Data Exchange beyond Complete Data

Marcelo Arenas
Department of Computer Science
PUC Chile
marenas@ing.puc.cl

Jorge Pérez
Department of Computer Science
Universidad de Chile
jperez@dcc.uchile.cl

Juan Reutter
School of Informatics
University of Edinburgh
juan.reutter@ed.ad.uk

ABSTRACT

In the traditional data exchange setting, source instances are restricted to be *complete* in the sense that every fact is either true or false in these instances. Although natural for a typical database translation scenario, this restriction is gradually becoming an impediment to the development of a wide range of applications that need to exchange objects that admit several interpretations. In particular, we are motivated by two specific applications that go beyond the usual data exchange scenario: exchanging *incomplete information* and exchanging *knowledge bases*.

In this paper, we propose a general framework for data exchange that can deal with these two applications. More specifically, we address the problem of exchanging information given by *representation systems*, which are essentially finite descriptions of (possibly infinite) sets of complete instances. We make use of the classical semantics of mappings specified by sets of logical sentences to give a meaningful semantics to the notion of exchanging representatives, from which the standard notions of solution, space of solutions, and universal solution naturally arise. We also introduce the notion of *strong representation system for a class of mappings*, that resembles the concept of strong representation system for a query language. We show the robustness of our proposal by applying it to the two applications mentioned above: exchanging incomplete information and exchanging knowledge bases, which are both instantiations of the exchanging problem for representation systems. We study these two applications in detail, presenting results regarding expressiveness, query answering and complexity of computing solutions, and also algorithms to materialize solutions.

Categories and Subject Descriptors
H.2.5 [**Heterogeneous Databases**]: Data translation

General Terms
Algorithms, Theory

Keywords
Data exchange, knowledge exchange, data integration, representation system, metadata management, schema mapping

1. INTRODUCTION

In the typical data exchange setting, one is given a source schema and a target schema, a schema mapping $\mathcal{M}$ that specifies the relationship between the source and the target, and an instance I of the source schema. The basic problem then is how to materialize an instance of the target schema that reflects the source data as accurately as possible [13]. In data exchange terms, the problem is how to materialize the *best solution* for I under $\mathcal{M}$.

In this traditional setting, source instances are restricted to be *complete*: every fact in them is either true or false. While natural in many scenarios, this restriction cannot capture a wide range of applications dealing with objects that admit several interpretations. We are motivated by two such applications: exchanging *incomplete information* and exchanging *knowledge bases*.

Exchanging Incomplete Information. Universal solutions have been proposed as the preferred solutions for data exchange. Given a source instance I and a schema mapping $\mathcal{M}$, a universal solution J is a target instance that represents, in a precise sense, all the possible solutions for I [13]. Even in the scenario in which mappings are specified by source-to-target tuple-generating dependencies (st-tgds), it has been noted that universal solutions need *null* values to correctly reflect the data in the source [13]. Thus, the preferred solutions in this scenario are *incomplete databases* [20]. But what if one needs to exchange data from these instances with null values? What is the semantics of data exchange in this case? This issue has been raised before by Afrati et al. in the context of query answering [2] and also by Fagin et al. in the context of metadata management [16]. But the problem is much wider, as it is not even clear what a *good translation* is for a source instance with null values, even in the most simplest data exchange settings. Just as an example of the questions that need to be answered, given a source instance with null values, is a target instance with null values enough to correctly represent the source information?

Exchanging Knowledge Bases. Nowadays several applications use knowledge bases to represent their data. A prototypical example is the Semantic Web, where repositories store information in the form of RDFS graphs [19] or OWL specifications [24]. In both cases, we have not only data but also *rules* that allow one to infer new data. Thus, in a data exchange application over the Semantic Web, one would naturally have as input a schema mapping and a source specification consisting of data together with some rules, and then one would like to create a target specification materializing data and creating new rules to correctly represent the knowledge in the source. But what does it mean for a target knowledge base to be a valid translation of a source knowledge base? Or, in data exchange terms, when does a target knowledge base can be considered a solution for a source knowledge base under a schema mapping? And more importantly, what constitutes a *good* solution for a source knowledge base? These questions motivate the development of a general *knowledge exchange* framework.

In this paper, we propose a general framework for data exchange that can deal with the above two applications. More specifically, we address the problem of exchanging information given by *represen-*

PODS'11, June 13–15, 2011, Athens, Greece.

tation systems, which are essentially finite descriptions of (possibly infinite) sets of complete instances. We make use of the classical semantics of mappings specified by sets of logical sentences to give a meaningful semantics to the notion of exchanging representatives, thus not altering the usual semantics of schema mappings. From this, the standard notions of solution, space of solutions, and universal solution naturally arise. We also introduce the notion of *strong representation system for a class of mappings*, which resembles the concept of strong representation system for a query language [20, 17]. A strong representation system for a class $\mathcal{C}$ of mapping is, intuitively, a *closed system* in the sense that for every representative $\mathcal{I}$ in the system and mapping $\mathcal{M}$ in $\mathcal{C}$, the space of solutions of $\mathcal{I}$ under $\mathcal{M}$ can be represented in the same system.

As our first application, we study the exchange of incomplete information. One of the main issues when managing incomplete information is that most of its associated tasks, in particular query answering, are considerably harder than in the classical setting of complete data [20, 17]. Thus, it is challenging to find representation systems that are expressive enough to deserve investigation, while admitting efficient procedures for practical data exchange purposes.

In this paper, we study the representation system given by *positive conditional tables*, which are essentially conditional tables [20] that do not use negation. For positive conditional tables we show that, given a mapping specified by st-tgds, it is possible to materialize universal solutions and compute certain answers to unions of conjunctive queries in polynomial time, thus matching the complexity bounds of traditional data exchange. But more importantly, we show that positive conditional tables are expressive enough to form a strong representation system for the class of mappings specified by st-tgds. We prove that this result is optimal in the sense that the main features of positive conditional tables are needed to obtain a strong representation system for this class of mappings. Moreover, we prove that instances with null values, that have been widely used as a representation system in data exchange [13, 21, 2, 22, 16], do not form a strong representation system for the class of mappings specified by st-tgds, and thus, cannot correctly represent the space of solutions of a source instance with null values. Finally, we show that positive conditional instances can be used in schema mapping management to solve some fundamental and problematic issues that arise when combining the *composition* and *inverse* operators [7, 8].

We then apply our framework to knowledge bases. A knowledge base is composed of explicit data, in our context a relational database, plus implicit data given in the form of a set of logical sentences Σ. This set Σ states how to infer new data from the explicit data. The semantics of a knowledge base is given by its set of *models*, which are all the instances that contain the explicit data and satisfy Σ. In this sense, a knowledge base is also a representation system and, thus, can be studied in our general framework. In fact, by applying this framework we introduce the notion of *knowledge exchange*, which is the problem of materializing a target knowledge base that correctly represents the source information. We then study several issues including the complexity of recognizing knowledge-base solutions, the problem of characterizing when a knowledge base can be considered a *good* solution, and the problem of computing such knowledge-base solutions for mappings specified by full st-tgds (which are st-tgds that do not use existential quantification). Our results are a first step towards the development of a general framework for exchanging specifications that are more expressive than the usual database instances. In particular, this framework can be used in the exchange of RDFS graphs and OWL specifications, a problem becoming more and more important in Semantic Web applications.

We have structured the paper into three parts. We present some terminology and our general exchange framework for representation systems in Sections 2 and 3. In Sections 4, 5 and 6, we present our results regarding the exchange of incomplete information. Finally, in Sections 7 and 8, we introduce and study the problem of exchanging knowledge bases.

2. PRELIMINARIES

A *schema* $\mathbf{S}$ is a finite set $\{R_1, \ldots, R_k\}$ of relation symbols, with each R_i having a fixed arity $n_i \geq 0$. Let $\mathbf{D}$ be a countably infinite domain. An instance I of $\mathbf{S}$ assigns to each relation symbol R_i of $\mathbf{S}$ a finite relation $R_i^I \subseteq \mathbf{D}^{n_i}$. $\textsc{Inst}(\mathbf{S})$ denotes the set of all instances of $\mathbf{S}$. We denote by $\mathrm{dom}(I)$ the set of all elements that occur in any of the relations R_i^I. We say that $R_i(t)$ is a fact of I if $t \in R_i^I$. We sometimes denote an instance by its set of facts.

Given schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, a *schema mapping* (or just *mapping*) from $\mathbf{S}_1$ to $\mathbf{S}_2$ is a subset of $\textsc{Inst}(\mathbf{S}_1) \times \textsc{Inst}(\mathbf{S}_2)$. We say that J is a *solution for I under $\mathcal{M}$* whenever $(I, J) \in \mathcal{M}$. The set of all solutions for I under $\mathcal{M}$ is denoted by $\textsc{Sol}_{\mathcal{M}}(I)$. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be schemas with no relation symbols in common and Σ a set of first-order logic (FO) sentences over $\mathbf{S}_1 \cup \mathbf{S}_2$. A mapping $\mathcal{M}$ from $\mathbf{S}_1$ to $\mathbf{S}_2$ is *specified* by Σ, denoted by $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma)$, if for every $(I, J) \in \textsc{Inst}(\mathbf{S}_1) \times \textsc{Inst}(\mathbf{S}_2)$, we have that $(I, J) \in \mathcal{M}$ if and only if (I, J) satisfies Σ. Notice that mappings are binary relations, and thus we can define the composition of mappings as for the composition of binary relations. Let $\mathcal{M}_{12}$ be a mapping from schema $\mathbf{S}_1$ to schema $\mathbf{S}_2$ and $\mathcal{M}_{23}$ a mapping from $\mathbf{S}_2$ to schema $\mathbf{S}_3$. Then $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is a mapping from $\mathbf{S}_1$ to $\mathbf{S}_3$ given by the set $\{(I, J) \in \textsc{Inst}(\mathbf{S}_1) \times \textsc{Inst}(\mathbf{S}_3) \mid \text{there exists } K \text{ such that } (I, K) \in \mathcal{M}_{12} \text{ and } (K, J) \in \mathcal{M}_{23}\}$ [14].

Dependencies: A relational atom over $\mathbf{S}$ is a formula of the form $R(\bar{x})$ with $R \in \mathbf{S}$ and $\bar{x}$ a tuple of (not necessarily distinct) variables. A tuple-generating dependency (tgd) over a schema $\mathbf{S}$ is a sentence of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where $\varphi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are conjunctions of relational atoms over $\mathbf{S}$. The left-hand side of the implication in a tgd is called the premise, and the right-hand side the conclusion. A *full tgd* is a tgd with no existentially quantified variables in its conclusion. We usually omit the universal quantifier when writing tgds.

Given disjoint schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, a *source-to-target tgd* (st-tgd) from $\mathbf{S}_1$ to $\mathbf{S}_2$ is a tgd in which the premise is a formula over $\mathbf{S}_1$ and the conclusion is a formula over $\mathbf{S}_2$. As for the case of full tgds, a full st-tgd is an st-tgd with no existentially quantified variables in its conclusion. In this paper, we assume that all sets of dependencies are finite.

