Logical Foundations of Relational Data Exchange

Pablo Barceló
Department of Computer Science, University of Chile
pbarcelo@dcc.uchile.cl

1 Introduction

Data exchange has been defined as the problem of taking data structured under a *source* schema and materializing an instance of a *target* schema that reflects as accurately as possible the source data [19]. In the last years, the need for data exchange applications has increased, particularly due to the proliferation of web data in various formats (relational, XML, RDF, etc) and the emergence of e-business applications that need to communicate data yet remain autonomous. Responding to this demand, commercial data exchange systems have been built recently [12].

Even though data exchange is an old and common data management problem, its most foundational aspects had not been studied until very recently. There are two reasons for this. First, most of the early research on databases concentrated on the stand-alone relational model, and much less on interoperability, integration, and exchange. Second, there were no solid foundation, nor even a proper formal model, for the problem of data exchange. Such a model was finally proposed in 2003 by Fagin, Kolaitis, Miller and Popa [19], and was quickly adopted as the right model for data exchange. A survey on the topic has already appeared in the premier conference on database theory, PODS, [30], and two workshops on exchange and integration of data have already been held [10, 39].

This survey is organized as follows. In Section 2, we present the basics of data exchange. One of the goals of data exchange is materializing a target instance that is consistent both with the source data and the specification of the relationship between the source and the target. Such a target instance is called a *solution* for the given source data. The work of Fagin et al. [19] identified a class of solutions, called *universal* solutions, with good properties for data exchange. We introduce the class of universal solutions in Section 3. In Section 4, we study the problem of the materialization of universal solutions. In particular, we show that there is a meaningful class of data exchange settings for which universal solutions are guaranteed to exist, if a solution exists at all, and, if that is the case, then a universal solution can always be constructed in polynomial time. Also in Section 4 we study the core of the universal solutions, which happens to be the smallest universal solution.

Database Principles Column. Column editor: Leonid Libkin, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK. E-mail: libkin@inf.ed.ac.uk.

The notion of query answering in data exchange is presented in Section 5. After defining the semantics, which is based on the notion of certain answers (that is, those tuples that appear in answers for all solutions), we show that the problem of evaluating queries in data exchange becomes tractable for a relevant class of queries; namely, unions of conjunctive queries, which conform the logical core of the SQL query language. In Section 6 we show that other semantics can be meaningfully applied in data exchange. In particular, we present a semantics based on the class of universal solutions and a semantics based on the notions of closed-world assumption and incomplete information. Finally, we present in Section 7 a brief survey of other work in data exchange. Concluding remarks are in Section 8.

2 Data Exchange Settings

A *schema* $\mathbf{R}$ is a finite sequence $\langle R_1, \ldots, R_m \rangle$ of relation symbols, with each R_i having a fixed arity $n_i > 0$. An *instance I of* $\mathbf{R}$ assigns to each relation symbol R_i of $\mathbf{R}$ a finite n_i-ary relation $I(R_i)$. The *domain* of instance I is the set of all elements that occur in any of the relations $I(R_i)$. It is often convenient to define instances by simply listing the tuples attached to the corresponding relation symbols. Sometimes we use the notation $R(\bar{t}) \in I$ instead of $\bar{t} \in I(R)$, and call $R(\bar{t})$ a *fact* of I. Finally, a *dependency* over $\mathbf{R}$ is a sentence in some logical formalism, typically first-order logic (FO), with which we assume familiarity.

Given schemas $\mathbf{S} = \langle S_1, \ldots, S_m \rangle$ and $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$, with no relation symbols in common, we denote by $\langle \mathbf{S}, \mathbf{T} \rangle$ the schema $\langle S_1, \ldots, S_m, T_1, \ldots, T_n \rangle$. Further, if I is an instance of $\mathbf{S}$ and J is an instance of $\mathbf{T}$, then (I, J) denotes an instance K of $\langle \mathbf{S}, \mathbf{T} \rangle$ such that $K(S_i) = I(S_i)$ and $K(T_j) = J(T_j)$, for each $i \in [1, m]$ and $j \in [1, n]$.

Definition 2.1 (Data exchange setting) *A* data exchange setting $\mathcal{M}$ *is a triple* $(\mathbf{S}, \mathbf{T}, \Sigma)$*, where* $\mathbf{S}$ *and* $\mathbf{T}$ *are disjoint schemas,* $\mathbf{S}$ *is called the* source *schema,* $\mathbf{T}$ *is called the* target *schema, and* Σ *is a finite set of dependencies over* $\langle \mathbf{S}, \mathbf{T} \rangle$.

Instances of $\mathbf{S}$ are called *source* instances, while instances of $\mathbf{T}$ are called *target* instances. It is usual in the data exchange literature to assume the existence of two disjoint and infinite set of values that populate instances. One is the set of *constants*, denoted by Const, and the other one

is the set of *nulls*, denoted by Var. The domain of a source instance is always contained in Const, while the domain of a target instance is contained in Const ∪ Var. We usually denote constants by lowercase letters $a, b, c, \ldots$, while nulls are denoted by symbols $\perp, \perp_1, \perp_2, \ldots$

In data exchange, the target instances that are consistent with both the source instance and the specification Σ are called solutions. Formally, given a source instance I, we say that the target instance J is a *solution for I under $\mathcal{M}$*, or simply a solution for I if $\mathcal{M}$ is clear from the context, if (I, J) satisfies every sentence in Σ.

Admitting the full expressive power of FO as a language for specifying dependencies in data exchange, easily yields to undecidability of some fundamental problems, like checking for the existence of solutions [18]. Thus, it is customary in the data exchange literature [19, 20, 30] to restrict the study to the class of settings $\mathcal{M}$, such that Σ can be split into two sets Σ_{st} and Σ_t that satisfy the following:

1. Σ_{st} consists of a set of *source-to-target* dependencies (stds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y} \psi_{\mathbf{T}}(\bar{x}, \bar{y}))$, where $\varphi_{\mathbf{S}}(\bar{x})$ and $\psi_{\mathbf{T}}(\bar{x}, \bar{y})$ are conjunctions of atomic formulas in $\mathbf{S}$ and $\mathbf{T}$, respectively; and
2. Σ_t is the set of *target* dependencies. It is the union of a set of *tuple-generating* dependencies (tgds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y}))$, where $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{y})$ are conjunctions of atomic formulas in $\mathbf{T}$, and a set of *equality-generating* dependencies (egds), i.e. dependencies of the form $\forall \bar{x}\,(\varphi(\bar{x}) \rightarrow x_i = x_j)$, where $\varphi(\bar{x})$ is a conjunction of atomic formulas in $\mathbf{T}$, and x_i, x_j are variables in $\bar{x}$.

