Expressiveness and Complexity of Bidirectional Constraints for Data Exchange

Marcelo Arenas[1], Gabriel Diéguez[1], and Jorge Pérez[2]

[1] Department of Computer Science, Pontificia Universidad Católica de Chile
[2] Department of Computer Science, Universidad de Chile

1 Introduction

A schema mapping is a high-level specification that describes how data from a source schema is to be mapped to a target schema. Schema mappings are of fundamental importance in data management today. In particular, they have proved to be the essential building block for several data-interoperability tasks such as data exchange [8], data integration [11] and peer data management [7].

Most of the research on schema mappings has focused on mappings specified by the so called *source-to-target tuple-generating-dependencies* (st-tgds), motivated by the highly influential formalization of data exchange proposed by Fagin et al. in 2003. Although natural and simple to specify, mappings given by st-tgds fail to impose enough conditions to unambiguously define what are the instances that should be materialized when exchanging data. This raises several issues, among others, what is a good semantics for data exchange with st-tgds, how to choose a good target instance when exchanging data, and how to answer target queries. For instance, a very successful line of research has been the definition of a *closed world semantics* for st-tgds. Although this new semantics departs from a classical First Order Logic (FO) semantics, it has proved to have good properties in terms of materialization of target instances [12]. Other lines of research include the study of alternative notions for answering target queries, in particular, non-monotone queries [10] and aggregate queries [1].

We argue that one can follow a different approach and, instead of using ad-hoc solutions for each of the issues mentioned above, one can use a mapping-specification language that imposes enough constraints over possible target instances in order to minimize the uncertainty when exchanging data. In that way one can use standard FO notions to define the semantics of mappings, and to define the possible target solutions as well as the process of answering target queries.

We propose to use what we call *bidirectional constraints* to specify mappings. These specifications impose at the same time constraints over the source and target instances participating in a mapping, and have the potential to minimize the ambiguity in the description of the target instances that should be materialized when exchanging data. Bidirectional constraints are formulas of the form $\forall \bar{x}\big(\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})\big)$ where $\varphi(\bar{x})$ is a formula over the source schema, and $\psi(\bar{x})$ is a formula over the target schema. One can obtain several different languages of bidirectional constraints depending on the formulas allowed in the source and target parts. Although natural and useful in many scenarios, mappings given by bidirectional constraints have been almost disregarded in the literature on the foundations of schema mappings and data exchange.

In this paper we present our ongoing work on the study of schema mappings defined by bidirectional constraints. We show that given a mapping specified by st-tgds without existential quantifiers, one can construct a set of bidirectional constraints that exactly defines the *canonical universal solutions* of the initial mapping. Canonical universal solutions are considered as the preferred solutions when exchanging data with st-tgds [8]. This result presents evidence in favor of our hypothesis that bidirectional constraints can be useful to unambiguously define exchange scenarios. We prove that in order to achieve this result, disjunctions and equalities are strictly necessary in the definition of bidirectional constraints. Moreover, we also study the *existence-of-solutions* problem for bidirectional constraints, showing that the complexity depends on the use of existential quantification in the target formulas: the problem can be solved in polynomial time if no existential quantification is allowed, while it may be intractable in the general setting.

The rest of the paper is organized as follows. Section 2 presents some notation and preliminary notions. Section 3 shows how the *canonical-universal-solution* semantics of st-tgds can be specified with bidirectional constraints. In Section 4 we study some complexity issues regarding bidirectional constraints. Section 5 presents some concluding remarks. For the sake of space all the proofs are in the extended version available online at `http://db.ing.puc.cl/~gdieguez/papers/amw2014-ext.pdf`.

2 Preliminaries

We assume some familiarity with database theory and data exchange [8]. We also assume that data is represented in the relational model. A *relational schema*, or just *schema*, is a finite set $\{R_1, \ldots, R_n\}$ of relation symbols, each relation having a fixed arity. Given a schema $\mathbf{R}$, we denote by $\mathrm{Inst}\,(\mathbf{R})$ the set of all instances of $\mathbf{R}$.