Queries and certain answers: A k-ary query Q over a schema $\mathbf{S}$, with $k \geq 0$, is a function that maps every instance $I \in \textsc{Inst}(\mathbf{S})$ into a k-relation $Q(I) \subseteq \mathrm{dom}(I)^k$. In this paper, CQ is the class of conjunctive queries and UCQ is the class of unions of conjunctive queries. If we extend these classes by allowing equalities or inequalities, then we use superscripts $=$ and $\neq$, respectively. Thus, for example, $\text{UCQ}^{\neq}$ is the class of union of conjunctive queries with inequalities. Let $\mathcal{M}$ be a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, I an instance of $\mathbf{S}_1$ and Q a query over $\mathbf{S}_2$. Then $\mathrm{certain}_{\mathcal{M}}(Q, I)$ denotes the set of *certain answers* of Q over I under $\mathcal{M}$, that is, $\mathrm{certain}_{\mathcal{M}}(Q, I) = \bigcap_{J \in \textsc{Sol}_{\mathcal{M}}(I)} Q(J)$.

3. SCHEMA MAPPINGS AND REPRESENTATION SYSTEMS

A (usual) database instance I is said to contain complete information as every fact $R(t)$ is either true or false in I. In that sense, there is a single possible interpretation of the information in I. On

the other hand, in a database instance with incomplete information some values are unknown (which are usually represented by null values) and, hence, one is not certain about its content. In that sense, one has several possible interpretations for the information in such instances. In the same spirit, a knowledge base usually has several models, which represent different ways to interpret the rules or axioms in the knowledge base. In this section, we present the notion of *representation system* [20, 3], which is a general way to deal with objects that admit different interpretations, and then we show how to extend a schema mapping to deal with representation systems. This extension is fundamental for our study as it allows us to extend, in a simple and natural way, the data exchange framework proposed by Fagin et al. [13] to the case of database instances with incomplete information as well as to the case of knowledge bases.

3.1 Exchanging information given by representation systems

A *representation system* is composed of a set $\mathbf{W}$ of *representatives* and a function rep that assigns a set of instances to every element in $\mathbf{W}$. We assume that every representation system $(\mathbf{W}, \text{rep})$ is uniform in the sense that for every $\mathcal{W} \in \mathbf{W}$, there exists a relational schema $\mathbf{S}$, that is called the type of $\mathcal{W}$, such that $\text{rep}(\mathcal{W}) \subseteq \text{INST}(\mathbf{S})$ [20]. Representation systems are used to describe sets of possible interpretations in a succinct way. Typical examples of representation systems are Codd tables, naive tables, conditional tables [20], and world-set decompositions [3].

Assume that $\mathcal{M}$ is a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$. Given a set $\mathcal{X}$ of instances of $\mathbf{S}_1$, define $\text{SOL}_{\mathcal{M}}(\mathcal{X})$ as $\bigcup_{I \in \mathcal{X}} \text{SOL}_{\mathcal{M}}(I)$. That is, $\text{SOL}_{\mathcal{M}}(\mathcal{X})$ is the set of possible solutions for the instances in $\mathcal{X}$. In the following definition, we use $\text{SOL}_{\mathcal{M}}(\cdot)$ to extend the notion of solution to the case of representation systems.

Definition 3.1 *Let $\mathcal{R} = (\mathbf{W}, rep)$ be a representation system, $\mathcal{M}$ a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ and $\mathcal{V}, \mathcal{W}$ elements of $\mathbf{W}$ of types $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. Then $\mathcal{W}$ is an $\mathcal{R}$-solution for $\mathcal{V}$ under $\mathcal{M}$ if $rep(\mathcal{W}) \subseteq \text{SOL}_{\mathcal{M}}(rep(\mathcal{V}))$.*

In other words, given a representation system $\mathcal{R} = (\mathbf{W}, \text{rep})$ and $\mathcal{V}, \mathcal{W} \in \mathbf{W}$, it holds that $\mathcal{W}$ is an $\mathcal{R}$-solution for $\mathcal{V}$ under a mapping $\mathcal{M}$ if for every $J \in \text{rep}(\mathcal{W})$, there exists $I \in \text{rep}(\mathcal{V})$ such that $(I, J) \in \mathcal{M}$.

Assume given a representation system $\mathcal{R} = (\mathbf{W}, \text{rep})$ and a mapping $\mathcal{M}$. An element of $\mathbf{W}$ can have a large number of $\mathcal{R}$-solutions under $\mathcal{M}$, even an infinite number in some cases, and, thus, it is natural to ask what is a *good* solution for this element under $\mathcal{M}$. Next we introduce the notion of universal $\mathcal{R}$-solution, which is a simple extension of the concept of $\mathcal{R}$-solution introduced in Definition 3.1.

Definition 3.2 *Let $\mathcal{R} = (\mathbf{W}, rep)$ be a representation system, $\mathcal{M}$ a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ and $\mathcal{V}, \mathcal{W}$ elements of $\mathbf{W}$ of types $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. Then $\mathcal{W}$ is a universal $\mathcal{R}$-solution for $\mathcal{V}$ under $\mathcal{M}$ if $rep(\mathcal{W}) = \text{SOL}_{\mathcal{M}}(rep(\mathcal{V}))$.*

This new notion captures the intuition of exactly representing the space of possible solutions of the interpretations of an element of a representation system.

3.2 Strong representation systems for a class of mappings

The classical work on incomplete databases [20] defines the notion of *strong representation system* for a class of queries. In fact, the classical result in [20] about these systems states that conditional tables are a strong representation system for relational algebra. In our context, we are interested in defining the notion of strong representation system for a class of mappings.

Definition 3.3 *Let $\mathcal{C}$ be a class of mappings and $(\mathbf{W}, rep)$ a representation system. Then $(\mathbf{W}, rep)$ is a strong representation system for $\mathcal{C}$ if for every mapping $\mathcal{M} \in \mathcal{C}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, and for every $\mathcal{U} \in \mathbf{W}$ of type $\mathbf{S}_1$, there exists $\mathcal{W} \in \mathbf{W}$ of type $\mathbf{S}_2$ such that $rep(\mathcal{W}) = \text{SOL}_{\mathcal{M}}(rep(\mathcal{U}))$.*

In other words, a representation system $\mathcal{R} = (\mathbf{W}, \text{rep})$ is a strong representation system for a class of mappings $\mathcal{C}$ if for every mapping $\mathcal{M} \in \mathcal{C}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, and for every $\mathcal{U} \in \mathbf{W}$ of type $\mathbf{S}_1$, a universal $\mathcal{R}$-solution for $\mathcal{U}$ under $\mathcal{M}$ can be represented in the same system (it is an element of $\mathbf{W}$). Notice that if $\mathcal{C}$ allows for mappings in which no solution exist for some of their instances, then any strong representation system for $\mathcal{C}$ must be able to represent the *empty* set of instances.

4. STRONG REPRESENTATION SYSTEMS FOR ST-TGDS

As pointed out before, one of the goals of this paper is to study the problem of exchanging databases with incomplete information. To this end, we first borrowed from [20, 3] the notion of representation system, which gives us a way to represent databases with incomplete information, and then we introduced the notion of strong representation system for a class of mappings, which essentially tell us that a particular way of representing databases with incomplete information is appropriate for a class of mappings. In this section, we apply these concepts to the widely used class of mappings specified by st-tgds and, in particular, we answer the question of what is a good representation system for this class. Notice that our mappings only contain instances with complete information. Thus, as opposed to previous work [13, 21, 16], we make a clear distinction between instances participating in a mapping and incomplete instances that are used as representatives for spaces of solutions.

The starting points for our study are naive tables, which are widely used in the data exchange context [13, 21, 16], and conditional tables, which are known to be an expressive way to represent databases with incomplete information [20]. In Section 4.1, we define the representation systems based on these two types of tables, together with a representation system based on *positive* conditional tables, a fragment of conditional tables proposed in this paper. In Section 4.2, we show that both conditional tables and positive conditional tables form strong representation systems for the class of mappings given by st-tgds, and we also show that naive tables are not expressive enough to form such a system. Finally, in Section 4.3, we give strong evidence that positive conditional tables are the right representation system for the class of mappings specified by st-tgds, by proving that the main features in these instances are needed to obtain a strong representation system for this class.

4.1 Naive and conditional instances

Our database instances are constructed by using elements of a countably infinite set $\mathbf{D}$. To represent incomplete information we assume the existence of a countably infinite set $\mathbf{N}$ of *labeled nulls*, disjoint with $\mathbf{D}$. To differentiate from nulls we call *constant values* the elements in $\mathbf{D}$. Fix a relational schema $\mathbf{S}$ for this section. Then a *naive instance* $\mathcal{I}$ of $\mathbf{S}$ assigns to each relation symbol $R \in \mathbf{S}$ of arity k, a finite k-ary relation $R^{\mathcal{I}} \subseteq (\mathbf{D} \cup \mathbf{N})^k$, that is, a k-ary relation including constants and null values. A *conditional instance* extends the notion of naive instance with a *local* condition attached to each fact. More precisely, an *element-condition* is a positive Boolean combination (only connectives $\wedge$ and $\vee$ are allowed) of formulas of the form $x = y$ and $x \neq y$, with $x \in \mathbf{N}$ and $y \in (\mathbf{D} \cup$

$\mathbf{N}$). Then a conditional instance $\mathcal{I}$ of $\mathbf{S}$ assigns to each relation symbol R of arity k, a pair $(R^{\mathcal{I}}, \rho_R^{\mathcal{I}})$, where $R^{\mathcal{I}} \subseteq (\mathbf{D} \cup \mathbf{N})^k$ and $\rho_R^{\mathcal{I}}$ is a function that associates to each tuple $t \in R^{\mathcal{I}}$ an element-condition $\rho_R^{\mathcal{I}}(t)$ (the local condition of the fact $R(t)$ [20]).

To define the sets of interpretations associated to naive and conditional instances, we need to introduce some terminology. Given a naive or conditional instance $\mathcal{I}$, define $\mathrm{nulls}(\mathcal{I})$ as the set of nulls mentioned in $\mathcal{I}$ (if $\mathcal{I}$ is a conditional instance, $\mathrm{nulls}(\mathcal{I})$ also includes the nulls mentioned in the local conditions of $\mathcal{I}$). Moreover, given a null substitution $\nu : \mathrm{nulls}(\mathcal{I}) \to \mathbf{D}$, define $\nu(R^{\mathcal{I}}) = \{\nu(t) \mid t \in R^{\mathcal{I}}\}$, where $\nu(t)$ is obtained by replacing every null n in t by its image $\nu(n)$. Then for every naive instance $\mathcal{I}$, and slightly abusing notation, define the set of representatives of $\mathcal{I}$, denoted by $\mathrm{rep}_{\text{naive}}(\mathcal{I})$, as:

$$\{I \in \textsc{Inst}(\mathbf{S}) \mid \text{there exists } \nu : \mathrm{nulls}(\mathcal{I}) \to \mathbf{D} \text{ such that for every } R \in \mathbf{S}, \text{ it holds that } \nu(R^{\mathcal{I}}) \subseteq R^I\}$$

Moreover, for every conditional instance $\mathcal{I}$, define the set of representatives of $\mathcal{I}$, denoted by $\mathrm{rep}_{\text{cond}}(\mathcal{I})$, as follows. Given an element-condition φ and a null substitution $\nu : V \to \mathbf{D}$, where V is a set of nulls that contains every null value mentioned in φ, notation $\nu \models \varphi$ is used to indicate that ν satisfies φ in the usual sense. Moreover, given a null substitution $\nu : \mathrm{nulls}(\mathcal{I}) \to \mathbf{D}$ and $R \in \mathbf{S}$, define $\nu(R^{\mathcal{I}}, \rho_R^{\mathcal{I}})$ as $\{\nu(t) \mid t \in R^{\mathcal{I}} \text{ and } \nu \models \rho_R^{\mathcal{I}}(t)\}$. Then $\mathrm{rep}_{\text{cond}}(\mathcal{I})$ is the following set of instances:

$$\{I \in \textsc{Inst}(\mathbf{S}) \mid \text{there exists } \nu : \mathrm{nulls}(\mathcal{I}) \to \mathbf{D} \text{ such that for every } R \in \mathbf{S}, \text{ it holds that } \nu(R^{\mathcal{I}}, \rho_R^{\mathcal{I}}) \subseteq R^I\}.$$

We use $\mathrm{rep}_{\text{naive}}$ and $\mathrm{rep}_{\text{cond}}$ to define two fundamental representation systems. Assume that $\mathbf{W}_{\text{naive}}$ and $\mathbf{W}_{\text{cond}}$ are the set of all possible naive instances and conditional instances (over all possible relational schemas), respectively. Then $\mathcal{R}_{\text{naive}} = (\mathbf{W}_{\text{naive}}, \mathrm{rep}_{\text{naive}})$ and $\mathcal{R}_{\text{cond}} = (\mathbf{W}_{\text{cond}}, \mathrm{rep}_{\text{cond}})$ are representation systems.