From now on, and unless stated otherwise, we assume all data exchange settings to be of the form $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\Sigma = \Sigma_{st} \cup \Sigma_t)$, for Σ_{st} a finite set of stds and Σ_t a finite set of tgds and egds. The intuition behind the different components of these settings is the following: Source-to-target dependencies in Σ_{st} are a tool for specifying which conditions on the source imply a condition on the target. But from a different point of view, one can also see them as a tool for specifying how source data gets translated into target data. In addition, the translated data must satisfy usual database constraints. This is represented by means of the target dependencies in Σ_t. It is important to notice that the data exchange settings described above are not restrictive from a database point of view. Indeed, tuple-generating dependencies together with equality generating dependencies precisely capture the class of *embedded* dependencies [17]. And the latter class contains all relevant dependencies that appear in relational databases, e.g. it contains functional and inclusion dependencies, among others.

Next example shows two interesting phenomena regarding solutions in data exchange. First, that solutions for a given source instance are not necessarily unique. Second, that there are source instances that have no solutions.

Example 2.2 *Consider a data exchange setting $\mathcal{M}$ in which $\mathbf{S}$ consists of the binary relations M and N, $\mathbf{T}$ consists of the ternary relation P and the binary relation Q, $\Sigma_t = \emptyset$ and Σ_{st} consists of the following stds (we implicitly assume universal quantification in front of all dependencies):*

$$\begin{aligned} M(x,y) &\rightarrow \exists w \exists z (P(x,y,z) \wedge Q(w,z)), \\ N(x,y) &\rightarrow \exists u P(x,y,u). \end{aligned}$$

Suppose we have a source instance $I = \{M(a,b), N(a,b)\}$. Since the stds in Σ_{st} do not completely specify the target, solutions for I are not unique up to isomorphism. For instance, one solution is:

$$J = \{P(a,b,\perp_1), P(a,b,\perp_2), Q(\perp_3, \perp_1)\},$$

where $\perp_1, \perp_2, \perp_3$ are values in Var *(nulls). Another solution, but with no nulls, is $J' = \{P(a,b,a), Q(b,a)\}$. Further, it is not hard to see that any other target instance that contains J or J' is a solution for I. Thus, I admits infinitely many solutions.*

Consider now the setting $\mathcal{M}'$ that extends $\mathcal{M}$ by adding the following egd to Σ_t: $P(x,y,z) \rightarrow x = y$. Then I has no solution under $\mathcal{M}$. Indeed, if there was at least one such solution J, then the first dependency in Σ_{st} implies that there is a fact of the form $P(a,b,z)$ in J, while the egd implies that that the constants a and b are equal, which is a contradiction. □

3 Universal Solutions

In this section we introduce a certain class of solutions that exhibit good properties for data exchange: the *universal* solutions. Notice, in Example 2.2, that the solution J' seems to be less general than J. This is because J' assumes that the values that witness the existentially quantified variables z and u, in the first and second std of Σ_{st}, respectively, are the same (namely, the constant a). It also assumes that the value that witnesses the existentially quantified variable w is the constant b. But none of these assumptions is part of the specification. On the other hand, solution J contains exactly what the specification requires. Since one of the basic problems in data exchange is materializing a target instance given a source instance, in this case one would like to materialize a solution like J rather than solution J'.

In order to give a precise mathematical definition of which solutions are the most general, we first have to define what a homomorphism between data exchange instances is. Let J and J' be two instances over the target schema $\mathbf{T}$ with values in Const ∪ Var. A *homomorphism* $h : J \rightarrow J'$ is a mapping from the domain of J into the domain of J', that is the identity on constants, and such that $\bar{t} = (t_1, \ldots, t_n) \in J(R)$ implies $h(\bar{t}) = (h(t_1), \ldots, h(t_n))$ is in $J'(R)$ for all $R \in \mathbf{T}$.

Definition 3.1 (Universal solutions) *Let J be a solution for I. Then J is a* universal *solution for I if for every solution J' for I, there exists a homomorphism $h : J \to J'$.*

Example 3.2 *Solution J' in Example 2.2 is not universal, since there is no homomorphism $h : J' \to J$. But it can be shown that J is a universal solution.* □

A universal solution is more general than an arbitrary solution because it can be homomorphically mapped into that solution. Furthermore, as we will see later, universal solutions possess good properties that justify materializing them (as opposed to arbitrary solutions). Unfortunately, universal solutions are not a general phenomenon. Indeed, [19] proved that there is a setting $\mathcal{M}$ and a source instance I, such that I has at least one solution under $\mathcal{M}$ but has no universal solutions. Thus, it is necessary to impose extra conditions on dependencies if one wants to make sure that the existence of solutions implies the existence of universal solutions. We study this issue in detail in the next section.

4 Materializing Solutions

One of the goals in data exchange is materializing a solution that reflects as accurately as possible the source data. Unfortunately, even the most basic problem of checking for the existence of solutions is undecidable:

Theorem 4.1 *[31] There exists a data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$, such that the problem of deciding for a given source instance I, whether I has a solution under $\mathcal{M}$, is undecidable.*

Thus, one would like to restrict the class of dependencies allowed in data exchange settings, in such a way that it satisfies the following: (C1) the existence of solutions implies the existence of universal solutions; (C2) checking the existence of solutions is a decidable (ideally, tractable) problem; and (C3) for every source instance that has a solution, at least one universal solution can be computed (hopefully, in polynomial time).