Schema mappings

Schema mappings are used to define a semantic relationship between two schemas. In this paper, we use a general definition of a schema mapping; given two schemas with no relation symbol in common, $\mathbf{S}$ and $\mathbf{T}$, a schema mapping (or just a mapping) $\mathcal{M}$ between $\mathbf{S}$ and $\mathbf{T}$ is a set of pairs (I, J), where I is an instance of $\mathbf{S}$, and J is an instance of $\mathbf{T}$. That is, a mapping $\mathcal{M}$ is just a subset of $\mathrm{Inst}\,(\mathbf{S}) \times \mathrm{Inst}\,(\mathbf{T})$. Given an instance I of $\mathbf{S}$, a mapping $\mathcal{M}$ associates to I a set of *possible solutions for* I, denoted by $\textsc{Sol}_{\mathcal{M}}(I)$, given by the set $\textsc{Sol}_{\mathcal{M}}(I) = \{J \in \mathrm{Inst}\,(\mathbf{T}) \mid (I, J) \in \mathcal{M}\}$.

In practice, schema mappings are represented by using logical formulas. In this paper we focus on using fragments of first-order logic (FO) to specify mappings (we assume some familiarity with FO). Given two disjoint relational schemas $\mathbf{S}$ and $\mathbf{T}$, and a set Σ of FO sentences over vocabulary $\mathbf{S} \cup \mathbf{T}$, we say that a mapping $\mathcal{M}$ is *specified* by Σ if for every pair of instances $(I, J) \in \mathrm{Inst}\,(\mathbf{S}) \times \mathrm{Inst}\,(\mathbf{T})$ it holds that $(I, J) \in \mathcal{M}$ if and only if $(I, J) \models \Phi$ for every $\Phi \in \Sigma$, where $\models$ denotes the standard satisfaction of FO formulas.

Besides FO, the main query languages that we consider in this paper are the languages of conjunctive queries (CQ), unions of CQ (UCQ), and the languages obtained from them by adding the equality predicate ($\mathrm{CQ}^{=}$, $\mathrm{UCQ}^{=}$ and $\mathrm{FO}^{=}$).

Schema mappings, bidirectional constraints, and source-to-target dependencies

Assume that we have two disjoint schemas $\mathbf{S}$ and $\mathbf{T}$. In this paper, we study mappings specified by sets of formulas of the following form

$$\forall \bar{x}\big(\ \varphi(\bar{x})\ \leftrightarrow\ \psi(\bar{x})\ \big), \tag{1}$$

where $\varphi(\bar{x})$ is an FO formula over $\mathbf{S}$ and $\psi(\bar{x})$ is an FO formula over $\mathbf{T}$, both formulas with $\bar{x}$ as tuple of free variables. We call this formula a *bidirectional constraint*. We usually drop the outermost universal quantification when specifying these constraints, and thus we only write $\varphi(\bar{x}) \leftrightarrow \psi(\bar{x})$ for formula (1). Depending on which fragments of FO we use to define formulas $\varphi(\bar{x})$ and $\psi(\bar{x})$, we obtain a wide range of possible fragments of bidirectional constraints. Given fragments $\mathcal{L}_1$ and $\mathcal{L}_2$ of FO, we say that a sentence Φ is an $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$ formula between $\mathbf{S}$ and $\mathbf{T}$, if Φ is a bidirectional constraint of the form (1) in which $\varphi(\bar{x})$ is an $\mathcal{L}_1$ formula over $\mathbf{S}$, and $\psi(\bar{x})$ is an $\mathcal{L}_2$ formula over $\mathbf{T}$. When the source and target schemas are clear from the context we will only talk about $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$ formulas. For example, consider schemas $\mathbf{S} = \{\texttt{Mother}(\cdot,\cdot), \texttt{Father}(\cdot,\cdot)\}$ and $\mathbf{T} = \{\texttt{Parent}(\cdot,\cdot)\}$. Then the following sentence

$$(\ \texttt{Father}(x,y) \vee \texttt{Mother}(x,y)\ )\ \leftrightarrow\ \texttt{Parent}(x,y) \tag{2}$$

is an example of a $\langle \text{UCQ}, \text{CQ} \rangle$ formula. An $\langle \mathcal{L}, \text{CQ} \rangle$ formula in which the target part is a CQ without existential quantifiers is called a *full* $\langle \mathcal{L}, \text{CQ} \rangle$ formula.