We conclude this section by introducing a fragment of the class of conditional instances that will be extensively used in this paper. We say that an element-condition is *positive* if it does not mention any formula of the form $x \neq y$. Then a conditional instance $\mathcal{I}$ of $\mathbf{S}$ is said to be positive if for every $R \in \mathbf{S}$ and $t \in R^{\mathcal{I}}$, it holds that $\rho_R^{\mathcal{I}}(t)$ is a positive element-condition. We denote by $\mathbf{W}_{\text{pos}}$ the set of all positive conditional instances, by $\mathrm{rep}_{\text{pos}}$ the restriction of function $\mathrm{rep}_{\text{cond}}$ to the class of positive conditional instances, and by $\mathcal{R}_{\text{pos}}$ the representation system $(\mathbf{W}_{\text{pos}}, \mathrm{rep}_{\text{pos}})$. When it is clear from the context, we just use rep instead of $\mathrm{rep}_{\text{naive}}$, $\mathrm{rep}_{\text{cond}}$ or $\mathrm{rep}_{\text{pos}}$.

4.2 Building a strong representation system for st-tgds

Fagin et al. showed in [13] that for the class of mappings specified by st-tgds, naive instances are enough to represent the space of solutions of any complete database. More precisely, assuming that $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, we have that for every instance I_1 of $\mathbf{S}_1$, there exists a naive instance $\mathcal{I}_2$ of $\mathbf{S}_2$ such that $\mathrm{rep}(\mathcal{I}_2) = \textsc{Sol}_{\mathcal{M}}(I_1)$. Thus, given that the target data generated by a mapping can be used as the source data in other mappings, it is natural to ask whether the same result holds when naive instances are considered as source instances. That is, it is natural to ask whether for every mapping $\mathcal{M}$ specified by a set of st-tgds and for every source naive instance $\mathcal{I}_1$, there exists a target naive instance $\mathcal{I}_2$ such that $\mathrm{rep}(\mathcal{I}_2) = \textsc{Sol}_{\mathcal{M}}(\mathrm{rep}(\mathcal{I}_1))$. Unfortunately, the following example shows that it is not the case.

Example 4.1. Let $\mathbf{S}_1 = \{P(\cdot,\cdot)\}$, $\mathbf{S}_2 = \{T(\cdot), R(\cdot,\cdot)\}$ and Σ_{12} a set consisting of the following st-tgds:

$$\begin{aligned} P(x,y) &\rightarrow R(x,y), \\ P(x,x) &\rightarrow T(x). \end{aligned}$$

Moreover, let $\mathcal{I}$ be a naive instance of $\mathbf{S}_1$ such that $P^{\mathcal{I}} = \{(n,a)\}$, where n is a null value and a is a constant. It is not difficult to prove that if $\mathcal{J}$ is a naive instance of $\mathbf{S}_2$, then $\mathrm{rep}(\mathcal{J}) \neq \textsc{Sol}_{\mathcal{M}}(\mathrm{rep}(\mathcal{I}))$. The reason for this is that a naive instance cannot represent the fact that if n is given value a in some representative of $\mathcal{I}$, then $T(a)$ holds in every solution for that representative. □

From the previous example, we conclude that:

Proposition 4.2 *Naive instances do not form a strong representation system for the class of mappings specified by st-tgds.*

What should be added to naive instances to obtain a strong representation system for the class of mapping given by st-tgds? A natural candidate are the local conditions presented in Section 4.1, as shown in the following example. In this example, and in the rest of the paper, we assume that $\top$ is an arbitrary element-condition that always holds (for example, $n = n$ with $n \in \mathbf{N}$).

Example 4.3. Let $\mathcal{M}$ and $\mathcal{I}$ be as in Example 4.1, and $\mathcal{J}$ be a conditional instance that contains the following facts and conditions in the relations R and T:

$R(n,a)$	$\top$
$T(n)$	$n = a$

Then it can be proved that $\mathrm{rep}(\mathcal{J}) = \textsc{Sol}_{\mathcal{M}}(\mathrm{rep}(\mathcal{I}))$. □

In the previous example, we use only positive element-conditions to represent the space of solutions of the source naive instance. Thus, it is natural to ask whether this is a general phenomenon, or whether one needs to consider non-positive element-conditions of the form $x \neq y$ to find a strong representation system for the class of mappings specified by st-tgds. In the following theorem, we prove that positive conditions are indeed enough.

Theorem 4.4 *Positive conditional instances form a strong representation system for the class of mappings specified by st-tgds.*

We conclude this section by showing that conditional instances also form a strong representation system for the class of mappings specified by st-tgds, thus giving us an alternative system to deal with incomplete information in schema mappings.

Theorem 4.5 *Conditional instances form a strong representation system for the class of mappings specified by st-tgds.*

4.3 Positive conditional instances are the needed fragment

In the previous section, we show that both conditional instances and positive conditional instances form strong representation systems for the class of mappings specified by st-tgds. Given these alternatives, it is natural to ask whether there exist other possible strong representation systems for this class of mappings and which one could be considered as the *right* system for this class. In this section, we give strong evidence that positive conditional instances are the right representation system for mappings specified by st-tgds, by proving that the main features in these instances are needed to obtain a strong representation system for this class of mappings.

In a positive conditional instance, a local condition is attached to each fact. The distinctive features of these local conditions are the use of disjunction, equalities of the form $n_1 = n_2$, with $n_1, n_2 \in \mathbf{N}$, and equalities of the form $n = c$, with $n \in \mathbf{N}$ and $c \in \mathbf{D}$. In this section, we show that if one removes any of these features and keeps the other two, then the resulting representation

system does not form a strong representation system for the class of mappings specified by st-tgds. More precisely, given a positive conditional instance $\mathcal{I}$ of a relational schema $\mathbf{S}$, we say that $\mathcal{I}$ is null-comparison free if no local condition in $\mathcal{I}$ mentions a formula of the form $n_1 = n_2$ with $n_1, n_2 \in \mathbf{N}$, and we say that $\mathcal{I}$ is null-constant-comparison free if no local condition in $\mathcal{I}$ mentions a formula of the form $n = c$ with $n \in \mathbf{N}$ and $c \in \mathbf{D}$. Moreover, we say that $\mathcal{I}$ is disjunction free if for every $R \in \mathbf{S}$ and $t \in R^{\mathcal{I}}$, it holds that $\rho_R^{\mathcal{I}}(t)$ does not mention Boolean connective $\vee$.

Theorem 4.6 *None of the following form a strong representation system for the class of mappings specified by st-tgds: (1) null-comparison free positive conditional instances, (2) null-constant-comparison free positive conditional instances and (3) disjunction free positive conditional instances.*

5. DATA EXCHANGE WITH POSITIVE CONDITIONAL INSTANCES

In the data exchange setting, one is given a mapping $\mathcal{M}$ from a source schema to a target schema and a source instance I, and the goal is to materialize a solution J for I under $\mathcal{M}$. This setting has been widely studied in the literature, where many important problems have been addressed in order to develop data exchange tools. In this section, we focus on three of the most important tasks in data exchange: materializing solutions, computing certain answers to target queries, and checking whether a target instance is a solution for a source instance under a mapping [13], and show how these tasks are performed in the presence of positive conditional instances. In particular, we prove that the fundamental problems of materializing solutions and computing certain answers to target queries can be solved efficiently in this extended data exchange scenario, thus showing that positive conditional instances not only allow a uniform way of dealing with the exchange of incomplete information, but also that they can be used in practice.

5.1 Materializing solutions

The most important problem in data exchange is the problem of materializing a *good* solution for a given source instance. For mappings specified with st-tgds, these are the *universal solutions*. Polynomial-time algorithms have been developed to compute these solutions [13], which have allowed the construction of practical data exchange tools. In the context of a representation system $\mathcal{R}$, universal $\mathcal{R}$-solutions are the preferred option as they are able to exactly represent the spaces of solutions of the source data. Thus, to show that positive conditional instances can be used in practice, one needs to develop an efficient algorithm for computing universal $\mathcal{R}_{\text{pos}}$-solutions. In the following theorem, we show that such an algorithms exists.

Theorem 5.1 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then there exists a polynomial-time algorithm that, given a positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$, computes a universal $\mathcal{R}_{pos}$-solution for $\mathcal{I}$ under $\mathcal{M}$.*

It is important to notice that in the previous result the set of st-tgds defining a schema mapping is assumed to be fixed. This is the usual assumption when studying the complexity of materializing solutions in data exchange [13].

The algorithm in Theorem 5.1 is based on the *chase* procedure, as usual in data exchange [13]. In particular, our procedure is based on the chase algorithms presented in [17], and is similar to the one recently proposed in [18]. Notice that, as shown in Section 4.2, the straightforward application of the chase may not deliver the expected result, as naive instances do not form a strong representation system for the class of mappings specified by st-tgds. Thus, one needs to modify the chase procedure to take into consideration the element-conditions in positive conditional instances. In particular, one has to consider that some relationships between null values can fire the application of a dependency, and one has to make explicit these relationships in the generated tuples by using new element-conditions. Due to the lack of space, we do not present this chase procedure in detail but show the basic ideas behind our algorithm in the following example.

Example 5.2. Let $\mathbf{S}_1 = \{P(\cdot,\cdot), R(\cdot,\cdot)\}$, $\mathbf{S}_2 = \{S(\cdot,\cdot), T(\cdot)\}$ and Σ_{12} a set consisting of the following st-tgds:

$$\begin{aligned} P(x,y) &\rightarrow S(x,y), \\ R(x,x) &\rightarrow T(x). \end{aligned}$$

Moreover, let $\mathcal{I}$ be a positive conditional instance given by:

$$\begin{array}{ll} P(n_1,n_2) & \top \\ R(n_1,n_2) & (n_1 = a), \end{array}$$

where n_1 and n_2 are null values and a is a constant. To create a universal $\mathcal{R}_{\text{pos}}$-solution $\mathcal{J}$, the procedure works as follows. For the first dependency, it works as the classical chase, that is, it adds tuple (n_1, n_2) to $S^{\mathcal{J}}$ with $\top$ as local condition. For the second dependency, the procedure considers that this dependency should be fired when condition $n_1 = n_2$ holds. In this case, the procedure also needs to carry along the element-condition $n_1 = a$. Thus, it adds the tuple (n_1) to $T^{\mathcal{J}}$, but this time the local condition consists of the conjunction of $(n_1 = a)$ with $(n_1 = n_2)$. Summing up, the following instance is constructed:

$$\begin{array}{ll} S(n_1,n_2) & \top \\ T(n_1) & (n_1 = a) \wedge (n_1 = n_2) \end{array}$$

It can be shown that this instance is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$. □

5.2 Computing certain answers

A second fundamental problem in data exchange is the task of computing certain answers to target queries. In our context, this problem is defined as follows. Given a positive conditional instance $\mathcal{I}$ of a schema $\mathbf{S}$ and a query Q over $\mathbf{S}$, define $Q(\mathcal{I})$ as $\bigcap_{I \in \text{rep}(\mathcal{I})} Q(I)$. Moreover, given a mapping $\mathcal{M}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, a positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$ and a query Q over $\mathbf{S}_2$, the set of certain answers of Q over $\mathcal{I}$ under $\mathcal{M}$, denoted by $\text{certain}_{\mathcal{M}}(Q, \mathcal{I})$, is defined as:

$$\bigcap_{\mathcal{J} \,:\, \mathcal{J} \text{ is an } \mathcal{R}_{\text{pos}}\text{-solution for } \mathcal{I} \text{ under } \mathcal{M}} Q(\mathcal{J}).$$

It should be noticed that this definition of certain answers, in the presence of an incomplete source instance $\mathcal{I}$, coincides with the definition in [2] for the case of naive instances (which is the representation system used in [2]).

