Data Exchange Beyond Complete Data

MARCELO ARENAS, Pontificia Universidad Católica de Chile
JORGE PÉREZ, Universidad de Chile
JUAN REUTTER, University of Edinburgh and Pontificia Universidad Católica de Chile

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 article, 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 [**Database Management**]: Heterogeneous Databases—*Data translation*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Data exchange, knowledge exchange, data integration, representation system, metadata management, schema mapping

ACM Reference Format:
Arenas, M., Pérez, J., and Reutter, J. 2013. Data exchange beyond complete data. J. ACM 60, 4, Article 28 (August 2013), 59 pages.
DOI: http://dx.doi.org/10.1145/2505985

1. INTRODUCTION

In the typical data exchange and data transformation settings, 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

M. Arenas was supported by Fondecyt grant 1090565; J. Pérez by Fondecyt grant 11110404 and by VID grant U-Inicia 11/04, Universidad de Chile; and J. Reutter by EPSRC grant G049165 and FET-Open project FoX.
Authors' addresses: M. Arenas and J. Reutter, Department of Computer Science, Pontificia Universidad Católica de Chile; J. Pérez, Department of Computer Science, Universidad de Chile. Correspondence email: marenas@ing.puc.cl.

DOI: http://dx.doi.org/10.1145/2505985

problem then that one wants to address is how to materialize an instance of the target schema that reflects the source data as accurately as possible [Fagin et al. 2005a]. 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. Although natural for a typical data translation scenario, this restriction is gradually becoming an impediment to a wide range of applications that need to exchange 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 [Fagin et al. 2005a]. 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 [Fagin et al. 2005a]. Thus, the preferred solutions in this scenario are *incomplete databases* [Imielinski and Lipski 1984]. But what if one needs to exchange data from these target instances with null values? What is the semantics of data exchange in this case? This issue has been raised before by Afrati et al. [2008] in the context of query answering and also by Fagin et al. [2009] in the context of metadata management. 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 simplest of 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 [Hayes 2004] or OWL specifications [Patel-Schneider et al. 2004]. 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. This immediately raises the question of 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 as 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 article, 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, 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 [Imielinski and Lipski 1984; Grahne 1991]. 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 compared with the classical setting of complete data [Imielinski and Lipski 1984; Grahne 1991]. 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 article, we study the representation system given by *positive conditional tables*, which are essentially conditional tables [Imielinski and Lipski 1984] 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 [Fagin et al. 2005a, 2009; Libkin 2006; Afrati et al. 2008; Libkin and Sirangelo 2008], 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 [Bernstein 2003; Bernstein and Melnik 2007].

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 this article 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.

It is important to mention that this article is a substantially extended version of Arenas et al. [2011]. Besides containing the complete proofs of all the results stated in Arenas et al. [2011], this version includes new material. In Section 4, we present a more detailed analysis of the features needed in positive conditional instances to obtain a strong representation system for st-tgds (Proposition 4.9). Sections 4 and 5.2 give a detailed description of the chase procedures for positive conditional instances and conditional instances, with both st-tgds and equality-generating dependencies, which are not explained in Arenas et al. [2011]. In this version of the article, we have also fixed an error in a result in Arenas et al. [2011], where it is stated that the problem of verifying, given a mapping $\mathcal{M}$ specified by st-tgds and positive conditional instances $\mathcal{I}$ and $\mathcal{J}$, whether $\mathcal{J}$ is a solution for $\mathcal{I}$ under $\mathcal{M}$ is in NP. The proof of this result was incorrect.[1] In fact, in Theorem 5.11 in this article, we prove that this problem is Π_2^P-complete. Section 8 also presents substantial new material. In particular, we present in this section algorithm OPTIMALSAFE (together with a proof of its correctness in Theorem 8.9) to compute *optimal-safe sets* in the context of knowledge exchange. This algorithm is not presented in Arenas et al. [2011].

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 $R_i^I \subseteq \mathbf{D}^{n_i}$. $\text{INST}(\mathbf{S})$ denotes the set of all instances of $\mathbf{S}$. We denote by $\text{dom}(I)$ 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 $\text{INST}(\mathbf{S}_1) \times \text{INST}(\mathbf{S}_2)$. 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 $\text{SOL}_{\mathcal{M}}(I)$. 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 $(I, J) \in \text{INST}(\mathbf{S}_1) \times \text{INST}(\mathbf{S}_2)$, 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 $\{(I, J) \in \text{INST}(\mathbf{S}_1) \times \text{INST}(\mathbf{S}_3) \mid$ there exists K such that $(I, K) \in \mathcal{M}_{12}$ and $(K, J) \in \mathcal{M}_{23}\}$ [Fagin et al. 2005b].