In this paper we also consider the language of source-to-target dependencies, which are formulas of the form $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$ where $\varphi(\bar{x})$ and $\psi(\bar{x})$ are formulas over the source and the target schema, respectively (we also usually drop the universal quantification in this case). An $\mathcal{L}_1$-TO-$\mathcal{L}_2$ dependency is a formula of the above form in which $\varphi(\bar{x})$ is in $\mathcal{L}_1$ and $\psi(\bar{x})$ is in $\mathcal{L}_2$. Similarly, as for the case of bidirectional constraints, we call *full* $\mathcal{L}$-TO-CQ dependencies to formulas in which the target part is a CQ without existential quantifiers. We usually refer to CQ-TO-CQ dependencies as source-to-target tuple-generating dependencies (st-tgds). A mapping defined by source-to-target dependencies is called an st-mapping.

Given a mapping $\mathcal{M}$ specified by FO-TO-CQ dependencies and a source instance I, one can define a particular class of solutions in $\text{SOL}_{\mathcal{M}}(I)$ called *universal solutions*. These solutions are the *most general* among all the possible solutions for I under $\mathcal{M}$ [8]. Moreover, a particular class of universal solutions, called *canonical universal solutions*, can be generated (in polynomial time) by means of the classical *chase procedure*. For the sake of space, we refer the reader to [8] for precise definitions of these notions. We call $\text{USOL}_{\mathcal{M}}(I)$ and $\text{CUS}_{\mathcal{M}}(I)$, the set of universal solutions and canonical universal solutions for I under $\mathcal{M}$, respectively. In general, for source-to-target mappings, we have that $\text{CUS}_{\mathcal{M}}(I) \subsetneq \text{USOL}_{\mathcal{M}}(I) \subsetneq \text{SOL}_{\mathcal{M}}(I)$.

3 Characterization of semantics for data exchange

One of our main goals in our ongoing project is to study the relationship between bidirectional constraints and the different semantics for data exchange. Let $\mathcal{M}$ be an st-mapping and I a source instance. The standard FO semantics for data exchange defines $\text{SOL}_{\mathcal{M}}(I)$ as the set of all possible target instances considered when materializing

data or answering queries after the exchange [8]. Given that $\text{SOL}_{\mathcal{M}}(I)$ can have many (potentially infinite) non desired solutions, several works have proposed to consider only solutions in $\text{USOL}_{\mathcal{M}}(I)$ or $\text{CUS}_{\mathcal{M}}(I)$ (or even other more ad-hoc sets of solutions) for exchanging or answering queries [12,1,10]. A natural question is if one can obtain these alternative semantics by considering the standard FO semantics but for alternative mappings. More precisely, given an st-mapping $\mathcal{M}$, is there another mapping $\mathcal{M}'$ such that either $\text{USOL}_{\mathcal{M}}(I) = \text{SOL}_{\mathcal{M}'}(I)$ or $\text{CUS}_{\mathcal{M}}(I) = \text{SOL}_{\mathcal{M}'}(I)$, for every possible source instance I?

In this section we present a positive result for the above mentioned scenario using bidirectional constraints. In particular, we show that for an st-mapping $\mathcal{M}$ specified by full FO-TO-CQ dependencies, one can always construct a mapping $\mathcal{M}^{\leftrightarrow}$ specified by bidirectional constraints such that $\text{CUS}_{\mathcal{M}}(I) = \text{SOL}_{\mathcal{M}^{\leftrightarrow}}(I)$. The bidirectional constraints used are $\langle \text{UCQ}^{=}, \text{CQ} \rangle$ plus formulas of the form $\forall \bar{x}(\bot \leftrightarrow P(\bar{x}))$, where $\bot$ is an arbitrary contradiction. Notice that these last formulas can be specified in $\langle \text{FO}, \text{CQ} \rangle$ but we have decided to leave them separated as they are used only to state that some target predicates must remain empty in the solutions. Algorithm 1 presents this procedure. In the algorithm we use a procedure QUERYREWRITINGATOM that given a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ specified by st-tgds and a target atom $R(\bar{x})$, constructs a formula $\alpha(\bar{x})$ in $\text{UCQ}^{=}$ such that for every tuple $\bar{a}$ the following holds: $I \models \alpha(\bar{a})$ if and only if for every solution $J \in \text{SOL}_{\mathcal{M}}(I)$ it holds that $J \models R(\bar{a})$. That is, $\alpha(\bar{x})$ is a *rewriting of* $R(\bar{x})$ *over the source* [4].It was shown in [4] that QUERYREWRITINGATOM can be implemented in quadratic time w.r.t. the size of Σ. We also consider a restriction in the input of the algorithm. We assume that the set of dependencies given as input satisfies that each dependency in the set has a single atom in its right-hand side, since it is well known that every set of full FO-TO-CQ dependencies is equivalent to a set that satisfies this restriction.

