# Data Exchange and Incomplete Information

Leonid Libkin
University of Toronto and University of Edinburgh
libkin@cs.toronto.edu and libkin@inf.ed.ac.uk

*Dedicated to the memory of Alberto Mendelzon*

## ABSTRACT

Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target, and answering queries over target instances in a way that is semantically consistent with the information in the source. Theoretical foundations of data exchange have been actively explored recently. It was also noticed that the standard certain answers semantics may behave in very odd ways.

In this paper I explain that this behavior is due to the fact that the presence of incomplete information in target instances has been ignored; in particular, proper query evaluation techniques for databases with nulls have not been used, and the distinction between closed and open world semantics has not been made. I present a concept of target solutions based on the closed world assumption, and show that the space of all solutions has two extreme points: the canonical universal solution and the core, well studied in data exchange. I show how to define semantics of query answering taking into account incomplete information, and show that the well-known anomalies go away with the new semantics. The paper also contains results on the complexity of query answering, upper approximations to queries (maybe-answers), and various extensions.

## 1. Introduction

Data exchange is the problem of finding an instance of a target schema, given an instance of a source schema and a specification of the relationship between the source and the target. This is an old problem that has received renewed attention over the past few years. Commercial strength systems have been built [24], and theoretical foundations have been developed recently, starting with the influential papers by Fagin, Kolaitis, Miller, and Popa [10, 11]. A survey of the area was presented in the most recent PODS invited talk [18].

Here we revisit the basics of relational data exchange, as described in [10, 11, 18]. A *data exchange setting* is a triple $(\sigma, \tau, \Sigma)$ where $\sigma$ and $\tau$ are *source* and *target* schemas (relational vocabularies), and $\Sigma$ is a set of *source-to-target dependencies (STDs)*,

$$\psi_\tau(\bar{x}, \bar{z}) \;\; :- \;\; \varphi_\sigma(\bar{x}, \bar{y}),$$

where $\varphi_\sigma$ is a formula over $\sigma$ and $\psi_\tau$ is a conjunction of atomic $\tau$-formulae (here we follow the notation of [6]: the above is shorthand for FO formulae over $\sigma \cup \tau$ of the form $\forall\bar{x}\forall\bar{y}\ \big(\varphi_\sigma(\bar{x}, \bar{y}) \ \rightarrow\ \exists\bar{z}\ \psi_\tau(\bar{x}, \bar{z})\big)$).

Consider, for example, a schema $\sigma$ with two ternary relations $R_1$ and $R_2$ and a schema $\tau$ with one ternary relation $V$. Suppose $R_1$ and $R_2$ are databases of two different airlines, having attributes (departure_city, arrival_city, flight#). Assume these airlines merge and they want to offer the same flights as the first airline, and for each route of the second airline they want to make sure that the cities remain reachable with at most one change of planes. Assume that $V$ has attributes (departure_city, arrival_city, aircraft_type). Then this situation is captured by the following STDs:

$$\begin{array}{rcl} V(x_1, x_2, z) & :- & R_1(x_1, x_2, y) \\ V(x_1, z_1, z), V(z_1, x_2, z') & :- & R_2(x_1, x_2, y) \end{array}$$

If we have a source instance $S$, then our goal is to find a target instance $T$ and answer queries written over $\tau$ in a way that is semantically consistent with the information in $S$. The main contributions of [10, 11] and others were as follows:

- They defined the concept of a solution (an instance $T$ such that $(S, T) \models \Sigma$) and the concept of a *universal* solution (these are solutions that in a certain sense are more general than arbitrary ones).
- Two universal solutions of particular importance were studied: the *canonical* (universal) solution, and the *core* of universal solutions. For example, if $R_1$ has tuples `(JFK,CDG,001)` and `(JFK,CDG,003)`, and $R_2$ has one tuple `(JFK,LHR,005)`, then the canonical solution would have tuples (JFK,CDG,$\perp_1$), (JFK,CDG,$\perp_2$), corresponding to the first STD (that ensures there are flights from JFK to CDG), and (JFK,$\perp_3, \perp_4$), and ($\perp_3$,LHR,$\perp_5$), corresponding to


*PODS'06,* June 26–28, 2006, Chicago, Illinois, USA.

the second STD (that ensures LHR is reachable from JFK with one stop). The core, instead of two tuples corresponding to the JFK–CDG flights, will have just one, `(JFK,CDG,⊥)`.

Elements $\perp_i$ here are *null values*: we know that some values need to be put in the target instance, but we do not yet have their values.

- The semantics of answering queries was defined in terms of *certain answers*, and it was shown that the canonical solution and the core are good for answering conjunctive queries (with inequalities).

These results provided the basis for extensions dealing with rewritability, query answering, schema composition, algorithmic issues, other data models, etc [4, 6, 12, 13, 23]. However, the main concepts of solutions and query answering semantics still appear quite ad hoc. Furthermore, they give rise to some well known anomalies of the standard certain-answers semantics as introduced in [10]. Here we recall perhaps the strangest one [4]. A data exchange setting is *copying* if $\sigma = \{R_1, \ldots, R_n\}$, $\tau = \{R'_1, \ldots, R'_n\}$ and $\Sigma$ consists of STDs

$$R'_i(\bar{x}) \ :\!-\ R_i(\bar{x})$$

(that is, $R_i$ and $R'_i$ have the same arity). In other words, it says: copy each $R_i$ into $R'_i$. And yet in this setting one can define FO queries over the target that cannot be answered in FO at all, if the semantics of [10] is used!

It is natural to assume that the reason for such anomalies lies in some basic problems with the setting such as the definition of solutions and query answering semantics. In fact [11] tried to remedy this partially by introducing a different, rather ad hoc, certain-answers semantics which avoided some of the problems mentioned above, but did exhibit some anomalous behavior as well. This led Kolaitis and others [7] to ask: "what is so sacred about this [certain answers] semantics?"

My goal is to answer this question, and, more generally, re-think the basics of data exchange. My main point is that while target instances are tables with nulls, techniques for handling data with incomplete information have been completely ignored in data exchange: certain answers are defined in [10] with respect to sets of solutions as if each were a relation *without* null values.

But it is well known that answering queries over databases with incomplete information must be done with care: not treating nulls as such leads to semantically incorrect answers [2, 17, 22, 30]. Hence, we define the notions of solutions and query answering in data exchange treating solutions as databases with nulls.

Once the view of solutions as instances with nulls is adopted, we still need to address two more issues.

1. *Closed vs Open World Assumption* (CWA vs OWA): this is the standard issue in databases with incomplete information that needs to be clarified before the semantics can be defined [26, 17]. CWA states that the database is closed to adding new facts except those consistent with one of the incomplete tuples in it; OWA opens the database to such facts.

   In my view (explained below), CWA is the right assumption in data exchange, although most previous papers defined OWA-based semantics.

2. *Query answering semantics*. Indeed there is nothing sacred about the certain answers semantics; more than 25 years ago, Lipski [22] already suggested using both *certain* and *maybe* answers in the context of partial information, as providing lower and upper approximations to query results. Even more advanced forms of approximations were proposed [9, 8, 15, 21] but here we use the basic lower and upper ones.

Before outlining the main contributions, let me add two comments on points 1) and 2) above. First, the reason for adopting CWA is that in data exchange we need to move data from source to target based on STDs. Hence, query answering must be based on such data, and not data that can later be added to target instances. This is the CWA approach.