Given a data exchange setting $\mathcal{M}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, and a k-ary query Q over $\mathbf{S}_2$, we consider in this section the following decision problem:

Problem:	CERTAINANSWERS$(\mathcal{M}, Q)$
Input:	A positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$ and a k-tuple t of elements from $\mathbf{D}$.
Question:	Does t belong to $\text{certain}_{\mathcal{M}}(Q, \mathcal{I})$?

In the previous problem, we assume that the data exchange setting $\mathcal{M}$ and the query Q are fixed. Thus, we are interested in the data complexity of the problem of computing certain answers.

It was proved in [13] that for the class of mappings specified by st-tgds, each universal solution of an instance I can be directly used to compute the certain answers of any unions of conjunctive queries. In the following proposition, we show that this result can be extended to any query if one considers universal $\mathcal{R}_{\text{pos}}$-solutions.

Proposition 5.3 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, $\mathcal{I}$ a positive conditional instance of $\mathbf{S}_1$ and Q an arbitrary query over $\mathbf{S}_2$. Then for every universal $\mathcal{R}_{pos}$-solution $\mathcal{J}$ for $\mathcal{I}$ under $\mathcal{M}$, it holds that $\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) = Q(\mathcal{J})$.*

In Theorem 5.1, we showed that if a mapping $\mathcal{M}$ is specified by a set of st-tgds, then there exists a polynomial time algorithm that, given a source instance $\mathcal{I}$, computes a universal $\mathcal{R}_{\text{pos}}$-solution $\mathcal{J}$ for $\mathcal{I}$ under $\mathcal{M}$. Moreover, from the results in [17], it is possible to conclude that for every unions of conjunctive queries Q, there exists a polynomial time algorithm that, given a positive conditional instance $\mathcal{I}$, computes $Q(\mathcal{I})$. From these results, we conclude that:

Theorem 5.4 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, and Q be a union of conjunctive queries over $\mathbf{S}_2$. Then* CERTAINANSWERS$(\mathcal{M}, Q)$ *can be solved in polynomial time.*

This result matches the upper bound in [13] for the problem of computing certain answers to a union of conjunctive queries in a usual data exchange setting, thus giving more evidence that positive conditional instances can be used in practical data exchange tools.

We conclude this section by pointing out that Fagin et al. also showed in [13] that the above polynomial-time upper bound holds if one considers unions of conjunctive queries with at most one inequality per disjunct. Here we show the corresponding result for our framework, which is proved by a nontrivial extension of the techniques in [13] for the case of positive conditional instances.

Theorem 5.5 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, and Q be a union of conjunctive queries over $\mathbf{S}_2$ with at most one inequality per disjunct. Then* CERTAINANSWERS$(\mathcal{M}, Q)$ *can be solved in polynomial time.*

5.3 Complexity of recognizing solutions

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{R} = (\mathbf{W}, \text{rep})$ a representation system. In this section, we study the complexity of verifying, given a pair $(\mathcal{U}, \mathcal{W})$ of representatives, whether $\mathcal{W}$ is an $\mathcal{R}$-solution of $\mathcal{U}$ under a mapping $\mathcal{M}$. That is, we consider the following decision problem:

Problem:	CHECKSOLUTION$(\mathcal{M}, \mathcal{R})$
Input:	A pair of representatives $\mathcal{U}, \mathcal{W} \in \mathbf{W}$ of types $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively.
Question:	Is $\mathcal{W}$ an $\mathcal{R}$-solution for $\mathcal{U}$ under $\mathcal{M}$?

In a traditional data exchange setting, deciding whether an instance J is a solution for I under a fixed mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ can be solved by checking if $(I, J) \models \Sigma_{12}$, which gives a straightforward polynomial-time procedure when Σ_{12} is a set of FO sentences. For the case of naive, positive conditional and conditional instances, this problem becomes more interesting. Our first result shows that the complexity for positive conditional instances is no more than for naive instances, in both cases NP.

Theorem 5.6 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then* CHECKSOLUTION$(\mathcal{M}, \mathcal{R}_{naive})$ *and* CHECKSOLUTION$(\mathcal{M}, \mathcal{R}_{pos})$ *are both NP-complete.*

The following theorem shows that the complexity of the problem is higher for conditional instances. This gives evidence in favor of using positive conditional instances instead of conditional instances as a representation system for st-tgds.

Theorem 5.7 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then* CHECKSOLUTION$(\mathcal{M}, \mathcal{R}_{cond})$ *is Π_2^P-complete.*

It should be noticed that this result cannot be directly obtained from [1], since in that paper conditional instances allow *global conditions* that we do not consider in this paper.

6. METADATA MANAGEMENT WITH POSITIVE CONDITIONAL INSTANCES

In the previous sections, we have presented a number of results that give evidence that positive conditional instances are appropriate for data exchange purposes. In this section, we give a step forward in this direction, and show that they are also appropriate for metadata management purposes.

In the data exchange setting proposed by Fagin et al in [13] two types of schemas are considered: source and target schemas. In the former, only the usual instances with complete information are allowed, while in the latter naive instances are also considered. This setting has played a key role in the study and development of schema mapping operators, which are of fundamental importance in metadata management [7, 8].

Two of the most fundamental operations in metadata management are the *composition* and *inversion* of schema mappings. The problem of composing schema mappings was solved in [14] for the class of mappings specified by st-tgds. More precisely, Fagin et al. proposed in [14] the language of SO tgds (see Section 6.1 for a formal definition of this language), and showed that it is the *minimal* class of mappings capable of expressing the composition of mappings specified by st-tgds [14]. On the other hand, the definition of an inverse operator has turned out to be a very difficult problem, and even the definition of a *good* semantics for this operator has been the main topic of several papers in the area [12, 15, 6, 16, 5]. Furthermore, people have also realized that the composition and inverse operators have to be used together in many metadata management tasks, such as schema evolution [8]. This has brought more complexity into the picture, as the combined use of the composition and inverse operators requires that the target data generated by a mapping could be used by other mappings as the source data. This was recognized by Fagin et al. in [16], where the notion of inversion proposed in [6] was extended to deal with source naive instances. Nevertheless, even though the language of SO tgds has proved to be the right language for composing mappings specified by st-tgds, none of the proposed inverse operators has been properly assembled with the language of SO tgds. Indeed, SO tgds do not always admit an inverse under the notions of inversion defined in [12, 15, 6, 5], and it is not clear whether the notion of inversion introduced in [16] is appropriate for the language of SO tgds.

Why does the problem of combining the composition and inverse operators seem to be so difficult? We give strong evidence here that the reason is that naive instances are not expressive enough to deal with the spaces of solutions of SO tgds. But, most significantly, we show here that positive conditional instances can be used to overcome this limitation, as we prove that they form a strong representation system for the class of mappings given by SO tgds, and that SO tgds admit an inverse under the notion proposed in [6], if positive conditional instances are allowed in source and target schemas. It remains an open problem to show whether this inverse can always be specified with an SO tgd or not.

6.1 Positive conditional instances form a strong representation system for SO-tgds

A fundamental tool in the study of the composition of schema mappings is the language of second-order st-tgds (SO tgds [14]),

that we define next. Given schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, an SO tgd from $\mathbf{S}_1$ to $\mathbf{S}_2$ is a second-order formula of the form:

$$\exists f_1 \cdots \exists f_\ell \left(\forall \bar{x}_1(\varphi_1 \rightarrow \psi_1) \wedge \cdots \wedge \forall \bar{x}_n(\varphi_n \rightarrow \psi_n)\right),$$

where (1) each f_i is a function symbol, (2) each φ_i is a conjunction of relational atomic formulas of $\mathbf{S}_1$ and equality atoms of the form $t = t'$, where t and t' are terms built from $\bar{x}_i$ and $f_1, \ldots, f_\ell$, (3) each ψ_i is a conjunction of relational atomic formulas of $\mathbf{S}_2$ mentioning terms built from $\bar{x}_i$ and $f_1, \ldots, f_\ell$, and (4) each variable in $\bar{x}_i$ appears in some relational atomic formula of φ_i. For example, the following is an SO tgd:

$$\exists f \left(\forall x\,(E(x) \rightarrow R(x, f(x))) \wedge \forall x\,(E(x) \wedge x = f(x) \rightarrow T(x))\right). \quad (1)$$

A mapping $\mathcal{M}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ is said to be specified by an SO tgd σ_{12} from $\mathbf{S}_1$ to $\mathbf{S}_2$ [*], denoted by $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, if for every pair of instances I_1, I_2 of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, it holds that $(I_1, I_2) \in \mathcal{M}$ if and only if (I_1, I_2) satisfies σ_{12} in the usual second-order logic sense (see [14] for a precise definition of the semantics of SO tgds).

As our first result, we show that one can efficiently materialize positive conditional instances for a mapping given by an SO tgd.

Theorem 6.1 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, where σ_{12} is an SO tgd. Then there exists a polynomial-time algorithm, that given a positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$, computes a universal $\mathcal{R}_{pos}$-solution for $\mathcal{I}$ under $\mathcal{M}$.*

As a corollary, we obtain that positive conditional instances are appropriate for representing the spaces of solutions of SO tgds.

Corollary 6.2 *Positive conditional instances form a strong representation system for the class of mappings specified by SO tgds.*

An important remark about the previous results is that they follow directly from the fact that positive conditional instances form a strong representation system for the class of mappings specified by st-tgds (see Theorems 4.4 and 5.1), and the fact that for every mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, where σ_{12} is an SO tgd, there exists a finite sequence of mappings $\mathcal{M}_1, \ldots, \mathcal{M}_k$, each specified by a set of st-tgds, such that $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$ [14]. Indeed, in order to obtain a universal $\mathcal{R}_{\text{pos}}$-solution for a positive conditional instance $\mathcal{I}$ under $\mathcal{M}$, one can use the techniques described in Section 5 to construct a sequence $\mathcal{I}_1, \ldots, \mathcal{I}_k$ of positive conditional instances such that: (1) $\mathcal{I}_1$ is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}_1$ and (2) $\mathcal{I}_i$ is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}_{i-1}$ under $\mathcal{M}_i$, for every $i \in \{2, \ldots, k\}$. In this case one concludes that $\mathcal{I}_k$ is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$ since $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$. Notice that this approach cannot be used to prove similar results within the data exchange setting proposed by Fagin et al. [13], as naive instances do not form a strong representation system for the class of mappings specified by st-tgds.