2.1. 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 article, we assume that all sets of dependencies are finite.

[1]This fact was kindly brought to our attention by Pablo Barceló.

2.2. 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 $Q(I) \subseteq \text{dom}(I)^k$. In this article, 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 unions 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, $\text{certain}_{\mathcal{M}}(Q, I)$ denotes the set of *certain answers* of Q over I under $\mathcal{M}$, that is, $\text{certain}_{\mathcal{M}}(Q, I) = \bigcap_{J \in \textsc{Sol}_{\mathcal{M}}(I)} Q(J)$.

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* [Imielinski and Lipski 1984; Antova et al. 2007], 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. [2005a] 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 $\text{rep}(\mathcal{W}) \subseteq \textsc{Inst}(\mathbf{S})$ [Imielinski and Lipski 1984]. 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 [Imielinski and Lipski 1984], and world-set decompositions [Antova et al. 2007].

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 $\textsc{Sol}_{\mathcal{M}}(\mathcal{X})$ as $\bigcup_{I \in \mathcal{X}} \textsc{Sol}_{\mathcal{M}}(I)$. That is, $\textsc{Sol}_{\mathcal{M}}(\mathcal{X})$ is the set of possible solutions for the instances in $\mathcal{X}$. In the following definition, we use $\textsc{Sol}_{\mathcal{M}}(\cdot)$ to extend the notion of solution to the case of representation systems.

Definition 3.1. Let $\mathcal{R} = (\mathbf{W}, \text{rep})$ 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 $\text{rep}(\mathcal{W}) \subseteq \textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{V}))$.

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 $\mathcal{R} = (\mathbf{W}, \text{rep})$ 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 $\text{rep}(\mathcal{W}) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{V}))$.

This new notion captures the intuition of exactly representing the space of possible solutions of the interpretations of an element of a representation system. Notice that if $\mathcal{M}$ is a set of st-tgds and $\mathcal{V}$ is a complete instance, this definition boils down to the notion of universal solutions given in Fagin et al. [2005a]. However, in the following sections, we show that this is generally not the case when $\mathcal{V}$ is not a complete instance.

3.2. Strong Representation Systems for a Class of Mappings

The classical work on incomplete databases [Imielinski and Lipski 1984] defines the notion of *strong representation system* for a class of queries. In fact, the classical result in [Imielinski and Lipski 1984] 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 $(\mathbf{W}, \text{rep})$ a representation system. Then, $(\mathbf{W}, \text{rep})$ 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 $\text{rep}(\mathcal{W}) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{U}))$.

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 article is to study the problem of exchanging databases with incomplete information. To this end, we first borrowed from Imielinski and Lipski [1984] and Antova et al. [2007] 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 [Fagin et al. 2005a, 2009; Libkin 2006], 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 [Fagin et al. 2005a, 2009; Libkin 2006], and conditional tables, which are known to be an expressive way to represent databases with incomplete information [Imielinski and Lipski 1984]. 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 article. 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 $R(t)$ [Imielinski and Lipski 1984]).

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 nulls($\mathcal{I}$) as the set of nulls mentioned in $\mathcal{I}$. If $\mathcal{I}$ is a conditional instance, nulls($\mathcal{I}$) also includes the nulls mentioned in the local conditions of $\mathcal{I}$ (conditional instances might contain nulls that occur only in their local conditions, see, e.g., Imielinski and Lipski [1984]). Moreover, given a null substitution $\nu : \text{nulls}(\mathcal{I}) \to \mathbf{D}$, 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 $\nu(n)$. Finally, given a naive instance $\mathcal{I}$ of a schema $\mathbf{S}$, an instance I of $\mathbf{S}$ and a null substitution $\nu : \text{nulls}(\mathcal{I}) \to \mathbf{D}$, say that $\nu(\mathcal{I})$ is contained in I, denoted by $\nu(\mathcal{I}) \subseteq I$, if $\nu(R^{\mathcal{I}}) \subseteq R^I$ for every $R \in \mathbf{S}$. Then, for every naive instance $\mathcal{I}$, and slightly abusing notation, define the set of representatives of $\mathcal{I}$, denoted by $\text{rep}_{\text{naive}}(\mathcal{I})$, as:

$$\{I \in \text{INST}(\mathbf{S}) \mid \text{there exists } \nu : \text{nulls}(\mathcal{I}) \to \mathbf{D} \text{ such that } \nu(\mathcal{I}) \subseteq I\}. \tag{1}$$

