\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{OWA}}) \;=\; \mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{CWA}}) \;=\; Q(D)\,,$$

see [Imielinski and Lipski, 1984; Gheerbrant *et al.*, 2014; Libkin, 2016]. Thus, it is of interest to us what happens for more complex queries, e.g., $\mathrm{UCQ}^{\neq}$ or FO queries.

Bounds under OWA.

Now we have a relational query Q and we want to compute $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{OWA}})$ for a database D. It was shown already in [Arenas *et al.*, 2017] that its size could be exponential even for $\mathrm{UCQ}^{\neq}$ queries. That is, $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{OWA}})$ always exists if $Q \in \mathrm{UCQ}^{\neq}$, and, moreover, there exists a sequence D_n, $n \in \mathbb{N}$, of databases of increasing size and a $\mathrm{UCQ}^{\neq}$ query Q such that the size of $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D_n \rrbracket_{\mathrm{OWA}})$ is $2^{O(\|D_n\|)}$, where the size $\|D\|$ of a database is measured as the number of tuples in it.

When we move to arbitrary FO queries, the situation is much worse.

Theorem 1. *There is an* FO *query Q and a database D such that $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{OWA}})$ does not exist.*

Proof idea. Let D contain a unary relation U with a single element, and an empty binary relation R. Let q state that for each element of R, its indegree and outdegree are 1, and let Q return R if q is true, and $\{(x, x) \mid x \in U\}$ otherwise. A database D' in $\llbracket D \rrbracket_{\mathrm{OWA}}$ contains a graph R and a set U; then Q returns R if it is a disjoint union of directed cycles, and single loops on elements of U otherwise.

Let C_n be a directed cycle of length n. From the above and the fact that $C_n \subseteq C_n \sqcup C_m$ (where $\sqcup$ denotes disjoint union) it is easy to derive that $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{OWA}}) = \mathsf{glb}_{\sqsubseteq}\{C_n \mid n > 0\}$. Since $C_{mn} \sqsubseteq C_m$, we then get that the latter equals $\mathsf{glb}_{\sqsubseteq}\{C_{n!} \mid n > 0\}$. Now one can use techniques from [Hell and Nešetřil, 2004] to show that such a greatest lower bound does not exist. □

Bounds under CWA.

For CWA, we can recover the existence of greatest lower bounds for a large class of queries, but we still have very high bounds on their sizes. Recall that a query Q is *generic* [Abiteboul *et al.*, 1995] if $Q(\pi(D)) = \pi(Q(D))$ for every permutation of the domain (i.e., for a bijection $\pi : dom(D) \to dom(D)$). All queries in FO and many other logics over the relational vocabulary are such.

Theorem 2. *If Q is a generic query, then $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{CWA}})$ always exists, and its size is bounded by $2^{2^{O(\|D\|)}}$.*

Proof idea. Given a database D, a valuation v is *admissible* if its range is contained in $\mathsf{Const}(D)$ together with $|\,\mathsf{Null}(D)|+1$ new constants not present in D. Let $\mathcal{A}(Q, D) = \{Q(v(D)) \mid v \text{ admissible valuation}\}$. Note that $\mathcal{A}(Q, D) \subseteq Q(\llbracket D \rrbracket_{\mathrm{CWA}})$, and $\mathcal{A}(Q, D)$ is a finite set. The key lemma is to show that $\mathsf{glb}_{\sqsubseteq}(\llbracket D \rrbracket_{\mathrm{CWA}}) = \mathsf{glb}_{\sqsubseteq}\mathcal{A}(Q, D)$.

To construct $\mathsf{glb}_{\sqsubseteq}\mathcal{A}(Q, D)$, we have to find the core of the product of the elements of $\mathcal{A}(Q, D)$; since the number of admissible valuations is exponential, the product size is at most doubly exponential. □

As for the lower bound, we can adapt the construction of [Arenas *et al.*, 2017] to show the following:

Proposition 1. *There exists a family of incomplete databases D_n, for $n \in \mathbb{N}$, of increasing size, and a $\mathrm{UCQ}^{\neq}$ query Q such that the size of $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D_n \rrbracket_{\mathrm{CWA}})$ is $2^{O(\|D_n\|)}$.*

This still leaves a gap between the exponential lower bound and the double exponential upper bound. To give an idea why it is very hard to close the gap, note that lower bound proofs are based on constructing, for each n, a polynomial-size family of cores of size up to n that are closed under product; this results in an exponential size greatest lower bound. For a double-exponential bound, we would need to find an exponential-size family of cores closed under product. However the existence of such a family is not known.