Second, for query answering, the notions of certain and maybe answers can be applied at two different levels: at the level of each individual solution (which is a table with nulls), and at the level of all solutions, to combine individual answers into one. The former has always been ignored in the context of data exchange, while for the latter only the lower approximation (certain answers) has been considered [10].

Below is a summary of the paper.

1. I present a small number of requirements of what it means to be a good CWA solution. With these properly formalized, CWA solutions are characterized, and their space is shown to have two extreme points: a unique largest solution, that happens to be the canonical solution of [10], and a unique smallest solution, that happens to be the core [11].

2. Combining certain/maybe answers at the levels of individual solutions and all solutions gives us four reasonable semantics, and these are characterized as certain/maybe answers over the core/canonical solution.

   Thus, the problem of query answering in data exchange is reduced to the well-studied problem of query answering over databases with incomplete information, while these databases are the core and the canonical solution, which we know well how to construct.

   It is further shown that the new semantics does not exhibit the anomalous behavior explained above.

3. With the problem of query answering in data exchange reduced to that of finding certain or maybe answers over canonical solutions and cores, we study its complexity. Certain answers-based semantics are shown to be coNP-complete, and maybe answers-based semantics are NP-complete (in the size of the source instance). In special cases, such as conjunctive queries, our semantics fits in nicely with that of [11, 10] (which concentrated on conjunctive queries). We also look at representations for maybe-answers to queries.

4. Two extensions are considered: the OWA semantics (which is shown to be undecidable even for simple FO queries), as well as adding keys and foreign keys to data exchange settings.

**Organization**. Notations are given in Section 2. We present the CWA-based notion of solutions in Section 3. In Section 4 we define the semantics of query answering, and in Sections 5 and 6 we study its complexity and the special cases of monotone and conjunctive queries. In Section 7 we give a representation mechanism for maybe-answers. In Section 8 we consider extensions (OWA semantics and target constraints). In Section 9 we discuss practical applicability of these results. Section 10 points out some directions for future work. Due to space limitations, only a few proofs are given. Complete proofs are in the full version available from the author.

## 2. Notations

### Data exchange settings

The following definitions are standard [10, 11, 4, 18]. A *data exchange setting* is a triple $(\sigma, \tau, \Sigma)$ where $\sigma$ and $\tau$ are the source and the target schemas respectively (that is, sets of relation names with associated arities), and $\Sigma$ is a set of *source-to-target dependencies (STDs)* of the form $\psi_\tau(\bar{x}, \bar{z}) :\!-\, \varphi_\sigma(\bar{x}, \bar{y})$, where $\varphi_\sigma$ is a first-order (FO) formula over vocabulary $\sigma$, and $\psi_\tau$ is a conjunction of atomic $\tau$-formulae. We assume that $\sigma$ and $\tau$ have no relation names in common, and that elements of the source instance come from a countably infinite domain $\mathsf{Const}$ (in data exchange terminology, called *constants*). New elements created in target instances are null values, and we assume that nulls come from a countably infinite domain $\mathsf{Null}$ disjoint from $\mathsf{Const}$. Elements of $\mathsf{Const}$ are typically denoted by lowercase letters, and elements of $\mathsf{Null}$ by $\perp$ with sub/superscripts.

There are some minor differences between this setting and [10, 11]. Frist, as in [4], we let $\varphi_\sigma$'s be FO formulae (as opposed to [10, 11] where they were restricted to conjunctive queries). Also, as in [4], we shall assume that one can distinguish nulls from constants. One way of enforcing this is to assume that we have a unary predicate testing for nulls, like the `IS NULL` condition in SQL.

Given a data exchange setting $(\sigma, \tau, \Sigma)$ and a source instance $S$, a target instance $T$ is called a *solution* for $S$ if $(S, T) \models \Sigma$. More precisely, for every $\psi(\bar{x}, \bar{z}) :\!-\, \varphi(\bar{x}, \bar{y})$ in $\Sigma$ and every pair of tuples $\bar{a}, \bar{b}$ such that $\varphi(\bar{a}, \bar{b})$ holds in $S$, there is a tuple $\bar{c}$ such that $\psi(\bar{a}, \bar{c})$ holds in $T$.

Given two instances $T, T'$ over $\tau$, a *homomorphism*[1] $h : T \to T'$ is a mapping $h$ from $\mathsf{Null}$ to $\mathsf{Null}$ such that for each relation symbol $R$ in $\tau$ and each tuple $\bar{t}$ in the interpretation $R^T$ of $R$ in $T$, the tuple $h(\bar{t})$ is in $R^{T'}$, the interpretation of $R$ in $T'$. (Of course by $h(\bar{t})$ we mean the tuple obtained from $\bar{t}$ by replacing each null $\perp$ in it by $h(\perp)$, and leaving the constants intact.)

We say that $T'$ is a subinstance of $T$ if for each relation symbol $R \in \tau$ and its interpretations $R^T$ and $R^{T'}$, we have $R^{T'} \subseteq R^T$. In this case we also write $T' \subseteq T$.

A 1-1 homomorphism is just a renaming of nulls. We say that $T$ is *contained* in $T'$ if there is a renaming of nulls $h$ such that $h(T) \subseteq T'$, and we shall identify instances which are the same up to renaming of nulls.

### Canonical universal solution and the core

Two solutions play a special role in data exchange: the canonical universal solution [10], and the core [11]. We start with the definition of the canonical universal solution, following the presentation of [4]. Let $(\sigma, \tau, \Sigma)$ be a data exchange setting, and $S$ a source instance. For each STD $\psi(\bar{x}, \bar{z}) :\!-\, \varphi(\bar{x}, \bar{y})$ and for each pair of tuples $\bar{a}, \bar{b}$ such that $\varphi(\bar{a}, \bar{b})$ holds in $S$, create fresh tuples of distinct nulls $\bar{\perp} = \bar{\perp}_{(\varphi,\psi,\bar{a},\bar{b})}$ (so that $|\bar{\perp}| = |\bar{z}|$) and put tuples in the target so that $\psi(\bar{a}, \bar{\perp})$ holds. We recall that $\psi$ is a conjunction of atomic formulae. The result is the *canonical (universal) solution* $\mathrm{C}\scriptstyle{\mathrm{AN}}\displaystyle\mathrm{S}\scriptstyle{\mathrm{OL}}\displaystyle^{(\sigma,\tau,\Sigma)}(S)$. Typically the data exchange setting is understood from the context, and we write just $\textsc{CanSol}(S)$.

For example, let $\sigma = \{E\}, \tau = \{R\}$, with both $E$ and $R$ binary, and let $\Sigma$ contain $R(x, z) :\!-\, E(x, y)$. Then if $E$ has tuples $\{(a, b_1), (a, b_2)\}$, then the canonical solution would have tuples $\{(a, \perp_1), (a, \perp_2)\}$ in relation $R$.

A subinstance $T'$ of $T$ is a *core* of $T$ if there is a homomorphism $h : T \to T'$ but no homomorphism from $T$ to a proper subinstance of $T'$ [16, 11]. Cores always exist and even though an instance may have multiple cores, they are all isomorphic (that is, the same up to renaming of nulls) [16]. Thus we can speak of the core of $T$. We shall denote the core of $\textsc{CanSol}^{(\sigma,\tau,\Sigma)}(S)$ by $\textsc{Core}^{(\sigma,\tau,\Sigma)}(S)$ (again, omitting $(\sigma, \tau, \Sigma)$ if it is understood from the context).

In the previous example, both $\{(a, \perp_1)\}$ and $\{(a, \perp_2)\}$ are cores; of course they are isomorphic so we can say that $\{(a, \perp)\}$ is the core of $\{(a, \perp_1), (a, \perp_2)\}$.