The main algorithmic tool that the data exchange community has applied in order to check for the existence of solutions is the well-known *chase* procedure [37, 9], that was originally designed to reason about the implication problem for data dependencies. In data exchange, the chase is used as a tool for constructing a universal solution for a given source instance. The basic idea is the following. The chase starts with the source instance I, and then triggers every dependency in $\Sigma_{st} \cup \Sigma_t$ that is being violated, as long as this process is applicable. In doing so, the chase may fail (if firing an egd forces two constants to be equal) or it may never terminate (for instance, in some cases when the set of tgds is *cyclic*). It follows from [19], that if the chase fails then I has no solution; and that if the chase does not fail and terminates, then the resulting target instance is guaranteed to be a universal solution for I. Nothing can be said in the case when the chase does not terminate. The next example shows an application of the chase procedure.

Example 4.2 *Let $\mathcal{M}$ be the setting such that the source schema consists of the binary relation E, the target schema consists of the binary relations G and L, and Σ_{st} consists of the std $\varphi = E(x, y) \to G(x, y)$. Assume first that Σ_t consists of the tgd $\theta_1 = G(x, y) \to \exists z L(y, z)$, and let I be the source instance $E(a, b)$. The chase starts by firing φ and, thus, by populating the target with the fact $G(a, b)$. In a second stage, the chase realizes that θ_1 is being violated, and thus, θ_1 is triggered. The target is then extended with a fact $L(b, \perp)$, where $\perp$ is a fresh null value. At this stage, no dependency is being violated, and thus, the chase stops with result $J = \{G(a, a), L(b, \perp)\}$. It is not hard to see that J is a universal solution for I.*

Assume now that Σ_t is extended with the tgd $\theta_2 = L(x, y) \to \exists z G(y, z)$. Clearly, J does not satisfy θ_2 and the chase triggers this tgd. This means that a fact $G(\perp, \perp_1)$ is added to the target, where $\perp_1$ is a fresh null value. But θ_1 is now being violated again, and a new fact $L(\perp_1, \perp_2)$, where $\perp_2$ is a fresh null value, will have to be added. It is clear that this process will continue indefinitely, and thus, that the chase does not terminate.

Assume finally that Σ_t consists of the egd $\alpha = G(x, y) \to x = y$. Then the chase for I fails, since after populating the target instance with the fact $G(a, b)$, the egd α forces to equate the constants a and b. Notice that, in this case, I has no solution. □

As we have seen, the main problem with the application of the chase is non-termination. Fortunately, there is a meaningful class of data exchange settings, that is introduced next, for which the chase is guaranteed to terminate; further, it does so in at most polynomially many steps. It will follow that this class of settings satisfies our desiderata expressed above as conditions C1, C2 and C3.

Assume that Σ is a set of tgds over $\mathbf{T}$. We construct the *dependency* graph of Σ as follows. The nodes (positions) of the graph are all pairs (T, A), for $T \in \mathbf{T}$ and A an attribute of T. We add edges as follows. For every tgd $\forall \bar{x}(\varphi(\bar{x}) \to \exists \bar{y} \psi(\bar{x}, \bar{y}))$ in Σ, and for every $x \in \bar{x}$ that occurs in φ in position (T, A) and that also occurs in ψ, do the following:

- for every occurrence of x in ψ in position (S, B), add an edge from (T, A) to (S, B) (if the edge does not already exist); and
- for every existentially quantified variable $y \in \bar{y}$ and for every occurrence of y in ψ in position (R, C), add an edge labeled $\star$ from (T, A) to (R, C) (if the edge does not already exist).

Finally, we say that Σ is *weakly acyclic* if the dependency graph of Σ does not have a cycle going through an edge labeled $\star$.

Example 4.3 *Let θ_1 and θ_2 be as in Example 4.2. It is not hard to see that the set $\{\theta_1\}$ is weakly acyclic. On the other hand, the set $\{\theta_1, \theta_2\}$ is not.* □

Interesting classes of weakly acyclic sets of tgds include the following (see [30]): (1) Sets of tgds without existential quantifiers, and (2) *acyclic* sets of inclusion dependencies as defined in [13]. The notion of weak acyclicity was first formulated by Deutsch and Popa, and later independently used in [19] and [15]. The intuition behind this notion is as follows. Edges labeled $\star$ keep track of positions (R, C) for which the chase will have to create a fresh null value every time the left hand side of the corresponding tgd is triggered. Thus, a cycle through an edge labeled $\star$ implies that a fresh null value created in a position at a certain stage of the chase may determine the creation of another fresh null value, in the same position, at a later stage. Therefore, sets of tgds that are not weakly acyclic may yield non-terminating chase sequences (e.g. the set $\{\theta_1, \theta_2\}$). On the other hand, it has been proved in [19] that the chase always terminates for data exchange settings with a weakly acyclic sets of tgds. Further, in this case the chase for I terminates in at most polynomially many stages. This good behavior also implies the good behavior of this class of settings with respect to data exchange, as summarized below:

Theorem 4.4 *[19] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a fixed data exchange setting, such that Σ_t is the union of a set of egds and a weakly acyclic set of tgds. Then there is a polynomial time algorithm such that for every source instance I, it first decides whether a solution for I exists, and if that is the case, it computes a universal solution for I in polynomial time.*

Thus, the class of settings with a weakly acyclic set of tgds satisfies conditions C1, C2, and C3, as defined above, and therefore, it constitutes a good class for data exchange according to our definition. In this case, the universal solution that can be constructed in polynomial time (if a solution exists at all) is precisely the result of the chase. This is usually called the *canonical* universal solution [19].

Let us mention that there are interesting classes of dependencies for which the problem of checking for the existence of solutions is trivial. For instance, by inspecting the proof of Theorem 4.4, one notices that for settings with a weakly acyclic set of tgds but without egds, source instances always have (universal) solutions. In particular, for settings without target dependencies it is the case that every source instance has at least one universal solution.

Other extensions Due to the advent of data exchange, the last years have seen a renewed interest on finding restrictions that guarantee termination of the chase. As we have just mentioned, one such restriction is the class of settings with a weakly acyclic set of tgds. However, both [16] and [38] showed that there are even broader classes of settings that preserve the good properties for data exchange. For some of these classes, the gain in expressive power comes at the cost of complexity: checking whether a setting belongs to some of these classes is in coNP, while it is clearly polynomial to verify whether a set of tgds is weakly acyclic.