A double exponential lower bound can be produced for monadic second-order logic, MSO (the fragment of second-order where all second-order quantification is over sets, and both first- and second-order free variables are allowed).

Proposition 2. *There is a family of incomplete databases D_n, for $n \in \mathbb{N}$, of increasing size, and an* MSO *query Q such that the size of $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D_n \rrbracket_{\mathrm{CWA}})$ is $2^{2^{\Omega(\|D_n\|)}}$.*

Proof idea. We first show how with an MSO query, we can generate, on $\llbracket D_n \rrbracket_{\mathrm{CWA}}$, outputs which are directed cycles of lengths up to 2^n, where D_n has $O(n)$ tuples. Thus, $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D_n \rrbracket_{\mathrm{CWA}})$ must contain core$(\times_{i \leqslant 2^n} C_i)$, which is known to be C_N, where N is the least common multiplier of numbers up to 2^n. Thus, $N \geqslant \prod\{p \leqslant 2^n \mid p \text{ prime}\}$, and $\ln N \geqslant \theta(2^n)$, where $\theta(x)$ is the sum of $\ln p$ over prime $p \leqslant x$. It is known that $\theta(x) \geqslant x$ for sufficiently large x, see [Rosser and Schoenfeld, 1962], and thus $N \geqslant e^{2^n}$, from which the bound is easily derived. □

4 Comparing certainty notions

We have defined certain answers as greatest lower bounds $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_*)$ when $*$ is OWA or CWA. On the other hand, we have standard notions of certain answers in the literature, namely $\Box_{\mathrm{CWA}}(Q, D)$ and $\Box_{\mathrm{OWA}}(Q, D)$. How do they compare?

We know that $\Box_{\mathrm{CWA}}(Q, D)$ and $\Box_{\mathrm{OWA}}(Q, D)$ are candidate answers, under the appropriate semantics; that is, $\Box_*(Q, D) \;\sqsubseteq\; Q(D')$ for each $D' \in \llbracket D \rrbracket_*$. However, they are not in general informativeness-based certain answers: we already saw, in Example 1, that $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{CWA}})$ and $\Box_{\mathrm{CWA}}(Q, D)$ can be different. In that example, $\Box_{\mathrm{CWA}}(Q, D) = \{(2, \bot)\}$, while $\mathsf{glb}_{\sqsubseteq} Q(\llbracket D \rrbracket_{\mathrm{CWA}}) = \{(\bot_1, 1), (2, \bot_2)\}$. A similar example can be constructed under OWA as well.

In fact Example 1 showed that the intersection-based notion can miss some of the answers justified under the informativeness-based notion. On the other hand, the informativeness-based notion of certainty does not behave well computationally: it may not exist, or may be prohibitively large. To reconcile good properties of the two notions, it would thus be nice to produce *proper justifications* for answers of the form $\Box_*(Q, D)$. To do so, we observe the following: in Example 1, each of the possible worlds for D is *explained* by means of a valuation v. If we take, in that example, the tuple $\bar{a} = (\bot_1, 1)$ which belongs to $\mathsf{glb}_{\sqsubseteq} Q([\![D]\!]_{\text{CWA}})$ but not to $\Box_{\text{CWA}}(Q, D)$, we shall always have some valuation v' so that $v'(\bar{a}) \in Q(v(D))$, but we cannot ensure that v' always equals v. In other words, we cannot ensure that *the same explanation* accounts for a possible world for an incomplete database, and for the tuple being an answer in that possible world.

This is an informal description of why the notions of $\Box_*(Q, D)$ are justified: they appear to be the most informative answers under the assumption that *the same explanation accounts for a possible world and the query answer in that possible world*. We next formalize the notion of explanations.

5 Explanations: the framework

We now present an abstract framework to reason about explanations applied to incomplete database objects, while providing analogies with the common OWA and CWA relational semantics. This is done to keep the intuition clear; detailed treatment of these semantics will be presented in Section 6.