6.2 Positive conditional instances as first class citizens

In the next sections, we study the composition and inversion of schema mappings in the presence of positive conditional instances. But for doing this, we have to show first how positive conditional instances are included as first class citizens in schema mappings.

We have defined mappings as sets of pairs of instances with complete information. Here we do not deviate from this definition and, thus, we introduce a new terminology to refer to mappings that also contain positive conditional instances. In general, a *positive conditional* mapping, or just PC-mapping, from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ is a set of pairs $(\mathcal{I}_1, \mathcal{I}_2)$, where $\mathcal{I}_1$ is a positive conditional instance of $\mathbf{S}_1$ and $\mathcal{I}_2$ is a positive conditional instance of $\mathbf{S}_2$. In this section, we will be mostly dealing with PC-mappings that are generated from a usual mapping by using the notion of solution for positive conditional instances. More precisely, given a (usual) mapping $\mathcal{M}$ from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, define the PC-mapping *generated* from $\mathcal{M}$, denoted by $\text{PC}(\mathcal{M})$, as:

$$\{(\mathcal{I}_1, \mathcal{I}_2) \mid \mathcal{I}_1,\ \mathcal{I}_2 \text{ are positive conditional instances of } \mathbf{S}_1 \text{ and } \mathbf{S}_2,\ \text{respectively, and } \mathcal{I}_2 \text{ is an } \mathcal{R}_{\text{pos}}\text{-solution for } \mathcal{I}_1 \text{ under } \mathcal{M}\}.$$

That is, $\text{PC}(\mathcal{M})$ is obtained from $\mathcal{M}$ by including the pairs $(\mathcal{I}_1, \mathcal{I}_2)$ of positive conditional instances such that $\mathcal{I}_2$ is a solution for $\mathcal{I}_1$ under $\mathcal{M}$, according to the notion of solution for instances with incomplete information introduced in this paper.

Given a mapping $\mathcal{M}$, $\text{PC}(\mathcal{M})$ only includes positive conditional instances in the source and target schemas. We have decided to exclude the usual instances with complete information from $\text{PC}(\mathcal{M})$, as if $\mathcal{M}$ is specified by a set of st-tgds (and, more generally, by an SO tgd), then the relationship between the usual instances according to $\mathcal{M}$ is captured by $\text{PC}(\mathcal{M})$. More precisely, an instance I of a schema $\mathbf{S}$ can be considered as a positive conditional instance without null values and with the element-condition $\top$ associated to every fact. Then it is possible to prove the following.

Proposition 6.3 *Let $\mathcal{M}$ be a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ that is closed-down on the left and closed-up on the right. Then for every pair of instances I_1, I_2 of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, it holds that $(I_1, I_2) \in \mathcal{M}$ iff $(I_1, I_2) \in \text{PC}(\mathcal{M})$.*

In this proposition, a mapping $\mathcal{M}$ is said to be closed-down on the left if for every $(I, J) \in \mathcal{M}$ and instance I' such that $I' \subseteq I$, it holds that $(I', J) \in \mathcal{M}$, and it is said to be closed-up on the right if for every $(I, J) \in \mathcal{M}$ and instance J' such that $J \subseteq J'$, it holds that $(I, J') \in \mathcal{M}$. For example, every mapping specified by a set of st-tgds satisfies these conditions, as well as every mapping specified by an SO tgd.

6.3 Composition in the presence of positive conditional instances

In [14], SO tgds were introduced to deal with the problem of composing schema mappings. In fact, it was proved in [14] that (1) the composition of a finite number of mappings specified by st-tgds can always be specified by an SO tgd, (2) that SO tgds are closed under composition, and (3) that every SO tgd specifies the composition of a finite number of mappings specified by st-tgds. Thus, SO tgds are a natural candidate to study the composition of schema mappings including positive conditional instances. We confirm this intuition by showing that SO tgds satisfy the conditions (1), (2) and (3) for the case of PC-mappings. Notice that for mappings $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, $\text{PC}(\mathcal{M}_{12})$ and $\text{PC}(\mathcal{M}_{23})$ are binary relations and, thus, the composition $\text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$ is defined as the usual composition of binary relations. More precisely, $\text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$ is the set of all pairs of positive conditional instances $(\mathcal{I}_1, \mathcal{I}_3)$ for which there exists a positive conditional instance $\mathcal{I}_2$ such that $(\mathcal{I}_1, \mathcal{I}_2) \in \text{PC}(\mathcal{M}_{12})$ and $(\mathcal{I}_2, \mathcal{I}_3) \in \text{PC}(\mathcal{M}_{23})$. In this study, the following lemma is the key ingredient.

Lemma 6.4 *Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \sigma_{23})$, where σ_{12} and σ_{23} are SO tgds. Then $\text{PC}(\mathcal{M}_{12} \circ \mathcal{M}_{23}) = \text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$.*

[*] We consider a single SO tgd in this definition as this class of dependencies is closed under conjunction (thus, a finite set of SO tgds is equivalent to a single SO tgd).

From the results in [14] and Lemma 6.4, it is straightforward to prove the desired results.

Corollary 6.5

(1) For every $i \in \{1, \ldots, k-1\}$, let $\mathcal{M}_{i\,i+1} = (\mathbf{S}_i, \mathbf{S}_{i+1}, \Sigma_{i\,i+1})$ with $\Sigma_{i\,i+1}$ a set of st-tgds. Then there exists a mapping $\mathcal{M}_{1k} = (\mathbf{S}_1, \mathbf{S}_k, \sigma_{1k})$, where σ_{1k} is an SO tgd, such that $\text{PC}(\mathcal{M}_{12}) \circ \cdots \circ \text{PC}(\mathcal{M}_{k-1\,k}) = \text{PC}(\mathcal{M}_{1k})$.

(2) Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \sigma_{23})$, where σ_{12} and σ_{23} are SO tgds. Then there exists a mapping $\mathcal{M}_{13} = (\mathbf{S}_1, \mathbf{S}_3, \sigma_{13})$, where σ_{13} is an SO tgd, such that $\text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23}) = \text{PC}(\mathcal{M}_{13})$.

(3) Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, where σ_{12} is an SO tgd. Then there exists a sequence $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_k$ of mappings, each specified by a set of st-tgds, such that $\text{PC}(\mathcal{M}) = \text{PC}(\mathcal{M}_1) \circ \text{PC}(\mathcal{M}_2) \circ \cdots \circ \text{PC}(\mathcal{M}_k)$.

We have shown that SO tgds are the right language to deal with the composition of schema mappings including positive conditional instances. Interestingly, we show in the following section that the inclusion of this type of instances also allow mappings specified with SO tgds to become *invertible*.

6.4 Inversion in the presence of positive conditional instances

We consider in this section the notion of mapping inversion introduced in [6]. In that paper, the authors give a formal definition for what it means for a mapping $\mathcal{M}'$ to recover *sound information* with respect to a mapping $\mathcal{M}$. Such a mapping $\mathcal{M}'$ is called a recovery of $\mathcal{M}$ in [6]. Given that, in general, there may exist many possible recoveries for a given mapping, an order relation on recoveries is introduced in [6] that naturally gives rise to the notion of maximum recovery, which is a mapping that brings back the maximum amount of sound information.

Let $\mathbf{S}_1$, $\mathbf{S}_2$ be relational schemas, $\mathcal{P}_{12}$ a PC-mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ and $\mathcal{P}_{21}$ a PC-mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{P}_{21}$ is said to be a *recovery* of $\mathcal{P}_{12}$ if for every positive conditional instance $\mathcal{I}_1$ of $\mathbf{S}_1$, it holds that $(\mathcal{I}_1, \mathcal{I}_1) \in \mathcal{P}_{12} \circ \mathcal{P}_{21}$. Moreover, $\mathcal{P}_{21}$ is said to be a *maximum recovery* of $\mathcal{P}_{12}$ if $\mathcal{P}_{21}$ is a recovery of $\mathcal{P}_{12}$ and for every PC-mapping $\mathcal{P}'_{21}$ that is a recovery of $\mathcal{P}_{12}$, it holds that $\mathcal{P}_{12} \circ \mathcal{P}_{21} \subseteq \mathcal{P}_{12} \circ \mathcal{P}'_{21}$. That is, the smaller the space of solutions generated by $\mathcal{P}_{12} \circ \mathcal{P}_{21}$, the more informative $\mathcal{P}_{21}$ is about the initial source instances.

It was shown in [5] that there exist mappings specified by SO tgds that admit neither a Fagin-inverse [12] nor a quasi-inverse [15] nor a maximum recovery [6]. The same has been shown for the notion of CQ-maximum recovery studied in [5], and it is not clear whether the notion of inversion introduced in [16] is appropriate for the language of SO tgds. Thus, up to this point, no inverse notion has shown to be appropriate for the fundamental language of SO tgds. As our most important result regarding metadata management, we show that the situation is completely different if positive conditional instances are allowed in source and target schemas.

Theorem 6.6 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, where σ_{12} is an SO tgd. Then $\text{PC}(\mathcal{M})$ admits a maximum recovery.*

7. KNOWLEDGE BASES

In this section, we apply our framework for representation systems to *knowledge bases*. In particular, we introduce the novel notion of *exchanging implicit knowledge* via a schema mapping. A knowledge base is composed of *explicit data*, in our context given by a database instance, and *implicit data* usually given by a set of *rules* specified in some logical formalism. Examples of knowledge bases are Datalog programs (where the explicit data is called *extensional database* and the implicit data *intentional database*), and Description Logics specifications (where the explicit data is called *ABox* and the implicit data *TBox*). Let us motivate this section with a simple example.

Example 7.1. Consider a schema $\mathbf{S}_1$ consisting of relations $F(\cdot,\cdot)$, $M(\cdot,\cdot)$, $P(\cdot,\cdot)$ and *GP*$(\cdot,\cdot)$, which are used to store genealogical data (F stands for *father*, M for *mother*, P for *parent*, and *GP* for *grandparent*). Consider the following set Σ_1 of rules that states some natural implicit knowledge over $\mathbf{S}_1$:

$$\begin{aligned} F(x,y) &\rightarrow P(x,y) \\ M(x,y) &\rightarrow P(x,y) \\ P(x,y) \wedge P(y,z) &\rightarrow \textit{GP}(x,z) \end{aligned}$$

Thus, if $I = \{F(a,b), M(c,b), F(b,d)\}$, then from I and Σ_1 one can *infer* that a and c are *parents* of b, and that a and c are *grandparents* of d. That is, one can infer the atoms $P(a,b)$, $P(c,b)$, *GP*(a,d), and *GP*(c,d). Now assume that one needs to exchange data from $\mathbf{S}_1$ to a new schema $\mathbf{S}_2 = \{F'(\cdot,\cdot), \textit{GP}'(\cdot,\cdot)\}$ by using the following set Σ_{12} of st-tgds:

$$\begin{aligned} F(x,y) &\rightarrow F'(x,y), \\ \textit{GP}(x,y) &\rightarrow \textit{GP}'(x,y) \end{aligned}$$

In this case, one would like to create a knowledge base over $\mathbf{S}_2$ that represents both the explicit data in I and the implicit data given by Σ_1. Thus, one could try first to represent Σ_1 over $\mathbf{S}_2$ according to the relationship established by Σ_{12}. Given that one is copying F and *GP* through Σ_{12}, the following rule over $\mathbf{S}_2$ is a natural way of representing the implicit knowledge in Σ_1 that is transferred to $\mathbf{S}_2$ through Σ_{12}:

$$F'(x,y) \wedge F'(y,z) \rightarrow \textit{GP}'(x,z),$$

This dependency states that if in schema $\mathbf{S}_2$ we have that x is the father of y and that y is the father of of z, then x should be a grand parent of z. Let Σ_2 be consisting of the above rule. If one considers Σ_2 as the implicit knowledge over $\mathbf{S}_2$, then one can materialize the instance $J = \{F'(a,b), F'(b,d), \textit{GP}'(c,d)\}$ to obtain a natural knowledge base over $\mathbf{S}_2$ that represents the initial knowledge base given by I and Σ_1. Notice that the fact *GP*$'(c,d)$ needs to be explicitly included in J, since it comes from an atom that is inferred from predicate M in $\mathbf{S}_1$, and one does not have any information about M in $\mathbf{S}_2$. On the other hand, one does not need to include in J the fact *GP*$'(a,d)$, as it can be inferred from J and Σ_2. □

This example shows that for the case of knowledge bases, one might be interested in exchanging not only explicit data but also the implicit information in the source knowledge base. As we will see, in general one would have many possibilities when deciding what to make explicit and what to keep implicit when exchanging knowledge bases.