For arbitrary structures, computing cores is intractable [16], but for a fixed setting $(\sigma, \tau, \Sigma)$, $\textsc{Core}(S)$ can be constructed in time polynomial in $S$ [11][2].

### Relations with incomplete information

We very briefly review some standard definitions [2, 17]. A database instance with incomplete information is an instance

[1] This is a stricter notion than the one used in [10, 18] and others. We discuss this at the end of the section.

[2] Results of [11] refer to the setting where homomorphism may map nulls to constants. However if we, in linear time, modify all STDs so that they also collect nulls in a unary relation, then homomorphisms from $\textsc{CanSol}(S)$, even in the setting of [11], map nulls to nulls, and complexity bounds of [11] apply.

whose domain is a subset of Const $\cup$ Null. Nulls are treated as "unknown" (as opposed to "nonexistent") values [30]. A *valuation* is a partial map $v$ : Null $\to$ Const. Given an instance $T$ with incomplete information and a valuation $v$ defined on all the nulls present in $T$, let $v(T)$ stand for the instance of the same schema over Const in which every null $\perp$ present in $T$ is replaced by $v(\perp)$. We then define

$$Rep(T) = \{v(T) \mid v \text{ is a valuation}\},$$

where $v$ ranges over all valuations defined on all nulls present in $T$. Note that $Rep(T)$ is a potentially infinite object (e.g., if $T$ has a unary relation $U$ with one tuple $\{\perp\}$, then $Rep(T) = \{\{c\} \mid c \in \mathsf{Const}\}$).

In order to evaluate a query $Q$ on an instance with nulls (where $Q$ comes from a language that works on instances without nulls, e.g., a fragment of FO or relational algebra), one normally considers $\{Q(R) \mid R \in Rep(T)\}$. To represent this set (even for an FO query $Q$), one needs rather complicated *conditional tables* [17]. Instead of exact representation, one may use lower and upper approximations, namely *certain* and *maybe* answers, defined by:

$$\Box Q(T) = \bigcap\{Q(R) \mid R \in Rep(T)\}$$

$$\Diamond Q(T) = \bigcup\{Q(R) \mid R \in Rep(T)\}.$$

That is, certain answers $\Box Q(T)$ contain tuples present in the answer no matter what values are assigned to nulls, and maybe answers $\Diamond Q(T)$ contain tuples present in at least one answer to $Q$ for some assignment of values to nulls. Notice that $\Box Q(T)$ is a finite object (since it is contained in $Q(v(T))$ for every valuation $v$), but $\Diamond Q(T)$ may well be infinite and thus some finite representation of it needs to be found.

*Remark.* In [10, 18] and others, homomorphisms are mappings from Null to Null $\cup$ Const. Since we assume that there is a unary predicate that plays the role of `IS NULL`, such predicate has to be preserved by homomorphisms of structures and thus homomorphisms under those assumptions should map nulls to nulls. Note that each homomorphism $h$ in the sense of [10, 11] can be factored as $v \circ g$, where $g$ is a homomorphism as we define it, and $v$ is a partial valuation. Hence all the same complete instances will be arising as $Rep(T)$, regardless of the definition, but the definition we use helps separate concepts related to homomorphisms and valuations. From the technical point of view, our definition does not impose any restrictions, as we could drop the assumption that homomorphisms preserve nulls and obtain exact analogs of all the results given here.

## 3. Data exchange: solutions

The main idea of solutions under the CWA is that every fact in the target instance is directly justified by the source instance and the STDs. We formulate this – first, at a rather informal level – as follows:

1. For all nulls, their presence is justified by the source instance and the STDs.
2. Justifications for nulls should not be overused: that is, each justification for producing a null does not generate multiple nulls.
3. Each fact in the target instance is justified by the source instance and the STDs. That is, solutions should not invent new facts compared to what can be inferred if no additional assumptions are made about the nulls.

We now formalize these notions. Fix a data exchange setting $(\sigma, \tau, \Sigma)$ where $\Sigma$ is a collection

$$\{\psi_i(\bar{x}_i, \bar{z}_i) \;\text{:-}\; \varphi_i(\bar{x}_i, \bar{y}_i) \mid 1 \le i \le m\}.$$

Let $S$ be a source instance. A justification for a null over $S$ consists of:

1. an STD $\psi_i(\bar{x}_i, \bar{z}_i) \text{ :- } \varphi_i(\bar{x}_i, \bar{y}_i)$ in $\Sigma$,
2. a tuple $(\bar{a}, \bar{b})$ witnessing its body (i.e., $\varphi_i(\bar{a}, \bar{b})$), and
3. a position in the head, among $\bar{z}_i$, corresponding to a null to be introduced.

Formally, *justification* for a null over $S$ is a quadruple $(i, \bar{a}, \bar{b}, k)$ where $i \le m$, $\bar{a}$ and $\bar{b}$ are tuples such that $S \models \varphi_i(\bar{a}, \bar{b})$, and $k \le |\bar{z}_i|$. We let $\mathcal{J}(S)$ stand for the set of all justifications (if the setting is understood from the context).

In the example from the introduction, one possible justification is a tuple $\big(1, (\mathtt{JFK,CDG}), (\mathtt{001}), 1\big)$ which says that for the first rule, tuples $\bar{a} = (\mathtt{JFK,CDG}), \bar{b} = (\mathtt{001})$, satisfy the body of the rule, and thus a null corresponding to the position of $z$ in $V(x_1, x_2, z)$ (which happens to be 1, since there is only one variable that is not present in the body), a null must be produced.

We want each null in the target instance to be associated with a justification for it. Furthermore, in the spirit of CWA, we do *not* want the same justification to justify different nulls. We do, however, allow different justifications to justify the same null.

This is captured as follows. Let $\Pi$ be a partition of $\mathcal{J}(S)$ with blocks $B_1, \ldots, B_l$. We then create a target instance $T_\Pi(S)$ in which justifications from each block $B_j$ are all represented by the same null $\perp_j$, $j = 1, \ldots, l$. That is, for each $\psi_i(\bar{x}_i, \bar{z}_i) \text{ :- } \varphi_i(\bar{x}_i, \bar{y}_i)$ in $\Sigma$ and for each $\bar{a}, \bar{b}$ such that $S \models \varphi_i(\bar{a}, \bar{b})$, consider all the justifications

$$(i, \bar{a}, \bar{b}, 1), \ldots, (i, \bar{a}, \bar{b}, |\bar{z}_i|)$$

Define a tuple of nulls $\bar{\nu} = (\nu_1, \ldots, \nu_{|\bar{z}_i|})$ so that $\nu_j = \perp_p$ if $(i, \bar{a}, \bar{b}, j)$ is in $B_p$, the $p$th block of the partition. Then add tuples to $T_\Pi(S)$ to satisfy $\psi_i(\bar{a}, \bar{\nu})$. (Recall that $\psi_i$ is a conjunction of atoms.)

We call instances of the form $T_\Pi(S)$, where $\Pi$ is a partition on the set of justifications $\mathcal{J}(S)$, *CWA-presolutions*. It is immediate from the definition that each CWA-presolution $T_\Pi(S)$ is a solution in the sense of [10]: that is, $(S, T_\Pi(S)) \models \Sigma$.