Assume that we have a set $\mathbb{D}$ of database objects, as in Section 3. On this set we have an *oblivious* preorder $\preceq_\varepsilon$; intuitively, $x \preceq_\varepsilon y$ means that we know that y is at least as informative as x without having to provide additional information. For example, under OWA, this will be the subset ordering $D \subseteq D'$ (tuples can be added freely), and under CWA, it is just the identity $D = D'$.

Next, we need *actions*; these are functions on objects that return objects. Some actions with special properties will be viewed as *explanations* of incomplete information. The set of actions Ω is a monoid: actions can be composed, and there is the identity (unit) action. This is captured by:

Definition 1 (Action Structure)**.** *An action structure $\mathbb{A}$ consists of:*

- *a set $\mathbb{D}$ of database objects with an oblivious preorder $\preceq_\varepsilon$;*
- *a monoid Ω with an associative composition operation denoted by $\omega\omega'$ and unit ε (i.e., $\varepsilon\omega = \omega\varepsilon = \omega$); and*
- *a function $\mathbb{D} \times \Omega \to \mathbb{D}$, whose result on $x \in \mathbb{D}$ and $\omega \in \Omega$ is denoted by x^ω, such that $x^\varepsilon = x$ and $x^{\omega\omega'} = (x^\omega)^{\omega'}$, and furthermore $x \preceq_\varepsilon y$ implies $x^\omega \preceq_\varepsilon y^\omega$ for all $x, y \in \mathbb{D}$ and $\omega, \omega' \in \Omega$.*

The intuition, in the case of relational databases, is as follows. Actions are functions that can modify database entries, i.e., they are functions $f : \mathsf{Const} \cup \mathsf{Null} \to \mathsf{Const} \cup \mathsf{Null}$. They can provide us with additional knowledge (e.g., by replacing a null with a constant), or to the contrary decrease our knowledge (by doing the reverse). The unit of the monoid of functions is the identity function. Then x^ω is the result of applying an action to an object. In case of relational databases, if we have a database D and a function f as above, the result of applying this action is simply $f(D)$. The last condition says that if x is obliviously at most as informative as y, then changing information in both x and y in the same way does not change that.

We extend $\preceq_\varepsilon$ to all elements of Ω by defining $x \preceq_\omega y$ iff $x^\omega \preceq y$. It says that by applying the action ω, object x is at most as informative as y.

Lemma 1. *$x \preceq_\omega y$ and $y \preceq_{\omega'} z$ imply $x \preceq_{\omega\omega'} z$.*

Explanations are special kinds of actions. A relational intuition for actions is replacing some values in relational databases by others. When we replace nulls by other values, this *explains* why one object is more informative than another. In particular, we can replace all nulls by constants, resulting in a complete object, to which no further explanations apply. The following definition captures this distinction between arbitrary actions and explanations.

Definition 2 (Explanation Structure)**.** *An explanation structure $\mathbb{E}$ is a restriction of an action structure $\mathbb{A}$ to a submonoid Σ of Ω, whose elements are called* explanations*, so that for each $x \in \mathbb{D}$, there is $\sigma \in \Sigma$, called x-*complete*, such that $(x^\sigma)^{\sigma'} = x^\sigma$, for all $\sigma' \in \Sigma$. If ε is x-complete, then x is called a* complete *object, and the set of such objects is denoted by $\mathbb{C}$.*

If σ is x-complete, then no other explanation changes x^σ, i.e., σ already explained all the incompleteness of x. A complete object in $\mathbb{C}$ is one for which ε is x-complete, i.e., there is no incompleteness to explain, and $x^\sigma = x$ for all $\sigma \in \Sigma$. In the relational case, explanations are maps $f : \mathsf{Const} \cup \mathsf{Null} \to \mathsf{Const} \cup \mathsf{Null}$ that are the identity on constants (i.e., homomorphisms). For a relational database D, an explanation is D-complete if it is a valuation on D, i.e., it maps D to a complete database. Complete databases, under this definition, will be those without nulls.

Given an explanation structure $\mathbb{E}$, we define the semantics of an object $x \in \mathbb{D}$ with respect to an explanation $\sigma \in \Sigma$ as the set of all complete objects which are at least as informative as x with the help of explanation σ, i.e.,

$$[\![x]\!]^{\mathbb{E}}_\sigma = \{c \in \mathbb{C} \mid x \preceq_\sigma c\}.$$

The semantics of an object is the set of complete objects that can be explained to be at least as informative as x:

$$[\![x]\!]^{\mathbb{E}} = \bigcup_{\sigma \in \Sigma} [\![x]\!]^{\mathbb{E}}_\sigma .$$

We also write $x \preceq y$ if $x \preceq_\sigma y$ for some $\sigma \in \Sigma$. Below we summarize basic properties of these notions.