In this section, the complexity analysis of the problem of checking the existence of solutions has been carried out assuming settings to be fixed. In other terms, we have studied the *data* complexity of the problem. While data complexity makes sense in a lot of data exchange scenarios, a more refined complexity analysis should consider both source instances and settings to be the input. This corresponds to the *combined* complexity of the problem. Kolaitis et al. have shown in [31] that the combined complexity of the problem of checking for the existence of solutions for settings with a weakly acyclic set of tgds is in EXPTIME, and can be EXPTIME-hard even if restricted to settings without tgds.

4.1 The core

Let us recall Examples 2.2 and 3.2. We mentioned that the instance $J = \{P(a, b, \perp_1), P(a, b, \perp_2), Q(\perp_3, \perp_1)\}$ is a solution for the source instance $\{M(a, b), N(a, b)\}$. Indeed, it is not hard to see that J is the canonical universal solution for I. Now, consider the instance $J^* = \{P(a, b, \perp_1), Q(\perp_3, \perp_1)\}$ that is contained in J. Then J^* is also a solution for I and, moreover, there is a homomorphism $h : J \to J^*$. Thus, J^* is also a universal solution.

We can draw an interesting conclusion from this example: among all possible universal solutions, the canonical universal solution is not necessarily the smallest (as J^* is strictly contained in J). Moreover, in the example, J^* is actually the smallest universal solution (up to isomorphism).

The first natural question is whether there is always a unique smallest universal solution. This was answered positively by Fagin et al. in [20]. The authors of [20] also argued that this smallest universal solution is the "best" universal solution, since it is the most economical one in terms of size, and that this solution should be the preferred one at the moment of materializing a solution. The whole issue is then how to characterize this smallest universal solution.

One of the main contributions in [20] is showing that the smallest universal solution always coincides with the *core* of the universal solutions. The *core* is a concept that originated in graph theory [28]; here we present it for arbitrary instances. Let K be an instance with values in $\texttt{Const} \cup \texttt{Var}$, and let K' be a subinstance of K. We say that K' is a *core* of K if there is a homomorphism from K to K' (recall that homomorphisms have to be the identity on constants), but there is no homomorphism from K' to a proper subinstance of itself. It is known that every instance has a core, and all cores of an instance are isomorphic [28, 20]. Thus, we can talk about *the* core of an instance K.

An instance J of $\mathbf{R}$ is a *subinstance* of I if the domain of J is contained in the domain of I and $J(R) \subseteq I(R)$, for every R in $\mathbf{R}$. If one of the inclusions is proper, we refer to J as a *proper subinstance* of I.

Example 4.5 *(Example 2.2 continued) The solution $J^* = \{P(a, b, \perp_1), Q(\perp_3, \perp_1)\}$ is the core of the universal solutions for I, since there is a homomorphism from J to J^* but there is no homomorphism from J^* to a proper subinstance of itself.* □

The next result summarizes some of the good properties of cores in data exchange.

Proposition 4.6 *[20] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a setting. If I is a source instance and J is a universal solution for I, then the core of J is also a universal solution for I that is precisely the smallest universal solution.*

Thus, the core of the universal solutions has good properties for data exchange. This naturally raises the question about the computability of this core. As we have mentioned, the chase yields a universal solution that is not necessarily the core of the universal solutions, so different techniques have to be applied in order to compute this solution.

It is well-known that computing the core of an arbitrary graph is a computationally intractable problem. However, in data exchange we are interested in computing the core of a universal solution and not of an arbitrary instance, and the intractability of the former problem does not follow from the intractability of the latter. Indeed, it has been shown in [26] that computing the core of the universal solutions under the class of settings with a weakly acyclic set of tgds is a tractable problem.

Theorem 4.7 *[26] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a fixed data exchange setting, such that Σ_t consists of a set of egds and a weakly acyclic set of tgds. There is a polynomial-time algorithm that for every source instance I, checks whether a solution for I exists, and if that is the case, computes the core of the universal solutions for I.*

A simple greedy algorithm that computes the core in polynomial time can be defined in the case when Σ_t consists of a set of egds [20, 5]. Unfortunately, this algorithm cannot be easily adapted to more complex settings, and much more sophisticated techniques have to be developed if one wants to prove that computation of cores of universal solutions continues to be a tractable problem in the presence of tgds. These techniques are based on the *blocks* method, that was also introduced in [20]. Gottlob in [25] developed a refined version of the blocks method, and proved with it that computing cores of universal solutions for settings whose set of target dependencies consist of egds and a set of tgds without existential quantifiers can be done in polynomial time. Later, in [26], Gottlob and Nash developed an even more sophisticated version of this method, and proved Theorem 4.7 with it.

5 Query Answering

Another big issue in data exchange is the semantics of query answering. Assume that a user poses a query Q over the target schema $\mathbf{T}$, and I is a given source instance. Then, what does it mean to answer Q with respect to I? Clearly, there is an ambiguity here, since there may be many solutions for I, and the evaluation of Q over different solutions may give different answers. The fact that one materializes only a single solution does not mean that others should not be taken into account when defining the semantics of the query.

There is a general agreement in the database community that in the context of data exchange and other related scenarios, like databases with incomplete information and data integration [32, 33], the right semantics is that of *certain* answers. Intuitively, an answer is *certain* if it occurs in the result of evaluation of query Q over every possible solution J. Notice that the semantics of a query is independent of a particular solution that is materialized.

Formally, let $\mathcal{M}$ be a data exchange setting, let Q be a query in some query language (typically FO), and let I be a source instance. We define $\texttt{certain}_{\mathcal{M}}(Q, I)$, the set of *certain answers of Q with respect to I under $\mathcal{M}$*, as $\bigcap\{Q(J) \mid J \text{ is a solution for } I\}$. We omit $\mathcal{M}$ if it is clear from the context. If Q is a query of arity 0 (a *Boolean* query), then $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{true}$ iff Q is true in every solution J for I; otherwise, $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$.