Consider the finest partition $\Pi_{\rm sng}$ in which each block is a singleton $\bot_{(i,\bar a,\bar b,k)}$ associated with the justification $(i,\bar a,\bar b,k)$. Then $T_{\Pi_{\rm sng}}(S)$ is the canonical solution $\textsc{CanSol}(S)$. In general, if we define a homomorphism $h_\Pi : \{\bot_{(i,\bar a,\bar b,k)}\} \to \{\bot_1,\ldots,\bot_l\}$ by

$$h_\Pi(\bot_{(i,\bar a,\bar b,k)}) = \bot_p \;\Leftrightarrow\; \bot_{(i,\bar a,\bar b,k)} \in B_p,$$

then $h_\Pi(\textsc{CanSol}(S)) = T_\Pi(S)$.

Next we deal with the third requirement. Notice that CWA-presolutions can make certain assumptions equating nulls, and thus may generate new associations between elements in the target. Some of these facts are already there even if we do not make any assumptions about equating nulls, that is, they are in $T_{\Pi_{\rm sng}}(S) = \textsc{CanSol}(S)$. Some, however, may be genuinely new: for example, if $\textsc{CanSol}(S)$ has tuples $(a,\bot_1)$ and $(\bot_2,b)$, then equating $\bot_1$ and $\bot_2$ will tell us that there is a path of length 2 between $a$ and $b$. The CWA should prohibit inventing facts based on equating nulls unless such nulls are made equal by the STDs (in other words, they are equal in $T_{\Pi_{\rm sng}}(S)$).

We formalize this is as follows. A *fact* is a formula $f(\bar a)$, where $\bar a$ is over Const, of the form $\exists \bar z\ \alpha(\bar a,\bar z)$, where $\alpha$ is a conjunction of $\tau$-atoms. It is satisfied in a target instance $T$ if there is a tuple of nulls $\bar\bot$ such that $\alpha(\bar a,\bar\bot)$ is true. Then solutions are presolutions in which every true fact can be inferred without equating nulls (unless STDs force them to be equal).

**Definition 3.1**. *A CWA-presolution $T$ is called a* CWA-solution *if every fact true in $T$ is also true in $T_{\Pi_{sng}}(S) = \textsc{CanSol}(S)$.*

*The set of all CWA-solutions for $S$ is denoted by $[\![S]\!]^{(\sigma,\tau,\Sigma)}_{CWA}$, or just $[\![S]\!]_{CWA}$ if the data exchange setting is clear from the context.*

Each (pre)solution $T$ represents a set $Rep(T)$ of target instances without incomplete information. Notice that $Rep(T) \subseteq Rep(T_{\Pi_{\rm sng}}(S)) = Rep(\textsc{CanSol}(S))$, and hence by imposing extra conditions on presolutions we do not lose any instances that are represented by them.

Almost directly from the definitions we obtain:

**Lemma 3.2**. *A CWA-presolution $T$ for $S$ is a CWA-solution iff there exists a homomorphism $T \mapsto \textsc{CanSol}(S)$.*

In particular, this implies that, in the terminology of [10], CWA-solutions are *universal* solutions, that is, they have homomorphisms into all other solutions.

A more important property is that there exists a unique minimal CWA-solution, namely, the core. Recall that "contains" means up to renaming of nulls.

**Lemma 3.3**. *If $T$ is a CWA-solution for $S$ in a data exchange setting $(\sigma,\tau,\Sigma)$, then $\textsc{Core}(S)$ is contained in $T$. Moreover, $\textsc{Core}(S)$ itself is a CWA-solution.*

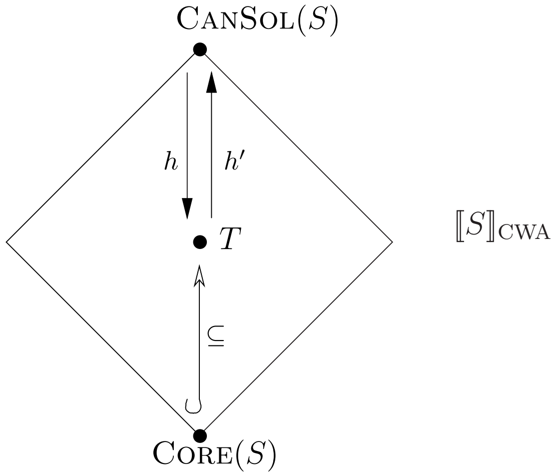


**Figure 1: A representation of $[\![S]\!]_{\mathbf{CWA}}$**

Combining, we have the following description of CWA-solutions.

**Theorem 3.4**. *A target instance $T$ is a CWA solution for $S$ iff the following are true:*

1. *$T$ is a homomorphic image of $\textsc{CanSol}(S)$;*
2. *there is a homomorphism $T \to \textsc{CanSol}(S)$;*
3. *$T$ contains $\textsc{Core}(S)$.*

Thus, the space of all CWA solutions contains two unique extreme points: the minimal solution, that is contained in all others, which is the core, and the maximal one, of which every solution is a homomorphic image, namely the canonical solution. This is illustrated in Figure 1. Note also the inclusions

$$Rep(\textsc{Core}(S)) \subseteq Rep(T) \subseteq Rep(\textsc{CanSol}(S))$$

for every CWA-solution $T$. We also remark that there could be CWA-solutions $T$ which are not contained in $\textsc{CanSol}(S)$. For instance, let $\tau$ have a ternary relation $R$, and let $\textsc{CanSol}(S)$ consist of two tuples $(a,\bot_1,\bot_2)$ and $(a,\bot_3,\bot_4)$. Then $\{(a,\bot_1,\bot'),(a,\bot_3,\bot')\}$ is a CWA-solution that is not contained in $\textsc{CanSol}(S)$.

It follows from [4, 10, 11] that both $\textsc{CanSol}(S)$ and $\textsc{Core}(S)$ can be computed in polynomial time, for a fixed data exchange setting. As we shall see in the next section, query answering under CWA can be done using just these two solutions.

## 4. Query answering: semantics

As mentioned earlier, previous approaches to query answering in data exchange ignored both the possibility of using *maybe* answers, and more importantly the fact that solutions themselves are databases with incomplete information, and thus appropriate techniques should be used for querying them.

For each individual solution $T$ and a query $Q$ over it, we have the lower and the upper approximations to the answer, which are given by $\Box Q(T) = \bigcap\{Q(R) \mid R \in Rep(T)\}$, and $\Diamond Q(T) = \bigcup\{Q(R) \mid R \in Rep(T)\}$. Furthermore, answers to queries over different solutions can be combined in two different ways: we can either look for certain answers which are true for all solutions (this is the semantics of [10] and others), or tuples true in some solutions. These combinations give rise to four different semantics for query answering, depending on how we combine possibility/certainty at the level of each solution, and all solutions. These are defined formally below. We are assuming a data exchange setting $(\sigma, \tau, \Sigma)$ and a source instance $S$.