Proposition 3. • *$\preceq$ is a preorder.*

- *If x^σ is a complete object, then $x^\sigma \in [\![x]\!]^{\mathbb{E}}_\sigma$.*
- *If $x \preceq_\sigma y$, then $[\![y]\!]^{\mathbb{E}}_{\sigma'} \subseteq [\![x]\!]^{\mathbb{E}}_{\sigma\sigma'}$, for each $\sigma' \in \Sigma$.*

Semantics and ordering

It is standard to define information ordering on objects by comparing their semantics, i.e., $x \preceq y$ iff $[\![y]\!] \subseteq [\![x]\!]$. Here

we took a different path and defined both the semantics and the ordering based on the actions of explanations on objects. Does the connection between them continue to hold?

The answer is positive in the presence of *canonical explanations*. These are analogous, in the case of relational databases, to replacing null values with new distinct constants. These are important for defining *naïve evaluation* of queries, when nulls are treated as constants. Crucially such explanations are invertible: one can change these new constants back into nulls, and apply this inverse to other objects that use the same constants. The following definition captures this intuition. First, we say that $\omega \in \Omega$ is *explainable* on x if $x^\omega = x^\sigma$ for some $\sigma \in \Sigma$. That is, the action of ω may not always be an explanation, but on x it is.

Definition 3 (Canonical explanation)**.** *For an object $y \in \mathbb{D}$, a y-complete $\tau \in \Sigma$ is a* canonical explanation for y *if two conditions hold. (1) There is $\tau' \in \Omega$ such that $y^{\tau\tau'} = y$. (2) If $\sigma \in \Sigma$ is a y-complete explanation, and $x \in \mathbb{D}$ satisfies $x \preceq_{\sigma'} y^\sigma$ for some $\sigma' \in \Sigma$, then $\sigma'\tau'$ is explainable on x.*

Theorem 3. *Given an explanation structure $\mathbb{E}$, let $y \in \mathbb{D}$ be such that there is a canonical explanation for it. Then, for $x \in \mathbb{D}$, we have $x \preceq y$ if and only if $[\![y]\!]^{\mathbb{E}} \subseteq [\![x]\!]^{\mathbb{E}}$.* □

Explainable answers to queries

Let $\mathbb{D}$ and $\mathbb{D}'$ be two sets of database objects, and let Σ be a monoid of explanations that gives rise to explanation structures $\mathbb{E}$ and $\mathbb{E}'$ on these sets. We use notations with $'$ for all the concepts in $\mathbb{E}'$, i.e., $\mathbb{C}'$ for complete objects, $\preceq'_\sigma$ and $\preceq'$.

Definition 4 (Explainable answers)**.** *A query Q is a map from $\mathbb{C}$ to $\mathbb{C}'$. Given $x \in \mathbb{D}$ and $a \in \mathbb{D}'$, we say that a is an* explainable candidate answer *to Q on x if for every $c \in [\![x]\!]_\sigma$ we have $a \preceq'_\sigma Q(c)$. A maximal, with respect to $\preceq'$, explainable candidate answer is called an* explainable certain answer.

That is, for explainable candidate answers, the *same* explanation has to account for c being a possible world of x, and for a being at most as informative as the answer in that world. Certain answers, as before, are maximal candidate answers. These behave in the expected way.

Theorem 4. *Let $Q : \mathbb{C} \to \mathbb{C}'$ be a query.*

- *Assume that $x \preceq_\sigma y$ for $x, y \in \mathbb{D}$. If a is an explainable candidate answer to Q on x, then a^σ is an explainable candidate answer to Q on y.*
- *If $\mathsf{glb}_{\preceq'} Q([\![x]\!])$ exists, then it is an explainable certain answer to Q on x.*
- *If $x \preceq y$ and a, b are explainable certain answers to Q on x and y respectively, then $a \preceq' b$.* □

These say that query answers are still more informative on more informative inputs, but extra information in query answers is not arbitrary: instead it is added according to the way in which information was added to the inputs.