Algorithm 1 BIDIRECTIONALMAPPINGFULL $(\mathcal{M})$

Input: an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full FO-TO-CQ dependencies, each dependency with a single atom in its right-hand side.
Output: a mapping $\mathcal{M}^{\leftrightarrow} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of full $\langle \text{UCQ}^{=}, \text{CQ} \rangle$ dependencies plus dependencies of the form $\forall \bar{x}(\bot \leftrightarrow P(\bar{x}))$.

1: Start with Δ as the empty set.
2: **for all** dependencies in Σ of the form $\varphi(\bar{x}) \rightarrow R(\bar{x})$ **do**
3: Assume that n is the arity of predicate R and let $\bar{y}$ be an n-tuple of distinct fresh variables.
4: Use QUERYREWRITINGATOM$(\mathcal{M}, R(\bar{y}))$ to compute a formula $\alpha(\bar{y})$ in $\text{UCQ}^{=}$ that is a rewriting of $R(\bar{y})$ over the source.
5: Add dependency $\alpha(\bar{y}) \leftrightarrow R(\bar{y})$ to Δ.
6: **end for**
7: **for all** target relation P that is not mentioned in any dependency in Σ **do**
8: Add dependency $\bot \leftrightarrow P(\bar{y})$ to Δ, where $\bar{y}$ is a tuple of distinct fresh variables of the same arity as P.
9: **end for**
10: **return** $\mathcal{M}^{\leftrightarrow} = (\mathbf{S}, \mathbf{T}, \Delta)$

Theorem 1. *Given an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ with Σ a set of full* FO-TO-CQ *dependencies with a single atom in its target side,* BIDIRECTIONALMAPPINGFULL$(\mathcal{M})$ *computes in time $O(\|\Sigma\|^2)$ a mapping $\mathcal{M}^{\leftrightarrow} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of full $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies plus formulas of the form $\forall \bar{x}(\bot \leftrightarrow P(\bar{x}))$, such that $\mathrm{SOL}_{\mathcal{M}^{\leftrightarrow}}(I) = \mathrm{CUS}_{\mathcal{M}}(I)$ for every source instance I.*

Algorithm 1 generates a mapping specified by $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies. An important question is whether all the features used in this fragment are needed in the output of the algorithm. The following result shows that the use of equality in the left-hand side of the dependencies is needed for Theorem 1 to hold, even if the full power of FO is allowed in the source side and disjunctions and equalities are allowed in the target side.

Theorem 2. *There exists an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full st-tgds, such that there is no mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Delta)$, with Δ a set of $\langle \mathrm{FO}, \mathrm{UCQ}^{=} \rangle$ dependencies, such that $\mathrm{SOL}_{\mathcal{M}'}(I) = \mathrm{CUS}_{\mathcal{M}}(I)$ for every source instance I.*

Finally the following result shows that disjunctions in the left-hand side are also necessary in the output of algorithm BIDIRECTIONALMAPPINGFULL.

Theorem 3. *There exists an st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of full st-tgds, such that there is no mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Delta)$, with Δ a set of $\langle \mathrm{CQ}^{=}, \mathrm{FO} \rangle$ dependencies, such that $\mathrm{SOL}_{\mathcal{M}'}(I) = \mathrm{CUS}_{\mathcal{M}}(I)$ for every source instance I.*

4 Complexity

Given that the language of $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies has an interesting expressive power for data exchange, it is worth studying some of its fundamental properties. In this section we study the complexity of the language; in particular, the complexity of the *existence-of-solutions* problem which is defined as follows. Given a fixed mapping $\mathcal{M}$ specified by $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies, the input of the problem is a source instance I, and the question is whether there exists an instance J such that $J \in \mathrm{SOL}_{\mathcal{M}}(I)$ (that is, whether $\mathrm{SOL}_{\mathcal{M}}(I) \neq \emptyset$). Notice that this problem is trivial for the case of st-tgds, but it becomes interesting for the case of bidirectional constraints. For example, consider a mapping $\mathcal{M}$ specified by the set $\{A(x) \leftrightarrow P(x), B(x) \leftrightarrow P(x)\}$, then for the instance $I = \{A(1)\}$ we have that $\mathrm{SOL}_{\mathcal{M}}(I) = \emptyset$.