Example 5.1 *(Example 2.2 continued) The certain answers of the query $Q = P(x, y, z)$ with respect to I is the empty set. On the other hand, it is not hard to see that $\texttt{certain}_{\mathcal{M}}(Q', I) = \{(a, b)\}$, for $Q' = \exists z P(x, y, z)$.* □

Given a setting $\mathcal{M}$ and a query Q, the problem of computing certain answers for Q under $\mathcal{M}$ is, given a source instance I and a tuple $\bar{t}$, determine whether $\bar{t} \in \texttt{certain}_{\mathcal{M}}(Q, I)$. We deal with data complexity, i.e., assume that both $\mathcal{M}$ and Q are fixed. Finding certain answers involves computing the intersection of a (potentially) infinite number of sets. This strongly suggests that computing certain answers for arbitrary FO queries is an undecidable problem. Indeed, this is a rather straightforward consequence of Theorem 4.1. This does not preclude, however, the existence of interesting classes of queries for which the problem of computing certain answers is decidable, and even tractable. Indeed, next theorem shows that this is the case for the class of unions of conjunctive queries. Recall that a *conjunctive* query is an FO formula of the form $\exists \bar{x} \varphi(\bar{x}, \bar{y})$, where $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms.

Theorem 5.2 *[19] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a data exchange setting, such that Σ_t consists of a set of egds and a weakly acyclic set of tgds, and let Q be a union of conjunctive queries. Then the problem of computing certain answers for Q under $\mathcal{M}$ can be solved in polynomial time.*

This is a very positive result, since unions of conjunctive queries are very common database queries: they correspond to the *select-project-join-union* fragment of relational algebra, and to the core of the standard query language for database systems, SQL. Theorem 5.2 can be easily proved when we put together the following three facts:

(1) Unions of conjunctive queries are preserved under homomorphisms, (2) every universal solution can be homomorphically mapped into any other solution, and (3) FO queries, and in particular, unions of conjunctive queries, have polynomial time data complexity.

Thus, in order to compute the certain answers to a union of conjunctive queries Q with respect to a source instance I it is sufficient to do the following. First, check whether a solution for I exists. If there is no solution, simply declare the setting to be inconsistent with respect to the given source instance. Otherwise, compute an arbitrary universal solution J for I. Finally, compute the set $\texttt{ground}(Q(J))$ of all those tuples in $Q(J)$ that do not contains nulls. From all the previous observations, and from the fact that tuples in the certain answers can only consist of constants, we derive that $\texttt{ground}(Q(J)) = \texttt{certain}_{\mathcal{M}}(Q, I)$. Notice that for the class of settings with a weakly acyclic set of tgds this algorithm runs in polynomial time. Finally, the algorithm also shows another desirable property of unions of conjunctive queries for data exchange; namely, their certain answers can be computed using a materialized target instance alone (e.g. the canonical universal solution or the core).

A mild, but interesting extension of the class of conjunctive queries is the class of conjunctive queries with inequalities. Unfortunately, this extension not only destroys preservation under homomorphisms but also leads to intractability of the problem of computing certain answers (first shown in [1] and then sharpened in [36]):

Theorem 5.3 *[36] There is a single Boolean conjunctive query Q with two inequalities and a setting $\mathcal{M}$ without target dependencies, such that the problem of computing certain answers for Q under $\mathcal{M}$ is* coNP*-complete.*

In particular, under widely believed complexity-theoretic assumptions, there is no polynomial-time algorithm that computes certain answers to conjunctive queries with inequalities using the canonical universal solution or the core, even in settings without target dependencies. On the other hand, by using techniques based on the chase procedure, Fagin et al. proved in [19] that the problem of computing certain answers to unions of conjunctive queries, with at most one inequality per disjunct, can be solved in polynomial time using an arbitrary universal solution.

Query rewriting In [4], Arenas et al. studied the following problem. Let Q be a FO query. Is it possible to find an FO query Q' such that for every source instance I, the certain answers of Q with respect to I correspond exactly to the evaluation of Q' over either the canonical universal solution or the core of the universal solutions for I? If that is the case, the query Q' is called a *rewriting* of Q over the corresponding universal solution. In particular, in [4] the authors developed techniques that helps to determine when a query admits a rewriting. This work also compared the canonical universal solution and its core in terms of the expressive power for allowing rewritings. It was shown that every query that admits a rewriting over the core of the universal solutions also admits a rewriting over the canonical universal solution, but not vice versa.

Extensions More expressive query languages, that allow for the presence of recursion and a restricted form of negation, and that preserve the good properties of unions of conjunctive queries for data exchange, were recently introduced by Arenas et al. [8]. Translations into this query language provide an alternative proof that computing certain answers for unions of conjunctive queries, with at most one inequality per disjunct, can be done in polynomial time. That paper also studied the combined complexity of computing certain answers in data exchange.

6 Alternative Semantics

The certain answers semantics is by no means unique, although it had been predominant in the early stages of data exchange research. In fact, the certain answers semantics has its problems, and a question "What is so sacred about the certain answers semantics?" was posed in [10]. Several attempts to address it were made; we survey them below.

6.1 Universal solutions semantics

Since [19], the seminal paper in data exchange, made a compelling case that the preferred solutions in data exchange should be the universal solutions, it seems natural to think of an alternative semantics that is completely based on those solutions. Formally, let $\mathcal{M}$ be a data exchange setting, let Q be a query, and let I be a source instance. The set of *universal certain answers to Q with respect to I under $\mathcal{M}$* [20], denoted by $\texttt{certain}^u_{\mathcal{M}}(Q, I)$, is defined as the set $\bigcap\{Q(J) \mid J \text{ is a universal solution for } I\}$. The problem of computing the universal certain answers to Q under $\mathcal{M}$ is defined analogously to the case of the standard semantics.

Since computing certain answers for a union of conjunctive queries can be done by posing the query over an arbitrary universal solution, it easily follows that for each query Q of that form and every source instance I, $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{certain}^u_{\mathcal{M}}(Q, I)$. But this is no longer the case if we allow inequalities.