6 Explainable certain answers in open and closed worlds

We now look at action and explanation structures for the common OWA and CWA semantics. These are easy to construct. Their elements are relational databases, complete objects are databases without nulls, actions are functions on database entries, and explanations are those actions that preserve constants (i.e., they either assign a constant to a null, or equate two nulls). The only difference between them is the oblivious ordering, which is subset for OWA (to account for adding tuples), and identity for CWA. We then prove that in both of these explanation structures, $\Box_{\text{OWA}}$ and $\Box_{\text{CWA}}$ are exactly the explainable certain answers.

6.1 Explainable answers under OWA

Let ρ be a relational vocabulary. We define action and explanation structures $\mathbb{A}_{\text{OWA}}(\rho)$ and $\mathbb{E}_{\text{OWA}}(\rho)$ for relational databases over ρ. In $\mathbb{A}_{\text{OWA}}(\rho)$:

- the domain $\mathbb{D}(\rho)$ is the set of all databases of vocabulary ρ over $\mathsf{Const} \cup \mathsf{Null}$;
- the oblivious preorder $\preceq_\varepsilon$ is the subset relation $\subseteq$;
- the monoid Ω contains all functions $f : \mathsf{Const} \cup \mathsf{Null} \to \mathsf{Const} \cup \mathsf{Null}$, with the identity function as its unit element, and composition of functions as its binary operation.

The relation $D \preceq_f D'$ is thus given by $f(D) \subseteq D'$.

For the explanation structure $\mathbb{E}_{\text{OWA}}(\rho)$, the submonoid of Ω is $\mathcal{H}$, the set of all homomorphisms, which are maps $h \in \Omega$ such that $h(c) = c$ for every $c \in \mathsf{Const}$. Recall that a homomorphism h is called a *valuation* if $h(\mathsf{Null}) \subseteq \mathsf{Const}$.

Let $[\![\,]\!]^{\mathbb{E}_{\text{OWA}}(\rho)}$ be the semantics defined by $\mathbb{E}_{\text{OWA}}(\rho)$, and let $\preceq$ be the associated preorder, as shown in the previous section. The theorem below summarizes properties of the structures, the preorder, and the semantics.

Theorem 5. *For each relational vocabulary ρ:*

- *$\mathbb{A}_{\text{OWA}}(\rho)$ is an action structure and $\mathbb{E}_{\text{OWA}}(\rho)$ is an explanation structure;*
- *every valuation $v \in \mathcal{H}$ is a complete explanation;*
- *complete objects are exactly ρ-databases without nulls;*
- *$[\![\,]\!]^{\mathbb{E}_{\text{OWA}}(\rho)}$ coincides with $[\![\,]\!]_{\text{OWA}}$;*
- *the preorder $\preceq$ is $\sqsubseteq$ (the existence of a homomorphism);*
- *for $D \in \mathbb{D}(\rho)$, canonical explanations are valuations that are bijections with the range disjoint from $\mathsf{Const}(D)$.*

The existence of canonical explanations implies that $D \preceq D'$ iff $[\![D']\!]^{\mathbb{E}_{\text{OWA}}(\rho)} \subseteq [\![D]\!]^{\mathbb{E}_{\text{OWA}}(\rho)}$, which also follows from [Gheerbrant *et al.*, 2014].

A relational k-ary query Q takes a complete database from $\mathbb{D}(\rho)$ and produces a complete database in $\mathbb{D}(\rho')$, where ρ' consists of a single k-ary relation A (for answer). To define the notion of explainable candidate and certain answers on *arbitrary* databases, we need explanation structures on $\mathbb{D}(\rho)$ and $\mathbb{D}(\rho')$. As mentioned earlier, query answers are always interpreted in $\mathbb{E}_{\text{OWA}}(\rho')$. When query inputs are interpreted in $\mathbb{E}_{\text{OWA}}(\rho)$, we can view Q as a map from complete objects in $\mathbb{E}_{\text{OWA}}(\rho)$ to complete objects in $\mathbb{E}_{\text{OWA}}(\rho')$. This, by Definition 4, gives us the notions of explainable candidate and certain answers *under* OWA. The result below provides their precise characterization.

Theorem 6. *If Q is a k-ary relational query defined on complete databases, and D is an arbitrary database, then*

- *the explainable certain answer under* OWA *is* $\Box_{\text{OWA}}(Q, D)$*;*
- *explainable candidate answers under* OWA *are exactly the subsets of* $\Box_{\text{OWA}}(Q, D)$.