- The *certain answers semantics*: we collect tuples that belong to the answer no matter which solution is chosen and how nulls are instantiated, i.e.

  $$\bar{t} \in \mathsf{certain}_\Box(Q, S) \;\Leftrightarrow\; \forall T \in [\![S]\!]_{\mathrm{CWA}} : \; \bar{t} \in \Box Q(T)$$

- The *potential certain answers semantics*: we collect tuples that appear as certain answers for at least one CWA-solution. In other words,

  $$\bar{t} \in \mathsf{certain}_\Diamond(Q, S) \;\Leftrightarrow\; \exists T \in [\![S]\!]_{\mathrm{CWA}} : \; \bar{t} \in \Box Q(T)$$

- The *persistent maybe answers semantics*: we collect tuples that appear as maybe answers for all CWA-solutions. In other words,

  $$\bar{t} \in \mathsf{maybe}_\Box(Q, S) \;\Leftrightarrow\; \forall T \in [\![S]\!]_{\mathrm{CWA}} : \; \bar{t} \in \Diamond Q(T)$$

- The *maybe answers semantics*: we collect tuples that appear as maybe answers for at least one CWA-solution. In other words,

  $$\bar{t} \in \mathsf{maybe}_\Diamond(Q, S) \;\Leftrightarrow\; \exists T \in [\![S]\!]_{\mathrm{CWA}} : \; \bar{t} \in \Diamond Q(T)$$

While these seem to be rather diverse, there are simple connections between these semantics and characterizations in terms of canonical solutions and cores: to evaluate a query $Q$ under one of those semantics, one has to answer either $\Box Q$ or $\Diamond Q$ on either $\textsc{CanSol}(S)$ or $\textsc{Core}(S)$. Since we know how to construct canonical solutions and cores [10, 11, 24, 13], the problem of answering queries in data exchange is thus reduced to the classical and well studied problem of answering queries in databases with incomplete information [2, 3, 17].

**Theorem 4.1**. *The following characterizations of the semantics hold:*

$$\begin{array}{rcl} \mathsf{certain}_\Box(Q,S) & = & \Box Q(\textsc{CanSol}(S)) \\ \mathsf{certain}_\Diamond(Q,S) & = & \Box Q(\textsc{Core}(S)) \\ \mathsf{maybe}_\Box(Q,S) & = & \Diamond Q(\textsc{Core}(S)) \\ \mathsf{maybe}_\Diamond(Q,S) & = & \Diamond Q(\textsc{CanSol}(S)). \end{array}$$

*Proof.* Notice that for every CWA-solution $T$ we have $Rep(\textsc{Core}(S)) \subseteq Rep(T) \subseteq Rep(\textsc{CanSol}(S))$ as a consequence of Theorem 3.4. Therefore,

$$\begin{array}{rlcl} & \Box Q(\textsc{CanSol}(S)) & = & \bigcap_{R \in Rep(\textsc{CanSol}(S))} Q(R) \\ \subseteq & \Box Q(T) & = & \bigcap_{R \in Rep(T)} Q(R) \\ \subseteq & \Box Q(\textsc{Core}(S)) & = & \bigcap_{R \in Rep(\textsc{Core}(S))} Q(R), \end{array}$$

and likewise

$$\begin{array}{rlcl} & \Diamond Q(\textsc{CanSol}(S)) & = & \bigcup_{R \in Rep(\textsc{CanSol}(S))} Q(R) \\ \supseteq & \Diamond Q(T) & = & \bigcup_{R \in Rep(T)} Q(R) \\ \supseteq & \Diamond Q(\textsc{Core}(S)) & = & \bigcup_{R \in Rep(\textsc{Core}(S))} Q(R). \end{array}$$

Hence,

$$\mathsf{certain}_\Box(Q,S) = \bigcap_{T \in [\![S]\!]_{\mathrm{CWA}}} \Box Q(T) = \Box Q(\textsc{CanSol}(S))$$

$$\mathsf{certain}_\Diamond(Q,S) = \bigcup_{T \in [\![S]\!]_{\mathrm{CWA}}} \Box Q(T) = \Box Q(\textsc{Core}(S))$$

$$\mathsf{maybe}_\Box(Q,S) = \bigcap_{T \in [\![S]\!]_{\mathrm{CWA}}} \Diamond Q(T) = \Diamond Q(\textsc{Core}(S))$$

$$\mathsf{maybe}_\Box(Q,S) = \bigcup_{T \in [\![S]\!]_{\mathrm{CWA}}} \Diamond Q(T) = \Diamond Q(\textsc{CanSol}(S)),$$

as claimed. □

One can use Theorem 4.1 to establish the following relationship between these semantics.

**Corollary 4.2**. *The following inclusions hold:*

$$\begin{array}{rlcl} & \mathsf{certain}_\Box(Q,S) & \subseteq & \mathsf{certain}_\Diamond(Q,S) \\ \subseteq & \mathsf{maybe}_\Box(Q,S) & \subseteq & \mathsf{maybe}_\Diamond(Q,S). \end{array}$$

For the reader familiar with the semantics of [10, 11] (that is, the OWA-based certain answers semantics, and the universal solutions semantics), we remark that both produce subsets of $\mathsf{certain}_\Box(Q, S)$.

### New semantics and query rewriting/answering anomalies

Some of the well-known problems in data exchange – nonexistence of rewritings and anomalies in query answering – disappear with the new notion of CWA-solutions and the new semantic functions.

If we are given a data exchange setting $(\sigma, \tau, \Sigma)$, a source $S$, and some semantic function that associates an answer $\mathsf{answer}(Q, S)$ to a query $Q$ over the target, then a *rewriting* for $Q$ over some specific target instance $T$ is a query $Q'$ such that $Q'(T) = \mathsf{answer}(Q, S)$. Typically one considers rewritings over the canonical solution or the core. Results of [4, 10] show that with the semantics of [10] (or its modification proposed in [11]), for some simple FO queries rewritings may not exist (even in copying data exchange settings). However, for our semantics, rewritings always exist for *arbitrary* queries: they are either $\Box Q$ or $\Diamond Q$, over either the canonical solution or the core, as Theorem 4.1 shows. That is, one obtains rewritings for arbitrary queries by using proper techniques for evaluating queries in databases with incomplete information.

Let us now come back to the example of copying data exchange settings. Consider such a setting $(\sigma, \tau, \Sigma)$ with $\sigma = \{R_1, \ldots, R_n\}$, $\tau = \{R'_1, \ldots, R'_n\}$ and $\Sigma = \{R'_i(\bar{x}) \coloneq R_i(\bar{x}) \mid 1 \leq i \leq n\}$. In other words, each $R_i$

is copied into $R_i'$. A known anomaly of query answering in data exchange is that under the semantics of [10, 11], FO queries may not be rewritable in copying settings [4]. But notice that, in a copying setting, $[\![S]\!]_{\mathrm{CWA}}^{(\sigma,\tau,\Sigma)} = \{S\}$, for each source instance $S$. Hence,

$$\begin{array}{rcccl} & \mathsf{certain}_{\Box}(Q,S) & = & \mathsf{certain}_{\Diamond}(Q,S) & \\ = & \mathsf{maybe}_{\Box}(Q,S) & = & \mathsf{maybe}_{\Diamond}(Q,S) & = \quad Q(S), \end{array}$$

for every $Q$, as expected. In addition, the rewriting of $Q$ is $Q$ itself, as it should be in a copying setting. Thus, the correct choice of semantics resolves one of the most unpleasant anomalies in query answering in data exchange.

Another anomaly disappears under the new semantics. It is known that if $Q$ is a Boolean query, then under the semantics of [10] either the answer to $Q$ over *all* instances is false, or the answer to $\neg Q$ over *all* instances is false: that is, the answer to $Q$ cannot be true in some databases and false in others. The previous example shows that this anomaly does not arise under the new semantics.