Next we formalize the notion of knowledge base used in this paper, and introduce the notion of knowledge-base solution for a mapping. A *knowledge base* over a schema $\mathbf{S}$ is a pair (I, Σ), where $I \in \text{INST}(\mathbf{S})$ and Σ is a set of logical sentences over $\mathbf{S}$. Given a knowledge base (I, Σ), we denote by $\text{MOD}(I, \Sigma)$ the set of possible *models* of this base, which are all the instances that contain the explicit data in I and satisfy Σ:

$$\text{MOD}(I, \Sigma) = \{K \in \text{INST}(\mathbf{S}) \mid I \subseteq K \text{ and } K \models \Sigma\}.$$

Let $\mathbf{K}$ be the class of all knowledge bases (over all possible relational schemas). Then the pair $\mathcal{K} = (\mathbf{K}, \text{MOD})$ is a representation system, and thus, we can apply Definition 3.1 to obtain a

notion of solution for knowledge bases. More precisely, given a mapping $\mathcal{M}$ from $\mathbf{S}_1$ to $\mathbf{S}_2$ and knowledge bases (I, Σ_1), (J, Σ_2) over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, we have that (J, Σ_2) is a $\mathcal{K}$-solution for (I, Σ_1) under $\mathcal{M}$ if for every $L \in \text{MOD}(J, \Sigma_2)$ there exists an instance $K \in \text{MOD}(I, \Sigma_1)$ such that $(K, L) \in \mathcal{M}$, or equivalently $\text{MOD}(J, \Sigma_2) \subseteq \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$. In this case, we also call call (J, Σ_2) a *knowledge-base solution* of (I, Σ_1) under $\mathcal{M}$.

Example 7.2. Let (I, Σ_1), (J, Σ_2) and $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ be defined as in Example 7.1. Then it can be shown that (J, Σ_2) is a knowledge-base solution of (I, Σ_1) under the mapping $\mathcal{M}$. □

Many algorithmic problems arise in the context of knowledge bases and schema mappings. In Section 7.1, we study the fundamental problem of checking, given a schema mapping $\mathcal{M}$ and knowledge bases K_1 and K_2, whether K_2 is a knowledge-base solution for K_1 under $\mathcal{M}$. In Section 8, we study the novel notion of *exchanging knowledge*, that is, the problem of materializing (good) target knowledge bases. But before considering these problems, we introduce some notation that will be extensively used in the rest of the paper. We use the standard notion of *chase* of an instance with a set of full tgds (see [23] for a formalization of the chase). Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be disjoint schemas, and let I be an instance of $\mathbf{S}_1$. For a set of full tgds Σ_1 over a schema $\mathbf{S}_1$, we denote by $\text{chase}_{\Sigma_1}(I)$ the result of chasing I with Σ_1. Moreover, let Σ_{12} be a set of full st-tgds from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $J_\emptyset$ the empty instance of $\mathbf{S}_2$. Notice that the result of chasing $(I, J_\emptyset)$ with Σ_{12} is an instance $(I, J^\star)$ of $\mathbf{S}_1 \cup \mathbf{S}_2$. We denote by $\text{chase}_{\Sigma_{12}}(I)$ the resulting instance $J^\star$ (which is the standard notation in the data exchange context [13]). Thus, $\text{chase}_{\Sigma_1}(I)$ is an instance of $\mathbf{S}_1$, while $\text{chase}_{\Sigma_{12}}(I)$ is an instance of $\mathbf{S}_2$.

7.1 Complexity of recognizing solutions

Given a mapping $\mathcal{M}$ from $\mathbf{S}_1$ to $\mathbf{S}_2$, and a representation system $\mathcal{R} = (\mathbf{W}, \text{rep})$, the problem CHECKSOLUTION$(\mathcal{M}, \mathcal{R})$ was defined in Section 5.3 as the problem of verifying, given $\mathcal{U} \in \mathbf{W}$ of type $\mathbf{S}_1$ and $\mathcal{V} \in \mathbf{W}$ of type $\mathbf{S}_2$, whether $\mathcal{V}$ is an $\mathcal{R}$-solution of $\mathcal{U}$ under $\mathcal{M}$. In this section, we study the complexity of this problem for the class of mappings specified by st-tgds and for the representation system $\mathcal{K}$ of knowledge bases.

Two representation systems that are of particular interest in our study are the systems of *tgds knowledge bases* and *full-tgds knowledge bases*, denoted by $\mathcal{K}_{\text{tgd}} = (\mathbf{K}_{\text{tgd}}, \text{MOD})$, and $\mathcal{K}_{\text{full-tgd}} = (\mathbf{K}_{\text{full-tgd}}, \text{MOD})$, respectively. More specifically, $\mathcal{K}_{\text{tgd}}$ is the system obtained by restricting $\mathcal{K}$ to the class of all knowledge bases (I, Σ) with Σ a set of tgds, and $\mathcal{K}_{\text{full-tgd}}$ the representation system obtained by restricting $\mathcal{K}$ to the class of knowledge bases (I, Σ) with Σ a set of full tgds.

Our first theorem is an undecidability result for knowledge bases that are specified by general tgds. The undecidability result holds even for a fixed schema mapping $\mathcal{M}$ specified by full st-tgds.

Theorem 7.3 *There exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, with Σ_{12} a set of full st-tgds, for which* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{tgd}})$ *is undecidable.*

Theorem 7.3 tells us that to obtain decidability results, we have to focus on some fragments of $\mathcal{K}_{\text{tgd}}$. In what follows, we study the complexity of the problem for the class of knowledge bases given by full tgds. We start by stating the complexity of CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ when the source implicit knowledge or the target implicit knowledge is assumed to be fixed. In the former case, we assume that we are given a fixed set Σ_1 of full tgds over the source schema and the problem is to check, given a source instance I and a target knowledge base (J, Σ_2), whether (J, Σ_2) is a knowledge-base solution for (I, Σ_1) under $\mathcal{M}$. The latter case is defined analogously.

Theorem 7.4 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then the problem* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ *is: (1) in PTIME if both source implicit knowledge and target implicit knowledge are fixed, (2) NP-complete if source implicit knowledge is fixed, (3) coNP-complete if target implicit knowledge is fixed.*

In the general case, where the implicit knowledge is not assumed to be fixed, we obtain that the problem is complete for $\Delta_2^P[O(\log n)]$, which is the class of all problems that can be decided in polynomial time by a deterministic Turing machine that is allowed to make a logarithmic number of calls to an NP oracle [25].

Theorem 7.5 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ *is $\Delta_2^P[O(\log n)]$-complete. The problem is $\Delta_2^P[O(\log n)]$-hard even if $\mathcal{M}$ is specified by a set of full st-tgds.*

There are only a few natural problems that are complete for $\Delta_2^P[O(\log n)]$. It is interesting that a complete problem for this class arises in the simple framework of data exchange. Some other problems in the context of databases and logic programming that are complete for this class can be found in [11, 9].

A natural question at this point is whether one can obtain decidability for a representation system that is in between $\mathcal{K}_{\text{full-tgd}}$ and $\mathcal{K}_{\text{tgd}}$. An obvious candidate would be the class of knowledge bases defined by *weakly acyclic sets of tgds* [10, 13]. We leave for future research the study of the complexity in this case.

8. KNOWLEDGE EXCHANGE

The most important problem in data exchange is the problem of materializing a target solution for a given source instance. In the previous section, we have extended the notion of solution for knowledge bases and, thus, it is natural to consider the problem of *knowledge exchange*, that is, the problem of materializing a target knowledge base that correctly represents a source knowledge base according to a given mapping. To this end, the first question to answer is what is a *good* knowledge base to materialize. In Section 8.1, we consider the notion of *universal $\mathcal{K}$-solution* that is obtained by applying Definition 3.2 to the representation system $\mathcal{K}$ of knowledge bases. In Section 8.2, we show that there are other natural $\mathcal{K}$-solutions that extend universal $\mathcal{K}$-solutions and that can also be considered good alternatives to materialize. We present algorithms for computing such solutions in Section 8.3.

Given the undecidability results about knowledge bases specified by (non-full) tgds, proved in Section 7.1, we focus our investigation on full tgds. It is important to notice that this case includes some of the motivating scenarios for our investigation, such as RDFS graphs [19].

8.1 Universal $\mathcal{K}$-solutions

Let $\mathcal{K} = (\mathbf{K}, \text{MOD})$ be the representation system of knowledge bases. We can directly apply the notion of universal $\mathcal{K}$-solution to define a class of good solutions. More precisely, we obtain from Definition 3.2 that (J, Σ_2) is a universal $\mathcal{K}$-solution of (I, Σ_1) under a mapping $\mathcal{M}$ if

$$\text{MOD}(J, \Sigma_2) \quad = \quad \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1)). \tag{2}$$

It is easy to show that for $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and for every set Σ_1 of full tgds over $\mathbf{S}_1$, the knowledge base $(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)), \Sigma_2)$ with $\Sigma_2 = \emptyset$, is always a universal $\mathcal{K}$-solution of (I, Σ_1). Notice that this induces a straight-

forward procedure to compute a good solution: we just chase I with Σ_1 and then with Σ_{12}. Thus we obtain the following result.

Proposition 8.1 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, with Σ_{12} a set of full st-tgds. There exists an exponential-time algorithm that, given a knowledge base (I, Σ_1) over $\mathbf{S}_1$, with Σ_1 a set of full tgds, produces a polynomial-size universal $\mathcal{K}$-solution of (I, Σ_1) under $\mathcal{M}$.*

Moreover, it immediately follows from equation (2) that universal $\mathcal{K}$-solutions can be used to compute the certain answers of an arbitrary query Q over (I, Σ_1) under a mapping $\mathcal{M}$.

8.2 Minimal knowledge-base solutions

The universal $\mathcal{K}$-solutions generated in the previous section use the empty set as the implicit knowledge in the target. We argue in this section that there could be other natural $\mathcal{K}$-solutions that may not be universal but still desirable to materialize, mostly because they make good use of the implicit knowledge in the target schema.