This justifies the use of $\Box_{\text{OWA}}$ and its approximations for finding certain answers under the OWA semantics.

6.2 Explainable answers under CWA

We define action and explanation structures $\mathbb{A}_{\text{CWA}}(\rho)$ and $\mathbb{E}_{\text{CWA}}(\rho)$ just as we did for the OWA case, with a single change: now the oblivious preorder, which we denote by $\preccurlyeq_\varepsilon$ to distinguish it from the OWA case, is the identity (i.e., only $D \preccurlyeq_\varepsilon D$ is true). This reflects the fact that under CWA, no information is added to an incomplete database except by applying a map to its nulls. This extends to relations $\preccurlyeq_h$ for for $h \in \mathcal{H}$, where $D \preccurlyeq_h D'$ iff $D' = h(D)$. Let $\preccurlyeq$ be the union of all such relations, and let $[\![\,]\!]^{\mathbb{E}_{\text{CWA}}(\rho)}$ be the semantics associated with $\mathbb{E}_{\text{CWA}}(\rho)$. The following is an analog of Theorem 5 for CWA.

Theorem 7. *For each relational vocabulary* ρ*:*

- $\mathbb{A}_{\text{CWA}}(\rho)$ *is an action structure and* $\mathbb{E}_{\text{CWA}}(\rho)$ *is an explanation structure;*
- *every valuation* $h \in \mathcal{H}$ *is a complete explanation;*
- *complete objects are exactly* ρ*-databases without nulls;*
- $[\![\,]\!]^{\mathbb{E}_{\text{CWA}}(\rho)}$ *coincides with* $[\![\,]\!]_{\text{CWA}}$*;*
- $D \preccurlyeq D'$ *iff* $h(D) = D'$ *for some homomorphism* h*.*

Let Q be a query from complete databases in $\mathbb{D}(\rho)$ to complete databases in $\mathbb{D}(\rho')$, where again ρ' consists of a single k-ary relation A. We consider inputs as elements of the explanation structure $\mathbb{E}_{\text{CWA}}(\rho)$ and outputs, as before, as elements of the explanation structure $\mathbb{E}_{\text{OWA}}(\rho')$. Then again Definition 4 gives us notions of explainable candidate and certain answers, this time *under* CWA. They can be characterized as follows.

Theorem 8. *If* Q *is a* k*-ary relational query defined on complete databases, and* D *is an arbitrary database, then*

- *the explainable certain answer under* CWA *is* $\Box_{\text{CWA}}(Q, D)$*;*
- *explainable candidate answers under* CWA *are exactly the subsets of* $\Box_{\text{CWA}}(Q, D)$.

Thus, similarly to the OWA case, this result justifies the use of $\Box_{\text{CWA}}$ for finding certain answers under the CWA semantics.

6.3 Justification for intersection-based certain answers

Theorems 6 and 8 allow us to provide justifications for the commonly used intersection-based certain answers, i.e., $\Box^{\cap}_{*}(Q, D) = \bigcap\{Q(D') \mid D' \in [\![D]\!]_*\}$, where $*$ is OWA or CWA. We say that a tuple $\bar{a}$ is an *explainable answer to* Q *on* D *under semantics* $*$ if it belongs to the explainable certain answer under $*$. Then:

Corollary 1. *Let* Q *be a* k*-ary relational query,* D *a database, and* $\bar{c}$ *a* k*-tuple over* $\mathsf{Const}(D)$*. Then* $\bar{c}$ *is an explainable answer to* Q *on* D *under* $*$ *iff* $\bar{c} \in \Box^{\cap}_{*}(Q, D)$*, where* $*$ *is* OWA *or* CWA.

Thus, $\Box^{\cap}_{*}(Q, D)$ is the maximal explainable candidate answer containing only constant tuples, under both OWA and CWA. Also, Corollary 1 implies that for Boolean queries, $\Box^{\cap}_{*}(Q, D)$ is precisely the explainable certain answer, under both OWA and CWA, thereby providing justification for the notions of certainty most commonly found in query answering literature.