Some anomalies of the standard certain-answers semantics of [10] were noticed in [11], who proposed a different, universal-solutions semantics. The reasoning behind it was that since universal solutions (which have homomorphisms into all other solutions) are "preferable" in data exchange, perhaps one should only consider answers that are true in all such solutions, that is,

$$\bigcap\{Q(R) \mid R \text{ is a universal solution}\}.$$

While the reason behind this definition is somewhat ad hoc, these solutions have some nice properties: in particular every existential query is FO-rewritable over the core under the universal-solutions semantics [11]. Nonetheless, in a slight modification of copying settings, this semantics behaves as badly as the semantics of [10]. Namely, consider an extension of a copying setting with a domain predicate, that is, together with all the STDs $R'(\bar{x}) \coloneq R(\bar{x})$ we have new STDs $D(x) \coloneq R(\ldots, x, \ldots)$ that create a unary relation that collects all the elements of the active domain. Under the CWA semantics, $[\![S]\!]_{\mathrm{CWA}}$ is simply $S$ together with the active domain of $S$ as an interpretation of $D$, and hence every FO query is trivially rewritable (since the active domain is definable). However, it is known [4] that in such a setting one can construct simple $\exists^*\forall$ FO queries that are not rewritable over the canonical solution or the core under the universal-solutions semantics.

## 5. Query answering: complexity

Next we consider the data complexity of query answering in data exchange [4, 10, 11]. Suppose we are given a semantic function $\mathsf{answer}(Q, S)$ that, for a source instance $S$ and a query $Q$ over the target schema, produces an answer to $Q$ (it will be one of the four semantic functions in the previous section). Then the *data complexity* of $\mathsf{answer}$ is the complexity of the language $\{enc(S)\#enc(\bar{t}) \mid \bar{t} \in \mathsf{answer}(Q, S)\}$, where $enc$ is some suitable encoding of instances and tuples.

| Semantics | data complexity |
|---|---|
| $\mathsf{certain}_{\Box}$ | coNP-complete |
| $\mathsf{certain}_{\Diamond}$ | coNP-complete |
| $\mathsf{maybe}_{\Box}$ | NP-complete |
| $\mathsf{maybe}_{\Diamond}$ | NP-complete |

**Figure 2: Data complexity**

**Theorem 5.1**. *The data complexity of FO queries for the semantics* $\mathsf{certain}_{\Box}$*,* $\mathsf{certain}_{\Diamond}$*,* $\mathsf{maybe}_{\Box}$*, and* $\mathsf{maybe}_{\Diamond}$ *is as shown in Figure 2.*

In fact the upper bounds only require that the data complexity of $Q$ itself be polynomial. Thus we obtain:

**Corollary 5.2**. *If $\mathcal{L}$ is a query language that contains FO and has polynomial-time data complexity, then the data complexity of $\mathcal{L}$-queries for the semantics* $\mathsf{certain}_{\Box}$*,* $\mathsf{certain}_{\Diamond}$*,* $\mathsf{maybe}_{\Box}$*, and* $\mathsf{maybe}_{\Diamond}$ *is as shown in Figure 2.*

Of course it has long been known that the complexity of computing certain (maybe) answers for FO queries is coNP-complete (NP-complete, respectively) [1, 3], in the size of a table $T$. However, here we measure the complexity of answering queries on $\textsc{CanSol}(S)$ or $\textsc{Core}(S)$, in terms of the size of $S$. Thus, in the proof of Theorem 5.1, we provide hardness examples that, unlike those in [1, 3], arise as $\textsc{CanSol}(S)$ or $\textsc{Core}(S)$ for some fixed data exchange setting.

## 6. Query answering: monotone and positive queries

We now turn to the case of conjunctive queries, which was most heavily studied in the context of data exchange [10, 11, 23]. First recall that [10, 11] and others follow the naive approach to evaluation of queries on tables with nulls. We call this naive evaluation function $\mathrm{naive\_eval}(Q, T)$; it simply treats nulls as they were usual values (in other words, it assumes that the domain of the database comes from $\mathsf{Const} \cup \mathsf{Null}$, and thus the equality predicate is available on the entire domain; in particular, two nulls are equal if they are just symbolically the same null). This corresponds precisely to query evaluation over *naive tables* [2, 17].

Based on this naive evaluation, [10, 11] proposed a semantics for evaluating conjunctive queries which happened to coincide with their notion of certain answers. Define $T_{\downarrow}$ as the instance $T$ from which all tuples containing nulls have been removed. Then the evaluation function for conjunctive queries from [10, 11] was

$$\mathrm{CQ\_eval}(Q, S) \;=\; \mathrm{naive\_eval}(Q, \textsc{CanSol}(S))_{\downarrow}.$$

It turns out that this is precisely what two of the semantics we studied here do for the class of positive relational algebra queries (that is, $\{\sigma, \pi, \bowtie, \cup\}$ queries in which selection predicates are positive Boolean combinations of equalities).

**Proposition 6.1**. *If $Q$ is a monotone query, then* $\mathsf{certain}_\Box(Q,S) = \mathsf{certain}_\Diamond(Q,S)$. *Furthermore, if $Q$ is a positive relational algebra query, then*

$$\mathsf{certain}_\Box(Q,S) \;=\; CQ\_eval(Q,S).$$

*Proof.* From Corollary 4.2 we have $\mathsf{certain}_\Box(Q,S) \subseteq \mathsf{certain}_\Diamond(Q,S)$ for arbitrary queries, so we need to prove the converse, that is, $\Box Q(\textsc{Core}(S)) \subseteq \Box Q(\textsc{CanSol}(S))$, for an arbitrary $S$, if $Q$ is monotone.

Consider an arbitrary valuation $v$ on the nulls of $\textsc{CanSol}(S)$, and let $R = v(\textsc{CanSol}(S))$. Let $\iota : \textsc{Core}(S) \to \textsc{CanSol}(S)$ be the natural embedding of $\textsc{Core}(S)$ into $\textsc{CanSol}(S)$, and let $v_\iota$ be a valuation on the nulls of $\textsc{Core}(S)$ which is a composition of $\iota$ and $v$. Then clearly $v_\iota(\textsc{Core}(S)) \subseteq v(\textsc{CanSol}(S))$, and hence $Q(v_\iota(\textsc{Core}(S))) \subseteq Q(v(\textsc{CanSol}(S)))$.

Let $\mathrm{Val}(T)$ be the set of all valuations on an instance $T$. Then

$$\begin{aligned}
\Box Q(\textsc{Core}(S)) \;&=\; \bigcap_{v' \in \mathrm{Val}(\textsc{Core}(S))} Q(v'(\textsc{Core}(S)))\\
&\subseteq \bigcap_{v \in \mathrm{Val}(\textsc{CanSol}(S))} Q(v_\iota(\textsc{Core}(S)))\\
&\subseteq \bigcap_{v \in \mathrm{Val}(\textsc{CanSol}(S))} Q(v(\textsc{CanSol}(S)))\\
&= \;\Box Q(\textsc{CanSol}(S))
\end{aligned}$$

Thus, $\mathsf{certain}_\Box(Q,S) = \mathsf{certain}_\Diamond(Q,S)$ for a monotone $Q$.

Finally, $\textsc{CanSol}(S)$ is a naive table [17], and it is known that naive tables form a strong representation system for the positive fragment of relational algebra [17] (the results of [17] apply to both open and closed world semantics), and hence $\mathsf{certain}_\Box(Q,S) = \Box Q(\textsc{CanSol}(S)) = \mathrm{CQ\_eval}(Q,S)$. $\Box$

For maybe-answers, even for quantifier-free conjunctive queries we may have $\mathsf{maybe}_\Box(Q,S) \neq \mathsf{maybe}_\Diamond(Q,S)$. For example, it is easy to find a data exchange setting with a single target relation $R$ and an instance $S$ so that $\textsc{CanSol}(S) = \{(a,\perp_1),(a,\perp_2)\}$. Then $\textsc{Core}(S) = \{(a,\perp)\}$. If $Q(x,y,z) = R(x,y) \wedge R(x,z)$, then $\mathsf{maybe}_\Box(Q,S) = \Diamond Q(\textsc{Core}(S)) \subsetneq \Diamond Q(\textsc{CanSol}(S)) = \mathsf{maybe}_\Diamond(Q,S)$. The same proof as for Proposition 6.1 shows that $\mathsf{maybe}_\Box(Q,S) = \mathsf{maybe}_\Diamond(Q,S)$ for every anti-monotone query $Q$.