Example 8.2. In Example 7.1, we give a $\mathcal{K}$-solution (J, Σ_2) that can be considered as a good solution. However, we have that $\text{MOD}(J, \Sigma_2) \subsetneq \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$ and, thus, (J, Σ_2) is not a universal $\mathcal{K}$-solution for (I, Σ_1). The reason is that mapping $\mathcal{M}$ is *closed-up on the right* and, hence, if $K \in \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$ and $K \subseteq K'$, then $K' \in \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$. On the other hand, $\text{MOD}(J, \Sigma_2)$ does not satisfy this property. To see why this is the case, consider the instance $K = J \cup \{GP'(a, d)\}$. It is easy to see that $K \in \text{MOD}(J, \Sigma_2)$. But if we now consider the instance $K' = K \cup \{F'(b, e)\}$, then we have that $K \subseteq K'$ but $K' \notin \text{MOD}(J, \Sigma_2)$ since K' does not satisfy rule $F'(x, y) \wedge F'(y, z) \rightarrow GP'(x, z)$ (given that $F'(a, b), F'(b, e) \in K'$ but $GP'(a, e) \notin K'$). □

In what follows, we introduce a new class of good $\mathcal{K}$-solutions that captures the intuition in Example 7.1. But before we need to introduce some terminology. Let $\mathcal{X}$ be a set of instances over a schema $\mathbf{S}$. We say that $\mathcal{X}$ is *closed-up* if whenever $K \in \mathcal{X}$ and K' is an instance of $\mathbf{S}$ such that $K \subseteq K'$, we have that $K' \in \mathcal{X}$. Moreover, we define the set of *minimal instances* of $\mathcal{X}$ as:

$$\text{Min}(\mathcal{X}) = \{K \in \mathcal{X} \mid \text{there is no } K' \in \mathcal{X} \text{ such that } K' \subsetneq K\}.$$

A closed-up set of instances is characterized by its set of minimal instances, as if $\mathcal{X}$ and $\mathcal{Y}$ are closed-up, then $\mathcal{X} = \mathcal{Y}$ if and only if $\text{Min}(\mathcal{X}) = \text{Min}(\mathcal{Y})$.

For every mapping $\mathcal{M}$ specified by a set of st-tgds, and more generally for every mapping that is closed-up on the right, and for every knowledge base (I, Σ_1), it holds that $\text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$ is a closed-up set. Thus, since $\text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$ is essentially characterized by its minimal instances, we can naturally relax equation (2) by not requiring that $\text{MOD}(J, \Sigma_2)$ is equal to $\text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1))$, but instead that both sets coincide in their sets of minimal instances. Notice that by doing this we retain the same query answering properties as universal $\mathcal{K}$-solutions when considering monotone queries. All the above discussion suggests the following definition of *minimal knowledge-base solution*. In the definition, we use $\mathcal{X} \equiv_{\text{Min}} \mathcal{Y}$ to denote that $\text{Min}(\mathcal{X}) = \text{Min}(\mathcal{Y})$.

Definition 8.3 *Let $\mathcal{M}$ be a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, and (I, Σ_1), (J, Σ_2) knowledge bases over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. Then (J, Σ_2) is a* minimal knowledge-base solution *for (I, Σ_1) under $\mathcal{M}$ if:*

$$\text{MOD}(J, \Sigma_2) \quad \equiv_{\text{Min}} \quad \text{SOL}_{\mathcal{M}}(\text{MOD}(I, \Sigma_1)).$$

The following result is a simple yet useful characterization of minimal knowledge-base solutions for the case of full tgds. It also gives evidence of the naturalness of our definition of good solution.

Proposition 8.4 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, and (I, Σ_1), (J, Σ_2) be knowledge bases over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. If Σ_{12}, Σ_1 and Σ_2 are sets of full tgds, then the following are equivalent:*

(1) (J, Σ_2) is a minimal knowledge-base solution of (I, Σ_1).

(2) $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) = \text{chase}_{\Sigma_2}(J)$.

Notice that every universal $\mathcal{K}$-solution is a minimal knowledge-base solution, but, as the following example shows, the opposite does not hold in general.

Example 8.5. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, (I, Σ_1), and (J, Σ_2) be as in Example 7.1. We have that $\text{chase}_{\Sigma_1}(I)$ is the instance:

$$I' = \{ \quad F(a, b), M(c, b), F(b, d), P(a, b), \\ P(c, b), P(b, d), GP(a, d), GP(c, d) \quad \}.$$

If we compute $\text{chase}_{\Sigma_{12}}(I')$, we obtain the instance $\{F'(a, b), F'(b, d), GP'(a, d), GP'(c, d)\}$. If we now compute $\text{chase}_{\Sigma_2}(J)$, we obtain the instance $\{F'(a, b), F'(b, d), GP'(a, d), GP'(c, d)\}$. Thus, given that $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) = \text{chase}_{\Sigma_2}(J)$, we conclude from Proposition 8.4 that (J, Σ_2) is a minimal knowledge-base solution for (I, Σ_1) . □

8.3 Computing minimal knowledge-base solutions

As we pointed out in the previous section, when doing knowledge exchange, it is desirable to materialize target knowledge bases with *as much implicit knowledge as possible*. Yet there is another requirement that one would like to impose to this process. Consider a mapping $\mathcal{M}$ and a source knowledge base (I, Σ_1). In the computation of a solution (J, Σ_2) for (I, Σ_1), it would be desirable that the resulting set Σ_2 depends only on Σ_1 and $\mathcal{M}$, that is, one would like that the implicit knowledge in the target depends only on the mapping and the implicit knowledge in the source. This motivates the following definition of a *safe* set of dependencies.

Definition 8.6 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and Σ_1 be a set of full tgds over $\mathbf{S}_1$. Then a set Σ_2 of dependencies over $\mathbf{S}_2$ is* safe *for Σ_1 and $\mathcal{M}$ if for every instance I of $\mathbf{S}_1$, there exists an instance J of $\mathbf{S}_2$ such that (J, Σ_2) is a minimal knowledge-base solution of (I, Σ_1) under $\mathcal{M}$.*

There are many safe sets. In particular, $\Sigma_2 = \emptyset$ is safe for every Σ_1 and $\mathcal{M}$, but it is obviously useless as implicit knowledge. In general, one would like to materialize a safe set Σ_2 that is as informative as possible. In this section, we show how to compute such safe sets and how to use them to materialize knowledge-base solutions. More specifically, we show in Section 8.3.1 that there exists an algorithm that computes *optimal* safe sets; with input Σ_1 and $\mathcal{M}$, the algorithm computes a set Σ_2 such that Σ_2 is safe for Σ_1 and $\mathcal{M}$, and for every other safe set Σ_2' for Σ_1 and $\mathcal{M}$, it holds that Σ_2 logically implies Σ_2'. The output of the algorithm is a set of second-order logic sentences, which motivate us to consider in Section 8.3.1 the problem of generating nontrivial safe sets that, although not optimal, can be expressed in a much simpler language. Finally, we propose in Section 8.3.2 a strategy that uses safe sets to compute minimal knowledge-base solutions.

8.3.1 Computing safe implicit knowledge

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, Σ_1 a set of full tgds over $\mathbf{S}_1$ and Σ_2 an arbitrary set of dependencies over $\mathbf{S}_2$. From now on, we say that Σ_2 is optimal-safe for Σ_1 and $\mathcal{M}$ if: (1) Σ_2 is safe for Σ_1 and $\mathcal{M}$, and (2) for every Σ_2' that is safe for Σ_1 and $\mathcal{M}$, it holds that Σ_2 implies Σ_2'. In our first result, we show that there exists an algorithm for computing optimal-safe sets.

Theorem 8.7 *There exists a polynomial-time algorithm* OPTIMALSAFE *that, given $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and a set Σ_1 of full tgds over $\mathbf{S}_1$, computes a set Σ_2 of second-order logic sentences that is optimal-safe for Σ_1 and $\mathcal{M}$.*

A natural question at this point is whether one could modify OPTIMALSAFE to return a set of FO-sentences. Unfortunately, the following theorem gives a negative answer to this question.

Theorem 8.8 *There exist $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and a set Σ_1 of full tgds over $\mathbf{S}_1$ such that there is no set Σ_2 of FO-sentences that is optimal-safe for Σ_1 and $\mathcal{M}$.*

Theorem 8.8 shows that FO is not enough, in general, to specify an optimal-safe set of dependencies. Nevertheless, in practice one might be more interested in generating nontrivial safe sets that, although not optimal, can be expressed in a simple language. The ideal would be to have nontrivial safe sets specified by full tgds or a mild extension of full tgds. In what follows, we present an algorithm that, given a mapping $\mathcal{M}$ specified by a set of full st-tgds and an *acyclic* set Σ_1 of full tgds over the source schema, generates a set Σ_2 that is safe for Σ_1 and $\mathcal{M}$, and which is specified by a set of full tgds with inequalities in their premises.

A set Σ of full tgds is acyclic if there exists a function that assigns a natural number to each predicate symbol in Σ in such a way that for every $\sigma \in \Sigma$, if P is a relation symbol in the premise of σ and R is the relation symbol in the conclusion of σ, then $f(P) < f(R)$. A well-know property of an acyclic set Σ of full tgds is that it has a *finite unfolding*; for every relational atom $R(\bar{x})$ in the conclusion of a dependency of Σ, there exists a formula $\alpha(\bar{x})$ in $\text{UCQ}^{=}$ such that for every instance I, it holds that $R(\bar{a})$ is in $\text{chase}_{\Sigma}(I)$ if and only if $\alpha(\bar{a})$ holds in I. The *unfolding* of Σ, that we denote by Σ^{+}, is constructed by first computing $\alpha(\bar{x})$ for every $R(\bar{x})$ in the conclusion of a tgd in Σ, then adding $\beta(\bar{x}) \rightarrow R(\bar{x})$ to Σ^{+} for every $\beta(\bar{x})$ in $\text{CQ}^{=}$ that is a disjunct in $\alpha(\bar{x})$, and then eliminating equalities by using variable substitutions.

To present our algorithm, we need to introduce some terminology. Given a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and a query Q over $\mathbf{S}_1$, we say that Q is *target rewritable* under $\mathcal{M}$ if there exists a query Q' over $\mathbf{S}_2$ such that for every instance I of $\mathbf{S}_1$, it holds that $Q(I) = \text{certain}_{\mathcal{M}}(Q', I)$. It is implicit in [4] that if Σ_{12} is a set of full tgds and Q is a conjunctive query, then it is decidable in coNEXPTIME whether Q is target rewritable (see Theorems 4.1 and 4.3 in [4]). Moreover, from the results in [4], we know that there exists a procedure $\text{TREW}(\mathcal{M}, Q)$ that computes a query in $\text{UCQ}^{=,\neq}$ that is a target rewriting of Q under $\mathcal{M}$ (if such a rewriting exists). Besides, we also need a procedure to compose full st-tgds. In [14], the authors show that there exists a procedure COMPOSEFULL that given sets Σ_{12} and Σ_{23} of full st-tgds from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ and from $\mathbf{S}_2$ to a schema $\mathbf{S}_3$, respectively, computes a set Σ_{13} of full st-tgds from $\mathbf{S}_1$ to $\mathbf{S}_3$ such that $(I, J) \models \Sigma_{13}$ if and only if there exists K such that $(I, K) \models \Sigma_{12}$ and $(K, J) \models \Sigma_{23}$. It can be easily shown that if Σ_{12} is a set of full st-tgds with inequalities in the premises, then COMPOSEFULL returns a set of full st-tgds with inequalities in the premises that defines the composition of Σ_{12} and Σ_{23}. With procedures TREW and COMPOSEFULL, we have all the necessary ingredients for our algorithm.

Algorithm: $\text{FULLSAFE}(\mathcal{M}, \Sigma_1)$

Input:	$\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and an acyclic set Σ_1 of full tgds over $\mathbf{S}_1$.
Output:	A set Σ_2 of full tgds with inequalities over $\mathbf{S}_2$ that is safe for Σ_1 and $\mathcal{M}$.