7 Conclusions

Our goal was to bridge the gap between two existing ways of defining certainty of query answers over incomplete data. We have done it by introducing the notion of explanations in semantics of incompleteness, and defining certainty by saying that it is the same explanation that has to account for a possible world and the query answer in it. This way, we justified the most commonly used (although hitherto without proper justification) notion of certainty in the literature.

As the outcome of this work, we believe that the right notions of certainty to use, in the case of relational data, are intersection-based; that is $\Box_{\text{CWA}}$ or $\Box_{\text{OWA}}$, depending on the semantics of data. Now these notions come with a proper justification. If one is only interested in tuples of constants, the classical definition of $\bigcap\{Q(D') \mid D' \in [\![D]\!]_*\}$ is the right notion. The informativeness-based definition is still more general and can capture answers that intersection-based notions would miss, but pragmatically one should use intersection-based notions due to their computational properties.

In the future, we plan to extend the ideas of this paper in three ways. One extension is to other semantics of incompleteness (e.g., weaker notions of CWA, cf. [Reiter, 1980; Minker, 1982]). Another direction is to look for notions of explanations as they directly arise in applications such as data integration, data exchange, OBDA, and others. Finally, we would like to consider other data models, in particular graph models and RDF, where the study of querying incomplete data (such as blank nodes) and certain answers has recently received attention [Ahmetaj *et al.*, 2015; Gheerbrant and Fontaine, 2014; Hogan *et al.*, 2014; Nikolaou and Koubarakis, 2016].

Acknowledgments

Part of this work was done while the first author was visiting the University of Edinburgh. The first author was supported by MISE under projects PIUCultura and S2BDW, and by the EU Horizon 2020 programme under project MIREL, grant agreement 690974. The second author is supported by EPSRC grants M025268 and N023056. We are grateful to anonymous referees for their comments.

References

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

[Ahmetaj *et al.*, 2015] Shqiponja Ahmetaj, Wolfgang Fischl, Reinhard Pichler, Mantas Simkus, and Sebastian Skritek. Towards reconciling SPARQL and certain answers. In *WWW*, pages 23–33, 2015.

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

[Arenas *et al.*, 2017] Marcelo Arenas, Elena Botoeva, Egor V. Kostylev, and Vladislav Ryzhikov. A note on computing certain answers to queries over incomplete databases. In *AMW*, 2017.

[Bienvenu and Ortiz, 2015] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, pages 218–307, 2015.

[Buneman *et al.*, 1991] Peter Buneman, Achim Jung, and Atsushi Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science*, 91(1):23–55, 1991.

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

[Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[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.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336:89–124, 2005.

[Gheerbrant and Fontaine, 2014] Amélie Gheerbrant and Gaëlle Fontaine. Querying incomplete graphs with data. In *AMW*, 2014.

[Gheerbrant *et al.*, 2014] Amélie Gheerbrant, Leonid Libkin, and Cristina Sirangelo. Naïve evaluation of queries over incomplete databases. *ACM Trans. Database Syst.*, 39(4):31:1–31:42, 2014.

[Hell and Nešetřil, 2004] Pavol Hell and Jaroslav Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[Hogan *et al.*, 2014] Aidan Hogan, Marcelo Arenas, Alejandro Mallea, and Axel Polleres. Everything you always wanted to know about blank nodes. *J. Web Sem.*, 27:42–69, 2014.

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

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: a theoretical perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[Libkin, 2011] Leonid Libkin. Incomplete information and certain answers in general data models. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 59–70, 2011.

[Libkin, 2016] Leonid Libkin. Certain answers as objects and knowledge. *Artificial Intelligence*, 232:1–19, 2016.

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

[Minker, 1982] Jack Minker. On indefinite databases and the closed world assumption. In *CADE*, pages 292–308, 1982.

[Nikolaou and Koubarakis, 2016] Charalampos Nikolaou and Manolis Koubarakis. Querying incomplete information in RDF with SPARQL. *Artificial Intelligence*, 237:138–171, 2016.

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

[Reiter, 1980] Raymond Reiter. Equality and domain closure in first-order databases. *Journal of the ACM*, 27(2):235–249, 1980.

[Rosser and Schoenfeld, 1962] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(2):64–94, 1962.

[Rounds, 1991] Bill Rounds. Situation-theoretic aspects of databases. In *Situation Theory and Applications*, volume 26 of *CSLI*, pages 229–256. 1991.

[ten Cate and Dalmau, 2015] Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In *ICDT*, pages 161–176, 2015.