## 7. Representing maybe answers

While $\Box Q(T)$ is a finite object, $\Diamond Q(T)$ is inherently infinite: even if $Q$ is the identity query id, we have $\Diamond\mathsf{id}(T) = \bigcup\{R \mid R \in \mathit{Rep}(T)\}$. So it is natural to ask how maybe answers can be represented; in particular, how are they going to be presented to the user who wants an upper approximation to a query.

Various representations of maybe answers are possible, e.g. by viewing databases as logical theories in the spirit of [27] or by using conditional tables [17], but all of them are rather hard to use in practical query evaluation algorithms, and are furthermore not very intuitive. So instead we propose an approach that seems reasonable from the point of view of user getting an understandable result of a maybe-query.

Let $\bar{t}$ be a tuple in $T$ over $\mathsf{Const} \cup \mathsf{Null}$, and let $v$ be a *strict* valuation on $\bar{t}$, that is, a 1-to-1 mapping from the set of nulls in $\bar{t}$ into $\mathsf{Const}$ such that no value of $v$ occurs as a constant in $T$. We then let $\mathit{Rep}_s(\bar{t} \mid T) = \{v(\bar{t})\}$, where $v$ ranges over strict valuations. Next, if we have an instance $T$ with nulls and a query $Q$, then we call a table $W$ a *fair representation of* $\Diamond Q(T)$ if

$$\bigcup\{\mathit{Rep}_s(\bar{t} \mid T) \;\mid\; \bar{t} \in W\} \;=\; \Diamond Q(T).$$

Intuitively, tuples in a fair representation of $\Diamond Q(T)$ give null/constant patterns of tuples that appear in $\Diamond Q(T)$. For example, if a pair $(a_1,\perp_1,a_2,\perp_2)$ is in a fair representation of $\Diamond Q(T)$, then for every pair $(a_1',a_2')$ of constants not present in $T$, the tuple $(a_1,a_1',a_2,a_2')$ is in $Q(R)$ for some $R \in \mathit{Rep}(T)$.

We will now show that fair representations exist, and are of polynomial size. However, constructing them may be hard: we know that for FO queries $Q$, the problem of checking whether $\Diamond Q(T)$ is nonempty (with $T$ as the input), is NP-complete [3], and the problem remains NP-complete if $T$ is the canonical solution or the core (see Theorem 5.1). However, for positive relational algebra queries, the problem is tractable.

For these results, we shall use the standard notion of *generic* queries [2]: these are queries that commute with permutations of the domain. Queries computable in standard languages such as relational algebra, datalog, etc., are generic.

**Theorem 7.1**.

1. *For each generic query $Q$, there is a polynomial $p_Q$ such that a fair representation of $\Diamond Q(T)$ of size at most $p_Q(|T|)$ exists, for every $T$.*
2. *If the data complexity of a generic query $Q$ is NP (PSPACE), then a fair representation of $\Diamond Q(T)$ can be constructed in NP (respectively, PSPACE).*
3. *If $Q$ is a positive relational algebra query, then a fair representation of $\Diamond Q(T)$ can be constructed in polynomial time, in the size of $T$.*

The proof of this theorem also gives a slightly more general version of item 2): if the data complexity of $Q$ is $\mathrm{NTIME}(n^k)$, and $Q$ produces a relation of arity $\leq k$, then a fair representation of $\Diamond Q(T)$ can be constructed in $\mathrm{NTIME}(n^k)$ (and a similar result is true for space bounds).

## 8. Extensions

### Open World Semantics

So far we concentrated on CWA, but one can also define an OWA semantics if there is no requirement that facts true in solutions be also true if we do not impose conditions equating nulls, and if we open target instances to new tuples. This way, each OWA target instance would contain a CWA-presolution; hence we can describe OWA solutions as those that contain a homomorphic image of the canonical solution. The class of all such solutions is denoted by $[\![S]\!]_{\text{OWA}}$ (if the setting $(\sigma, \tau, \Sigma)$ is understood).

With each query evaluation semantics $\mathsf{answer}$ we associate a new semantics $\mathsf{answer}^{\text{OWA}}$ defined just as $\mathsf{answer}$ except that $[\![\ ]\!]_{\text{OWA}}$ is used in the place of $[\![\ ]\!]_{\text{CWA}}$.

There are many known cases when going from CWA to OWA makes a decidable problem undecidable (see, e.g., [1, 29]), and here we have another example: functions $\mathsf{answer}^{\text{OWA}}$ are not computable if $\mathsf{answer}$ is one of the four semantic functions we studied for CWA. The proof (typically for a proof of undecidability for OWA) is by reduction from FO validity in the finite.

**Proposition 8.1**. *If* $\mathsf{answer}$ *is one of the four semantic functions* $\mathsf{certain}_\Box$, $\mathsf{certain}_\Diamond$, $\mathsf{maybe}_\Box$, *or* $\mathsf{maybe}_\Diamond$, *then the problem of checking whether* $\mathsf{answer}^{OWA}(Q, S)$ *evaluates to true, for an FO Boolean query Q and a source instance S, is undecidable, even in copying data exchange settings.*

**Target Constraints**

Sometimes constraints on the target are also imposed. This changes the certain or maybe answers semantics for solutions $T$, as we are only interested in instances in $Rep(T)$ that satisfy target constraints. In other words, if we have a set of target constraints $\Sigma_t$, then the semantics of $\Diamond Q(T)$ or $\Box Q(T)$ is defined over $Rep_{\Sigma_t}(T) = \{v(T) \mid v(T) \models \Sigma_t\}$, where $v$ ranges over valuations on $T$.

We now show that even the simplest constraints complicate matters significantly. Assume that $\Sigma_t$ contains only *keys and foreign keys*. Then, for a data exchange setting with target constraints $(\sigma, \tau, \Sigma, \Sigma_t)$ we consider the problem EXISTENCE-OF-SOLUTIONS which has a source instance $S$ as an input, and outputs 'yes' if there is a CWA-solution $T$ such that $Rep_{\Sigma_t}(T) \neq \emptyset$.

**Proposition 8.2**. *If target constraints contain keys and foreign keys, then the problem* EXISTENCE-OF-SOLUTIONS *is NP-complete.*

The hardness is witnessed by a data exchange setting with just two STDs, two keys and three inclusion constraints. If inclusion constraints are dropped, it easy to show that the problem becomes tractable.

## 9. Practical considerations

Data exchange is an area where systems work is ahead of theoretical investigation: data exchange systems existed for a while (and are being worked on) [24, 25, 28], with theoretical foundations arriving a few years later. In fact the main goal of theoretical papers on data exchange is to offer insights into the semantics of query answering, and to justify – or suggest changes to – algorithms implemented in real-life systems. In this short section I would like to give a few remarks on applicability of results shown here.