Example 6.1 *(Example 2.2 continued) The universal certain answers to $Q = \exists x \exists y \exists z (P(x, y, z) \wedge x \neq z)$ with respect to I is* `true`*, since every universal solution J for I must contain a fact of the form $P(a, b, \perp)$, for $\perp$ a value in* `Var`*. On the other hand,* $\texttt{certain}_{\mathcal{M}}(Q, I) = \texttt{false}$ *since $J = \{P(a, b, a), Q(b, a)\}$ is a solution for I.*

We show next that the universal certain answers semantics presents some computational advantages over the usual semantics. An *existential* FO formula is a formula of the form $\exists \bar{x} \varphi(\bar{x}, \bar{y})$, where $\varphi(\bar{x}, \bar{y})$ is a quantifier-free formula

in disjunctive normal form. Existential formulas are known to have the following property: If K_1 is an induced subinstance of K_2 and $\bar{t}$ belongs to the evaluation of θ over K_1, where θ is an existential FO formula, then $\bar{t}$ belongs to the evaluation of θ over K_2. Since every universal solution contains an isomorphic copy of the core as an induced subinstance, it easily follows that the problem of computing the universal certain answers for an existential FO query θ boils down to evaluating θ over the core of the universal solutions, discarding all tuples that do not consist only of constants. Putting this together with Theorem 4.7 it is possible to obtain the following:

Theorem 6.2 *[20] Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ be a setting, such that Σ_t consists of a set egds and a weakly acyclic set of tgds, and let Q be a (domain-independent) existential* FO *formula. Then the problem of computing universal certain answers of Q under $\mathcal{M}$ can be solved in polynomial time.*

Notice that this is in deep contrast with Theorem 5.3 that shows that for very simple existential FO formulas (namely, conjunctive queries with two inequalities) the problem of computing certain answers becomes intractable.

6.2 Semantics based on the closed-world assumption and incomplete information

Both the certain answers and the universal certain answers semantics are based on an *open-world assumption* (OWA), i.e. solutions included in those semantics are open to adding new facts. This assumption, however, leads to some anomalies with respect to query answering, as described next. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$ be a data exchange setting without target dependencies. We say that $\mathcal{M}$ is *copying* if $\mathbf{S} = \langle R_1, \ldots, R_m \rangle$, $\mathbf{T} = \langle R'_1, \ldots, R'_m \rangle$, and Σ_{st} consists of the stds $\forall \bar{x}(R_i(\bar{x}) \rightarrow R'_i(\bar{x}))$ (that is, R_i and R'_i have the same arity). It is pointed out in [4] that there is a copying data exchange setting $\mathcal{M}$ and an FO query Q over the target schema, such that Q is not FO rewritable over the canonical universal solution under $\mathcal{M}$. That is, there is no FO query Q' such that for every source instance I, the evaluation of Q' over the canonical universal solution J for I coincides with the certain answers of Q with respect to I. This behavior is strange, as intuitively copying settings only change relation names and canonical universal solutions are just copies of source instances. As such, one would expect every FO query to be rewritable by itself over the canonical universal solution.

The previous anomaly arises because OWA assumes the canonical universal solution to be open to adding new facts – both under the certain answers and the universal certain answers semantics – while intuitively we would expect the canonical universal solution to be the only solution under copying settings. Because of that, Libkin proposed in [34] that the right assumption in data exchange should not be the OWA but its usual competitor, the *closed-world assumption* (CWA), in which solutions are closed to adding new facts. The reason behind this is that if data exchange is about transferring data from source to target, then the semantics for the data exchange problem should be based on the exchanged data only, and not on data that can be later added to the instances. Or in other terms, the semantics should be based on those solutions that contain no more than what is needed to satisfy the specification.

Unfortunately, defining CWA-solutions in data exchange is not so easy as defining the OWA solutions. Here we avoid the rather elaborate operational definition of CWA-solutions given in [34], and provide a more elegant semantic description of this class (which appears as a characterization of such solutions in [34]). As usual, K' is a *homomorphic image* of K if there is a homomorphism $h : K \rightarrow K'$ such that $h(K)$, the instance obtained from K by replacing each null $\perp$ by $h(\perp)$, is exactly K'.

Definition 6.3 *Let $\mathcal{M}$ be a setting without target dependencies, I a source instance and J a solution for I. Then J is a* CWA-solution *for I if (1) J is a universal solution for I, and (2) J is a homomorphic image of the canonical universal solution for I.*

Thus, if $\mathcal{M}$ is a copying setting and I is a source instance, then the unique CWA-solution for I under $\mathcal{M}$ is its canonical universal solution J (since any other universal solution is not a homomorphic image of J). Next we show a slightly more interesting example.

Example 6.4 *(Example 2.2 continued) Recall that $I = \{M(a, b), N(a, b)\}$. Then, with respect to I, both its canonical universal solution $J = \{P(a, b, \perp_1), P(a, b, \perp_2), Q(\perp_3, \perp_1)\}$ and the core of its universal solutions $J^* = \{P(a, b, \perp_1), Q(\perp_3, \perp_1)\}$ are* CWA-*solutions. On the other hand, the solution $J_1 = \{P(a, b, \perp_1), Q(\perp_1, \perp_1)\}$ is not a* CWA-*solution, simply because it is not a universal solution for I. The solution $J_2 = \{P(a, b, \perp_1), P(a, b, \perp_2), P(a, b, \perp_4), Q(\perp_3, \perp_1)\}$ is a universal solution for I that it is not a homomorphic image of J. Therefore, J_2 is not a* CWA-*solution.* □

Notice the following properties of the class of CWA-solutions. The core of the universal solutions is a "minimal" element of the class, in the sense that every other CWA-solution contains an isomorphic copy of the core as a subinstance. Also, the canonical universal solution is in a way the "maximal" element of the class, as any other CWA-solution is a homomorphic image of it. This observation will be relevant later once we study the properties of query answering.

In addition to defining the class of CWA-solutions, [34] also points out that while target instances in data exchange are tables with nulls, techniques for handling incomplete information over target instances had been completely ignored in previous data exchange literature. Indeed, the usual semantics of data exchange (e.g. certain answers or

universal certain answers) are defined over sets of solutions as if each one were a table without nulls.