Our first result establishes the complexity of the existence-of-solutions problem in the general case. In particular, we show that the problem may be intractable.

Theorem 4.
1. *For every mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies, the existence-of-solutions problem is in* NP.
2. *There exists a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies, for which the existence-of-solutions problem is* NP*-complete. Moreover, this even holds if Δ is a set of $\langle \mathrm{CQ}, \mathrm{CQ} \rangle$ dependencies.*

In contrast, if the dependencies do not have existential quantification in the target side, then the existence-of-solutions problem can be efficiently solved.

Theorem 5. *For every mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Delta)$, where Δ is a set of full $\langle \mathrm{UCQ}^{=}, \mathrm{CQ} \rangle$ dependencies, the existence-of-solutions problem is solvable in polynomial time.*

5 Concluding remarks

In this paper we have started the formal study of bidirectional constraints, in particular, their expressive power in terms of their capacity for defining some alternative semantics for data exchange with st-tgds, and the complexity of the existence-of-solutions problem. There has been some related work driven from practical motivations, most notably [13] and [5], where bidirectional constraints based on project-select relational algebra expressions are considered in an object-to-relational mapping system. Nevertheless, to the best of our knowledge this paper presents the first theoretical results on the fundamental properties of bidirectional constraints.

Besides the motivation that comes from defining more strictly what are the possible target instances in data exchange, mappings specified by bidirectional constraints have several other applications. Most notably, these specifications are expressive enough to define how source constraints should be *transformed* into target constraints. If one has a source schema with, for example, key constraints, and creates a new schema and a data exchange setting to migrate the data, it is natural to expect the source key constraints to be somehow reflected in the new schema. This issue is closely related with the recently raised topic of exchanging (or transforming) knowledge bases [2]. Mappings specified by st-tgds are not expressive enough to describe these type of transformations. We argue that bidirectional constraints are a good formalism to study these complex transformation scenarios, and it would be very interesting to explore these possibilities in future work. Another motivation comes from schema mapping management, where the inputs for schema mapping operators such as the *merge*, *extract* and *diff* operators are naturally defined using bidirectional mappings [6,3]. This is also part of our future work.

References

1. F. Afrati and P. Kolaitis. Answering aggregate queries in data exchange. *PODS*, 129-138, 2008.
2. M. Arenas, J. Pérez, and J. Reutter. Data exchange beyond complete data. *JACM*, 60(4), 2013.
3. M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. *PODS*, 227-238, 2010.
4. M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: Bringing exchanged data back. *TODS*, 34(4), 2009.
5. P. A. Bernstein, M. Jacob, J. Pérez, G. Rull, and J. F. Terwilliger. Incremental mapping compilation in an object-to-relational mapping system. *SIGMOD*, 1269-1280, 2013.
6. P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. *SIGMOD*, 1-12, 2007.
7. G. de Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. On reconciling data exchange, data integration, and peer data management. *PODS*, 133-142, 2007.
8. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *TCS*, 336(1): 89-124, 2005.
9. A. Fuxman, P. Kolaitis, R. Miller, and W.-C. Tan. Peer Data Exchange. *TODS*, 31(4), 2006.
10. A. Hernich. Semantics for Non-Monotone Queries in Data Exchange and Data Integration. *Data Exchange, Information, and Streams*, 5: 161-184, 2013.
11. M. Lenzerini. Data integration: a theoretical perspective. *PODS*, 233-246, 2002.
12. L. Libkin. Data exchange and incomplete information. *PODS*, 60-69, 2006.
13. S. Melnik, A. Adya, and P. A. Bernstein. Compiling mappings to bridge applications and databases. *TODS*, 33(4), 2008.