- Of the four semantics proposed here, it is probably the two extreme ones – $\mathsf{certain}_\Box(Q, S)$ and $\mathsf{maybe}_\Diamond(Q, S)$ – that are useful for providing approximation to query answers. As $\mathsf{certain}_\Box(Q, S) = \Box Q(\text{CANSOL}(S))$ and $\mathsf{maybe}_\Diamond(Q, S) = \Diamond Q(\text{CANSOL}(S))$, this strongly suggests materializing the canonical solution $\text{CANSOL}(S)$ rather than the core (in addition, it is easier to compute $\text{CANSOL}(S)$ from the algorithmic point of view).

- Once a query $Q$ is issued, certain answers to $Q$ should be computed and given to the user. Note that the issue of non-rewritability goes away with the closed-world semantics, so we simply compute $\Box Q$ over the materialized $\text{CANSOL}(S)$. Since $\text{CANSOL}(S)$ is a naive table [2], this is easily done for the positive fragment of relational algebra.

- If certain answers are not sufficient for the user, maybe answers should be computed to provide an upper approximation. No new materialization of the target is required, they can be computed over $\text{CANSOL}(S)$. Efficiency (outside the class of positive relational algebra queries) is another matter and is discussed in the next section.

## 10. Future work

There is still much left to do. We have only briefly looked into dealing with target constraints, but they should be explored further. In particular it would be nice to identify tractable cases of query answering in the presence of target constraints. One may also look at techniques of database repairs and query answering in inconsistent databases [5] if the canonical solution fails to satisfy some of the target constraints, although this will involve three different levels at which incomplete information appears (nulls in targets, repairs, potentially multiple target solutions). Furthermore, target instances are databases with nulls, and thus dealing with constraints such as FDs and IDs over them requires additional care [19, 20].

We reduced query answering in data exchange to query answering over naive tables, which may be intractable for queries outside of the positive fragment of relational algebra. It would be nice to find ways to overcome this; for example, by finding easily constructible and fairly large subsets of certain answers. As for maybe answers, one should look into designing fast incremental algorithms for constructing them, so that answers would be produced tuple by tuple.

While the CWA appears to be more realistic assumption than the OWA in data exchange, the OWA should not be dismissed completely. One can definitely envision a situation when certain facts are more reasonable to interpret under the OWA, although a decision to do so requires additional *semantic information* about target instances. Consider, for example, an STD $R(x, z) :- S(x, y)$. If there is a tuple $(a, b)$ in $S$, the most reasonable decision seems to be putting a tuple $(a, \perp)$ into the target instance. But, on the other hand, if we have an additional knowledge that the relationship between the two attributes of $R$ is one-to-many, it then seems reasonable to "open" that closed world target and consider solutions in which multiple tuples $(a, \perp_i)$ are permitted. Opening CWA databases was looked at previously [14] and I believe the right alternative to the OWA approach in data exchange is using the CWA semantics of this paper, and opening facts that need to be opened. But how to do it remains to be investigated.

Finally, data exchange techniques have recently been looked at in the XML context [6]. There is no clearly defined concept of a good solution in that case (as the analog of the canonical solution may fail to satisfy schema specifications), nor well-defined techniques for answering queries with incomplete information. Thus defining a proper semantics for solutions and query answering for XML remains open.

**Acknowledgments** Discussions with Marcelo Arenas during the early stages of this project were very helpful. I thank Pablo Barceló, Ronald Fagin, and anonymous reviewers for their comments. The author was supported by grants from NSERC, CITO, and PREA.

## 11. References

[1] S. Abiteboul, O. Duschka. Complexity of answering queries using materialized views. In *PODS 1998*, pages 254–263.

[2] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases*, Addison Wesley, 1995.

[3] S. Abiteboul, P. Kanellakis and G. Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78 (1991), 159–187.

[4] M. Arenas, P. Barceló, R. Fagin, L. Libkin. Locally consistent transformations and query answering in data exchange. In *PODS 2004*, pages 229–240.

[5] M. Arenas, L. Bertossi, J. Chomicki. Consistent query answers in inconsistent databases. In *PODS'99*, pages 68–79.

[6] M. Arenas, L. Libkin. XML data exchange: consistency and query answering. In *PODS'05*, pages 13–24.

[7] L. Bertossi, J. Chomicki, P. Godfrey, Ph. Kolaitis, A. Thomo, and C. Zuzarte. Exchange, integration, and consistency of data: report on ARISE/NISR workshop. *SIGMOD Record* 34 (2005), 87–90.

[8] P. Buneman, S. Davidson and A. Watters. A semantics for complex objects and approximate answers. *Journal of Computer and System Sciences* 43(1991), 170–218.

[9] P. Buneman, A. Jung, A. Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science* 91(1991), 23–55.

[10] R. Fagin, Ph. Kolaitis, R. Miller, L. Popa. Data exchange: semantics and query answering. In *ICDT'03*, pp. 207–224. Full version in *Theoretical Computer Science*, 336 (2005), 89–124.

[11] R. Fagin, Ph. Kolaitis, L. Popa. Data exchange: getting to the core. In *PODS'03*, pages 90–101. Full version in *ACM TODS* 30 (2005), 90–101.

[12] R. Fagin, Ph. Kolaitis, L. Popa, W.C. Tan. Composing schema mappings: second-order dependencies to the rescue. *PODS 2004*, pages 83-94

[13] G. Gottlob. Computing cores for data exchange: new algorithms and practical solutions. In *PODS'05*, pages 148–159.

[14] G. Gottlob and R. Zicari. Closed world databases opened through null values. In *VLDB'88*, pages 50–61.

[15] C. Gunter. The mixed powerdomain. *Theoretical Computer Science* 103 (1992), 311–334.

[16] P. Hell and J. Nešetřil. The core of a graph. *Discrete Math.* 109 (1992), 117–126.

[17] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of ACM* 31(1984), 761–791.

[18] Ph. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS 2005*.

[19] M. Levene, G. Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science* 206 (1998), 283–300.

[20] M. Levene, G. Loizou. Null inclusion dependencies in relational databases. *Information and Computation*, 136 (1997), 67–108.

[21] L. Libkin. Models of approximation in databases. *Theoretical Computer Science*, 190 (1998), 167–210.

[22] W. Lipski. On semantic issues connected with incomplete information in databases. *ACM Trans. Database Systems* 4 (1979), 262–296.

[23] A. Madry. Data exchange: on the complexity of answering queries with inequalities. *Information Processing Letters* 94 (2005), 253–257.

[24] R. Miller, M. Hernandez, L. Haas, L. Yan, C. Ho, R. Fagin, L. Popa. The Clio project: managing heterogeneity. *SIGMOD Record* 30 (2001), 78–83.

[25] L. Popa, Y. Velegrakis, R. Miller, M. Hernández, R. Fagin. Translating web data. In *VLDB 2002*, pages 598–609.

[26] R. Reiter. On closed world databases. In *"Logic and Databases"*, H. Gallaire and J. Minker eds, Plenum Press, 1978, pages 55–76.

[27] R. Reiter. Towards a logical reconstruction of relational database theory. In: *"On Conceptual Modeling"* (M. Brodie and J. Schmidt eds.), Springer Verlag, 1984, pages 163–189.

[28] N. Shu, B. Housel, R. Taylor, S. Ghosh, V. Lum. EXPRESS: a data extraction, processing, and restructuring system. *ACM TODS* 2 (1977), 134–174.

[29] M. Vardi. On the integrity of databases with incomplete information. In *PODS'86*, pages 252–266.

[30] C. Zaniolo. Database relations with null values. *Journal of Computer and System Sciences* 28 (1984), 142–166.