But already 25 years ago, Imielinski and Lipski [32] showed that answering queries over databases with nulls must be done with care, and that treating nulls in the same way as constants yields semantically incorrect answers. The basic idea in [32] is that a database T with nulls represents a set $\mathbf{Rep}(T)$ of "complete" databases. i.e., a set of databases without nulls. We formally define this as follows. A *valuation* is a mapping $\nu : \mathtt{Var} \to \mathtt{Const}$. If T is an instance with elements in $\mathtt{Const} \cup \mathtt{Var}$, then $\nu(T)$ represents the instance obtained from T by replacing each null $\perp$ with $\nu(\perp)$. Notice that all the elements in $\nu(T)$ are constants. Then we define $\mathbf{Rep}(T) = \{\nu(T) \mid \nu \text{ is a valuation}\}$.

In order to evaluate a query Q over an incomplete database T, the standard approach is to compute the set $\Box Q(T) = \bigcap\{Q(D) \mid D \in \mathbf{Rep}(T)\}$. This set is usually called the certain answers of Q with respect to T in the incomplete information literature, but we prefer to represent it by $\Box Q(T)$ here, in order to avoid confusion with the certain answers as defined for data exchange.

Although [34] proposes four different semantics for data exchange, here we concentrate on one that seems to be the most relevant for data exchange and that is, at the same time, the closest to the semantics we have seen so far. Let $\mathcal{M}$ be a setting without target dependencies, Q an FO query, and I a source instance. Then we define the *certain answers under* CWA *and incomplete information* of Q with respect to I (under $\mathcal{M}$), denoted by $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I)$, as the set of tuples that would be in $Q(D)$ for every CWA-solution J and every $D \in \mathbf{Rep}(J)$, that is, $\bigcap\{\Box Q(J) \mid J \text{ is a CWA-solution for } I \text{ under } \mathcal{M}\}$.

Notice that for each source instance I with canonical universal solution J, and for every CWA-solution J' for I, it must be the case that $\mathbf{Rep}(J') \subseteq \mathbf{Rep}(J)$. Thus, we have that $\Box Q(J) \subseteq \Box Q(J')$, for every FO query Q. Hence:

Theorem 6.5 *[34] Let $\mathcal{M}$ be a setting without target dependencies and Q an arbitrary query. Then $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I) = \Box Q(J)$, for every source instance I with canonical universal solution J.*

That is, in this case the problem of evaluating certain answers under CWA and incomplete information boils down to the problem of evaluating queries over incomplete databases (canonical universal solutions).

Example 6.6 *(Example 2.2 continued) Consider Q_1 be a query asking whether the interpretation of relation Q has exactly one tuple. It can be shown that $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q_1, I) = \mathtt{true}$. Indeed, every* CWA*-solution must contain at least one tuple in the interpretation of Q, but it cannot contain two tuples: this is because every* CWA*-solution is a homomorphic image of the canonical universal solution $J = \{P(a, b, \perp_1), P(a, b, \perp_2), Q(\perp_3, \perp_1)\}$ for I. On the other hand, $\mathtt{certain}_{\mathcal{M}}(Q, I) = \mathtt{certain}^{u}_{\mathcal{M}}(Q, I) = \mathtt{false}$.* □

The previous example shows that there is a setting $\mathcal{M}$ and a FO query Q such that $\mathtt{certain}^{\mathrm{CWA}}_{\mathcal{M}}(Q, I) \neq \mathtt{certain}_{\mathcal{M}}(Q, I)$ for some source instance I. On the other hand, there is an interesting class of queries for which the semantics based on CWA and incomplete information coincides with the usual semantics. This is the class of *monotone* FO queries. It follows that the problem of computing certain answers under CWA and incomplete information for conjunctive queries with inequalities is coNP-complete, and that for unions of conjunctive queries it becomes tractable. For arbitrary FO queries the problem is always in coNP, which contrasts sharply with the case of the usual semantics where the problem of computing certain answers may be undecidable.

Extensions Recently, Hernich and Schweikardt extended the notion of CWA-solutions and semantics based on incomplete information to data exchange settings with target dependencies [29]. Further, Libkin and Sirangelo [35] have recently proposed a mixed approach to data exchange that combines the closed- and the open-world assumption.

7 Related Work and Extensions

Data exchange is closely related to the problem of data integration [33]. A data integration system consists of a *local* schema, a *global* schema, and a specification of the relationship between the local and the global schema. The data resides at the local level, but the user can only query the data at the global level. Thus, a data integration system can be seen as a data exchange setting where the source schema corresponds to the local schema, and the target schema corresponds to the global schema. Furthermore, the goal in both data integration and data exchange is to compute the certain answers to a query that is posed over the target (global) schema.

The main difference between data exchange and *virtual* data integration is that in the latter, the data is never actually exchanged, as the global database corresponds only to a virtual representation of the local database. As such, the goal in data integration is to compute the certain answers to a query based on the local data, as opposed to data exchange where the goal is to do the same but using a materialized target instance. But there are many commonalities, and the formal study of the relationship between data exchange and data integration deserves further study (see, e.g., [14]).

An area of study that has evolved from data exchange, but that by now has become a prominent topic on its own, is schema management. The fundamental idea is that schema mappings correspond to metadata, and that it is important to have a conceptual framework in which this metadata is combined by applying some predefined operators [11]. The formal study of two of these operators, the *composition* and the *inverse*, has recently been started [18, 21, 22, 7]. The interesting part of this story for data exchange is that, in many cases, in order to completely understand the schema man-

agement operator under scope, it is necessary to use more expressive source-to-target dependencies than the ones used in this article. This defines a completely new data exchange problem.

Several extensions of the data exchange problem, as studied here, have been proposed in the recent years. For instance, Fuxman et al. proposed in [24] an extension of the class of data exchange settings in which dependencies from target-to-source are also allowed. The motivation for studying this class of settings comes from the area of peer data management systems [27], which model the case when different databases (peers) interact with each other, sharing and exchanging data. The source peer is the "authoritative" peer that contributes with data. On the other hand, the target peer restricts the data it is willing to accept by means of target-to-source dependencies, but has no right to modify the source data.

Other extensions include the class of queries that data exchange systems can support and the class of values allowed in source instances. With respect to the first one, Afrati and Kolaitis recently studied the topic of answering aggregate queries in data exchange [2]. In order to do so, they had to define a new semantics for query answering, based on the class of endomorphic images of the canonical universal solutions. This is because all the semantics we have seen so far yield rather trivial semantics for aggregate queries in data exchange. With respect to the latter, Fagin et al. [23] proposed an extension of data exchange that allows for null values also to appear in source instances. This represents the fact that incomplete information can also arise in the source; e.g. when the materialized target instance of one data exchange setting is used as the source instance of another setting.