1. Construct a set of formulas Σ_1^{+} by unfolding Σ_1.
2. Construct a set Σ' of full st-tgds with inequalities from $\mathbf{S}_2$ to $\mathbf{S}_1$ as follows. Begin with $\Sigma' = \emptyset$. For every tgd $\alpha(\bar{x}) \rightarrow R(\bar{x})$ in Σ_1^{+} do the following:
   2.1. If $\alpha(\bar{x})$ is target rewritable under $\mathcal{M}$, then let $\beta(\bar{x})$ be the query in $\text{UCQ}^{=,\neq}$ over $\mathbf{S}_2$ that is the output of $\text{TREW}(\mathcal{M}, \alpha(\bar{x}))$. For every disjunct $\gamma(\bar{x})$ in $\beta(\bar{x})$ add to Σ' the dependency $\gamma(\bar{x}) \rightarrow R(\bar{x})$ (and eliminate equalities by using variable substitutions).
3. Let $\hat{\mathbf{S}}_2$ be a copy of $\mathbf{S}_2$, and Σ'_{12} the set of full st-tgds from $\mathbf{S}_1$ to $\hat{\mathbf{S}}_2$ obtained from Σ_{12} by replacing every $R \in \mathbf{S}_2$ by $\hat{R}$.
4. Let Σ'' be the set of full st-tgds with inequalities from $\mathbf{S}_2$ to $\hat{\mathbf{S}}_2$ that is obtained as the output of $\text{COMPOSEFULL}(\Sigma', \Sigma'_{12})$.
5. Let Σ_2 be the set of formulas over $\mathbf{S}_2$ obtained from Σ'' by replacing every $\hat{R} \in \hat{\mathbf{S}}_2$ by R. Return Σ_2. □

Theorem 8.9 $\text{FULLSAFE}(\mathcal{M}, \Sigma_1)$ *computes a set Σ_2 of full tgds with inequalities in the premises which is safe for Σ_1 and $\mathcal{M}$.*

Example 8.10. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and Σ_1 be as defined in Example 7.1. It is not difficult to see that dependency σ given by

$$\exists y(F(x, y) \wedge F(y, z)) \quad \rightarrow \quad GP(x, z)$$

is in Σ_1^{+}. Now the query given by $\exists y\,(F(x, y) \wedge F(y, z))$ is target rewritable under $\mathcal{M}$, and its rewriting is $\exists y\,(F'(x, y) \wedge F'(y, z))$. Thus, in Step 2 of FULLSAFE, we add dependency:

$$\exists y(F'(x, y) \wedge F'(y, z)) \quad \rightarrow \quad GP(x, z)$$

to Σ'. In the set Σ'_{12} created in Step 3, we have the dependency:

$$GP(x, z) \quad \rightarrow \quad \widehat{GP'}(x, z).$$

Thus, the output of $\text{COMPOSEFULL}(\Sigma', \Sigma'_{12})$ contains the dependency $\exists y(F'(x, y) \wedge F'(y, z)) \rightarrow \widehat{GP'}(x, z)$, which implies that:

$$\exists y(F'(x, y) \wedge F'(y, z)) \quad \rightarrow \quad GP'(x, z) \tag{3}$$

is in the output of $\text{FULLSAFE}(\mathcal{M}, \Sigma_1)$. In fact, it can be proved that the set Σ_2 returned by $\text{FULLSAFE}(\mathcal{M}, \Sigma_1)$ is logically equivalent to the set consisting of dependency (3). □

8.3.2 *Using safe implicit knowledge to compute minimal knowledge-base solutions*

For a mapping $\mathcal{M}$ and a source knowledge base (I, Σ_1), a minimal knowledge-base solution of (I, Σ_1) consists of an instance J and a set Σ_2 of dependencies. Up to this point, we have described two alternative algorithms that compute the set Σ_2 from Σ_1 and $\mathcal{M}$. In this section, we propose a strategy to compute instance J.

Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and (I, Σ_1) a knowledge base over $\mathbf{S}_1$, where Σ_1 is a set of full tgds. As we pointed out before, $J = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ can always be used as the explicit data in a minimal knowledge-base solution of (I, Σ_1). However, such an instance does not need to make use of any implicit knowledge and, thus, it does not take advantage of any of the algorithms proposed in the previous section for computing safe sets. In fact, given these algorithms, one would expect that some parts of the instance $\text{chase}_{\Sigma_1}(I)$ are not necessary given the target implicit knowledge. In what follows, we propose an approach that given (I, Σ_1), $\mathcal{M}$ and a safe set Σ_2 for Σ_1 and $\mathcal{M}$, computes an instance J that makes use of the implicit knowledge in Σ_2. More precisely, the approach first constructs a minimal set Σ'_1 of full tgds such that for every instance I_1 of $\mathbf{S}_1$, it holds that:

(C1) $\text{chase}_{\Sigma'_1}(I_1)$ is contained in $\text{chase}_{\Sigma_1}(I_1)$, and
(C2) $(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma'_1}(I_1)), \Sigma_2)$ is a minimal knowledge-base solution of (I_1, Σ_1).

Then for the input knowledge base (I, Σ_1), it materializes knowledge base $(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma'_1}(I)), \Sigma_2)$. Notice that in the previous approach, the minimal set Σ'_1 can be used for any source knowledge base (I, Σ_1). This is an important feature of our proposal, as

the computation of Σ_1' only depends on Σ_1, $\mathcal{M}$ and Σ_2, which are usually much smaller than the source explicit data. Besides, this is the most typical scenario in practice [19, 24], where for a specific domain the rules in a knowledge base remains unchanged, while the explicit data changes from one repository to another.

We now present the algorithm that given $\mathcal{M}$, Σ_1 and Σ_2 as above, returns a set Σ_1' of full tgds satisfying conditions (C1) and (C2). In the algorithm, we assume that Σ_1 is an acyclic set of full tgds, as in this case the problem of verifying whether conditions (C1) and (C2) hold for every instance I_1 of $\mathbf{S}_1$ is decidable in exponential time.

Algorithm: MINIMIZE$(\mathcal{M}, \Sigma_1, \Sigma_2)$

Input: $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, an acyclic set Σ_1 of full tgds, and a set Σ_2 of full tgds with inequalities that is safe for Σ_1 and $\mathcal{M}$.

Output: A minimal set Σ_1' that satisfies conditions (C1) and (C2) for every instance I_1 of $\mathbf{S}_1$.

1. Let Σ_1^+ be the set obtained by unfolding Σ_1, and $\Gamma = \Sigma_1^+$.
2. If there exists $\sigma \in \Gamma$ such that the set $\Sigma_1' = \Gamma \setminus \{\sigma\}$ satisfies conditions (C1) and (C2) for every instance I_1 of $\mathbf{S}_1$, then remove σ from Γ and repeat Step 2.
3. Let $\Sigma_1' = \Gamma$, and return Σ_1'. □

Notice that algorithm MINIMIZE can compute different outputs depending on the order in which the dependencies in Γ are chosen in Step 2. Also notice that we are searching for a minimal set in order to minimize the explicit data materialized in the target. Putting together procedures FULLSAFE and MINIMIZE, we can give a complete strategy to compute minimal knowledge-base solutions.

Theorem 8.11 *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and Σ_1 an acyclic set of full tgds over $\mathbf{S}_1$. Moreover, let Σ_2 be the output of* FULLSAFE$(\mathcal{M}, \Sigma_1)$*, and Σ_1' the output of* MINIMIZE$(\mathcal{M}, \Sigma_1, \Sigma_2)$*. Then for every instance I of $\mathbf{S}_1$ it holds that $\big(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1'}(I)),\ \Sigma_2\big)$ is a minimal knowledge-base solution of (I, Σ_1) under $\mathcal{M}$.*

Example 8.12. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and Σ_1 be as in Example 7.1. From Example 8.10, we know that the output of FULLSAFE$(\mathcal{M}, \Sigma_1)$ is the set Σ_2 consisting of dependency $F'(x,y) \wedge F'(y,z) \rightarrow GP'(x,z)$. It can be proved that there exists an order over the dependencies in Σ_1^+ such that the output of MINIMIZE$(\mathcal{M}, \Sigma_1, \Sigma_2)$ is the following set Σ_1' of dependencies:

$$\begin{aligned} M(x,y) &\rightarrow P(x,y) \\ P(x,y) \wedge P(y,z) &\rightarrow GP(x,z) \\ F(x,y) \wedge P(y,z) &\rightarrow GP(x,z) \\ P(x,y) \wedge F(y,z) &\rightarrow GP(x,z) \end{aligned}$$

Consider now the source instance I of Example 7.1, that is, $I = \{F(a,b), M(c,b), F(b,d)\}$. If we chase I with Σ_1', we obtain instance $I' = \{F(a,b), M(c,b), F(b,d), P(c,b), GP(c,d)\}$. If we now chase I' with Σ_{12}, we obtain the instance $J = \{F'(a,b), F'(b,d), GP'(c,d)\}$. Thus, we conclude from Theorem 8.11 that (J, Σ_2) is a minimal knowledge-base solution for (I, Σ_1) under $\mathcal{M}$. Notice that this is exactly the solution that we considered as a good solution in Example 7.1. □

9. CONCLUDING REMARKS

We have presented a framework to exchange data beyond the usual setting in which instances are considered to have complete information. We showed the robustness of our proposal by applying it to the problems of exchanging incomplete information and exchanging knowledge bases. In the former case, we proved several results regarding expressiveness, query answering and complexity of materializing solutions. In particular, we made the case that positive conditional instances are the right representation system to deal with the inherent incompleteness that emerges when exchanging data by using st-tgds. We also applied our framework to define the novel notion of knowledge exchange. This can be considered as a starting point for formalizing and studying the exchange of data in the Semantic Web, in particular, the exchange of RDFS graphs and OWL specifications. Many problems remain open. In particular, we would like to study knowledge exchange under mappings defined by non full st-tgds, which will probably require combining the results for knowledge bases and positive conditional instances.

Acknowledgments: We thank the anonymous referees for many helpful comments. Arenas was supported by Fondecyt grant 1090565, and Reutter by EPSRC grant G049165 and FET-Open project FoX.

10. REFERENCES

[1] S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *TCS*, 78(1):158–187, 1991.

[2] F. Afrati, C. Li, and V. Pavlaki. Data exchange: Query answering for incomplete data sources. In *InfoScale*, 2008.

[3] L. Antova, C. Koch, and D. Olteanu. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *ICDE*, pages 606–615, 2007.

[4] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Foundations of schema mapping management. In *PODS*, pages 227–238, 2010.

[5] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Inverting schema mappings: Bridging the gap between theory and practice. *PVLDB*, 2(1):1018–1029, 2009.

[6] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: Bringing exchanged data back. *TODS*, 34(4), 2009.

[7] P. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.

[8] P. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *SIGMOD*, pages 1–12, 2007.

[9] G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artif. Intell.*, 109(1-2):297–356, 1999.

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

[11] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. In *PODS*, pages 261–273, 1992.

[12] R. Fagin. Inverting schema mappings. *TODS*, 32(4), 2007.

[13] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *TCS*, 336(1):89–124, 2005.

[14] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *TODS*, 30(4):994–1055, 2005.

[15] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Quasi-inverses of schema mappings. In *PODS*, pages 123–132, 2007.

[16] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.

[17] G. Grahne. *The Problem of Incomplete Information in Relational Databases*. Springer, 1991.

[18] G. Grahne, A. Onet. Closed World Chasing. In *LID* 2011.

[19] P. Hayes. RDF Semantics, W3C Recommendation. February 2004.

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

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

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

[23] D. Maier, A. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *TODS*, 4(4):455–469, 1979.

[24] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language, W3C Recommendation. February 2004.

[25] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *TCS*, 51:53–80, 1987.