Finally, in this article we studied the logical foundations of relation data exchange. However, semistructured models, and in particular, XML, have been specially designed for exchanging data on the web. Thus, it seems natural to develop a notion of XML data exchange. The foundational study of this problem was initiated by Arenas and Libkin [6], which unveiled many of the fundamental problems that appear in this new scenario; schema mappings for XML were studied in Amano et al. [3].

8 Concluding Remarks

Data exchange is nowadays one of the most active areas of database theory. In this paper we tried to give a brief introduction to the main lines of research involved in this problem, with a special emphasis on the topics of materializing solutions, query answering, and the notions of the semantics for data exchange.

Some of these topics are already quite mature, while others need further exploration. For instance, we have mentioned that doing proper query answering is one of the main goals of data exchange systems, but nevertheless, the knowledge we have about this topic is rather limited and several interesting questions remain open. One of the most challenging is finding bigger classes of database queries for which the problem of computing certain answers is decidable. Also, we believe that the time is ripe for having a fruitful discussion that compares the different semantics proposed for query answering in data exchange, and developing criteria that help determine when a semantics is appropriate. A natural question, for instance, is whether we should have an overall data exchange semantics that works for every query, or whether it would be better to take a more pragmatic view that takes into account the class of queries at hand, and permits a lower quality of the answer in favor of the performance of the system.

As we have mentioned, developing a systematic study of XML data exchange is crucial for understanding the transfer of data on the web. However, there are very few papers that have dealt with this issue up till now. Relevant, but unexplored topics related to XML data exchange include, for example, the problems of existence of solutions, materializing solutions, query answering beyond conjunctive queries, and notions of CWA and incomplete information.

Acknowledgment The author gratefully acknowledges support by FONDECYT grant 11080011. Also, comments by M. Arenas and L. Libkin on an early draft have been of great help.

References

[1] S. Abiteboul, and O. Duschka. Answering queries using materialized views. Gemo report 383.

[2] F. N. Afrati, and P. G. Kolaitis. Answering aggregate queries in data exchange. In *PODS*, pages 129–138, 2008.

[3] S. Amano, L. Libkin, and F. Murlak. XML schema mappings. In *PODS*, 2009.

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

[5] M. Arenas. Personal communication, 2004.

[6] M. Arenas, L. Libkin. XML data exchange: Consistency and query answering. *Journal of the ACM* 55(2), 2008.

[7] M. Arenas, J. Pérez, C. Riveros. The recovery of a schema mapping: bringing exchanged data back. In *PODS*, pages 13-22, 2008.

[8] M. Arenas, P. Barceló, J. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. In *ICDT*, 2009.

[9] C. Beeri, and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

[10] L. Bertossi, J. Chomicki, P. Godfrey, P. Kolaitis, A. Thomo, C. Zuzarte: Exchange, integration, and consistency of data: report on the ARISE/NISR workshop. SIGMOD Record 34(3): 87-90, 2005.

[11] P. Bernstein. Applying model management to classical meta-data problems. In *CIDR*, 209-220, 2003.

[12] Clio Project. www.almaden.ibm.com/software/projects/criollo

[13] S. Cosmadakis, P. Kanellakis. Functional and inclusion dependencies; A graph theoretica approach. In *Advances in Computing Research*, vol. 3, 163.184, 1986.

[14] G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS*, pages 133–142, 2007.

[15] A. Deutsch, V. Tannen. Reformulation of XML queries and constraints. In *ICDT*, 225-241, 2003.

[16] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *PODS*, pages 149-158, 2008.

[17] R. Fagin. Horn clauses and database dependencies. *Journal of the ACM*, 29(4): 952-985, 1982.

[18] R. Fagin, P. Kolaitis, L. Popa, W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.

[19] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005. Preliminary version in *ICDT*, 2003.

[20] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005.

[21] R. Fagin. Inverting schema mappings. In *PODS*, pages 50-59, 2006.

[22] R. Fagin, P. Kolaitis, L. Popa, W.-C. Tan. Quasi-inverses of schema mappings. *ACM Transactions on Database Systems* 33(2), (2008).

[23] R. Fagin, P. Kolaitis, L. Popa, W.-C. Tan. Reverse data exchange: Coping with nulls. In *PODS*, 2009.

[24] A. Fuxman, P. Kolaitis, R. Miller, W.-C. Tan. Peer data exchange. In *PODS*, pages 160-171, 2005.

[25] G. Gottlob. Computing cores for data exchange: New algorithms and practical solutions. In *PODS*, pages 148–159, 2005.

[26] G. Gottlob, A. Nash. Data exchange: Computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

[27] A. Halevy, Z. Ives, D. Suciu, I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, pages 505-518, 2003.

[28] P. Hell, J. Nešetřil. *Graphs and homomorphisms*. Oxford University Press, 2004.

[29] A. Hernich, N. Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *PODS*, pages 113-122, 2007.

[30] P. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[31] P. Kolaitis, J. Panttaja, and W.-C. Tan. The complexity of data exchange. In *PODS*, pages 30–39, 2006.

[32] T. Imielinski, W. Lipski. Incomplete information in relational databases. *Journal of the ACM* 31, 761–791, 1984.

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

[34] L. Libkin. Data exchange and incomplete information. In *PODS*, pages 60–69, 2006.

[35] L. Libkin, C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *PODS*, pages 139–148, 2008.

[36] A. Mądry. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters*, 94(6):253–257, 2005.

[37] D. Maier, A. Mendelzon, Y. Sagiv. Testing implications on data dependencies. *ACM Transactions on Database Systems*, 4(4), 455-469, 1979.

[38] M. Meier, M. Schmidt, G. Lausen. Stop the Chase. In *Alberto Mendelzon Workshop*, AMW, 2009.

[39] Second International Workshop on Exchange and Integration of Data, EIS06. http://www.scs.carleton.ca/ diis/EID06/