Moreover, for every conditional instance $\mathcal{I}$, define the set of representatives of $\mathcal{I}$, denoted by $\text{rep}_{\text{cond}}(\mathcal{I})$, 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 $\nu : \text{nulls}(\mathcal{I}) \to \mathbf{D}$ and $R \in \mathbf{S}$, define $\nu(R^{\mathcal{I}}, \rho_R^{\mathcal{I}})$ as $\{\nu(t) \mid t \in R^{\mathcal{I}} \text{ and } \nu \models \rho_R^{\mathcal{I}}(t)\}$. Finally, given a conditional instance $\mathcal{I}$ of a schema $\mathbf{S}$, an instance I of $\mathbf{S}$ and a null substitution $\nu : \text{nulls}(\mathcal{I}) \to \mathbf{D}$, say that $\nu(\mathcal{I})$ is contained in I, denoted by $\nu(\mathcal{I}) \subseteq I$, if $\nu(R^{\mathcal{I}}, \rho_R^{\mathcal{I}}) \subseteq R^I$ for every $R \in \mathbf{S}$. Then, $\text{rep}_{\text{cond}}(\mathcal{I})$ is defined again as in (1).

We use $\text{rep}_{\text{naive}}$ and rep_{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 $\mathcal{R}_{\text{naive}} = (\mathbf{W}_{\text{naive}}, \text{rep}_{\text{naive}})$ and $\mathcal{R}_{\text{cond}} = (\mathbf{W}_{\text{cond}}, \text{rep}_{\text{cond}})$ are representation systems.

We conclude this section by introducing a fragment of the class of conditional instances that will be extensively used in this article. 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 rep_{pos} the restriction of function rep_{cond} to the class of positive conditional instances, and by $\mathcal{R}_{\text{pos}}$ the representation system $(\mathbf{W}_{\text{pos}}, \text{rep}_{\text{pos}})$. When it is clear from the context, we just use rep instead of $\text{rep}_{\text{naive}}$, rep_{cond} or rep_{pos}.

4.2. Building a Strong Representation System for st-tgds

Fagin et al. [2005a] showed 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 $\text{rep}(\mathcal{I}_2) = \text{Sol}_{\mathcal{M}}(I_1)$. 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 $\text{rep}(\mathcal{I}_2) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}_1))$. Unfortunately, the following proposition shows that it is not the case.

PROPOSITION 4.1. *Naive instances do not form a strong representation system for the class of mappings specified by st-tgds.*

PROOF. 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. Next, we prove that if $\mathcal{J}$ is a naive instance of $\mathbf{S}_2$, then $\text{rep}(\mathcal{J}) \neq \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$. The intuition behind this proof 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.

For the sake of contradiction, assume that $\mathcal{J}$ is a naive instance over $\mathbf{S}_2$ such that $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$. We first prove that $T^{\mathcal{J}} = \emptyset$. If there exists a tuple t in $T^{\mathcal{J}}$, then for every $J \in \text{rep}(\mathcal{J})$, it holds that $T^J \neq \emptyset$. Let I_1 be an instance of $\mathbf{S}_1$ such that:

$$P^{I_1} = \{(c, a)\},$$

where c is an element from $\mathbf{D}$ such that $c \neq a$, and J_1 be an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} R^{J_1} &= \{(c, a)\}, \\ T^{J_1} &= \emptyset. \end{aligned}$$

Then, we have that $I_1 \in \text{rep}(\mathcal{I})$, $J_1 \in \text{Sol}_{\mathcal{M}}(I_1)$ and $J_1 \notin \text{rep}(\mathcal{J})$ (since $T^{J_1} = \emptyset$). But this contradicts our initial assumption that $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$.

Now let I_2 be an instance of $\mathbf{S}_1$ such that:

$$P^{I_2} = \{(a, a)\},$$

and J_2 be an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} R^{J_2} &= \{(a, a)\}, \\ T^{J_2} &= \{a\}. \end{aligned}$$

Given that $I_2 \in \text{rep}(\mathcal{I})$, $J_2 \in \text{Sol}_{\mathcal{M}}(I_2)$ and $\text{rep}(\mathcal{J}) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$, we conclude that $J_2 \in \text{rep}(\mathcal{J})$. Thus, there exists a null substitution $\nu : \text{nulls}(\mathcal{J}) \rightarrow \mathbf{D}$ such that $\nu(R^{\mathcal{J}}) \subseteq R^{J_2}$ and $\nu(T^{\mathcal{J}}) \subseteq T^{J_2}$. But then given that previously we prove that $T^{\mathcal{J}} = \emptyset$, we have that if J_3 is the instance:

$$\begin{aligned} R^{J_3} &= \{(a, a)\}, \\ T^{J_3} &= \emptyset, \end{aligned}$$

then $\nu(R^{\mathcal{J}}) \subseteq R^{J_3}$ and $\nu(T^{\mathcal{J}}) \subseteq T^{J_3}$, and, hence, $J_3 \in \text{rep}(\mathcal{J})$. Thus, given that $\text{rep}(\mathcal{J}) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$, we conclude that $J_3 \in \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$ and, therefore, there exists $I_3 \in$

rep($\mathcal{I}$) such that $J_3 \in \text{Sol}_{\mathcal{M}}(I_3)$. Given that (I_3, J_3) satisfies Σ_{12}, $R^{J_3} = \{(a, a)\}$ and $T^{J_3} = \emptyset$, we have that I_3 is the empty instance of $\mathbf{S}_1$. But this leads to a contradiction, since the empty instance of $\mathbf{S}_1$ is not a representative of $\mathcal{I}$. This concludes the proof of the proposition. □

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 article, we assume that $\top$ is an arbitrary element-condition that always holds (e.g., $n = n$ with $n \in \mathbf{N}$).

Example 4.2. Let $\mathcal{M}$ and $\mathcal{I}$ be as in the specification at the beginning of the proof of Proposition 4.1, and $\mathcal{J}$ be a conditional instance that contains the following facts and conditions in the relations R and T:

$$\begin{array}{ll} R(n, a) & \top \\ T(n) & n = a. \end{array}$$

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

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 nonpositive 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 4.3. *Positive conditional instances form a strong representation system for the class of mappings specified by st-tgds.*

In order to prove Theorem 4.3, we present an algorithm that, given a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of st-tgds, and a positive conditional instance $\mathcal{I}$ over $\mathbf{S}_1$, returns a positive conditional instance $\mathcal{J}$ that is a universal $\mathcal{R}_{\text{cond}}$-solution for $\mathcal{I}$ under $\mathcal{M}$. The algorithm is based on the *chase* procedure, as usual in data exchange [Fagin et al. 2005a]. In particular, our procedure is based on the chase algorithms presented in Grahne [1991], and is similar to the one recently proposed in Grahne and Onet [2011]. Notice that, as shown in Proposition 4.1 and Example 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. We first show the basic ideas behind our algorithm in an example.

Example 4.4. 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:

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

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 fact 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 $\mathcal{M}$. □

Naturally, one can ask whether the same procedure can be defined if we now assume that we start with any arbitrary (not necessarily positive) conditional instance. It turns out that this is indeed the case, and thus the same technique can also be used to show that conditional instances also form a strong representation system for the class of mappings specified by st-tgds. This gives 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.*

We now move to the definition of the algorithm and its proof of correctness, from which we obtain Theorem 4.3 and Theorem 4.5 as corollaries. The following terminology will be extensively used in our procedure. Given a conditional instance $\mathcal{I}$, the *domain* of $\mathcal{I}$, denoted by $\text{dom}(\mathcal{I})$, is the set of all elements that occur in any of the relations of $\mathcal{I}$ or in any element-condition of $\mathcal{I}$. Given a tuple $\bar{x}$, $|\bar{x}|$ denotes the length of $\bar{x}$, $x \in \bar{x}$ indicates that variable x is mentioned in $\bar{x}$, and $g : \bar{x} \to U$ indicates that g is a function from the variables mentioned in $\bar{x}$ to a set U. Moreover, given tuples $\bar{x}$ and $\bar{y}$ and a function $g : \bar{y} \to U$, $\bar{x} \subseteq \bar{y}$ indicates that every variable occurring in $\bar{x}$ is also mentioned in $\bar{y}$, $(\bar{x}, \bar{y})$ is a tuple obtained by putting the elements of $\bar{x}$ followed by the elements of $\bar{y}$, and $g(\bar{x})$ is a tuple obtained by replacing every $x \in \bar{x}$ by $g(x)$. Finally, given disjoint relational schemas $\mathbf{S}_1$ and $\mathbf{S}_2$, a CQ$^=$-TO-CQ dependency from $\mathbf{S}_1$ to $\mathbf{S}_2$ is a formula $\forall\bar{x}\forall\bar{y}\,(\varphi(\bar{x}, \bar{y}) \wedge \theta \to \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, where φ and ψ are conjunction of relational atoms over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, and θ is a conjunction of equalities between variables in $\bar{x}$ and $\bar{y}$. We usually omit the universal quantifiers of CQ$^=$-TO-CQ dependencies, just as we do with st-tgds. A mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ is *specified* by CQ$^=$-TO-CQ dependencies if Σ_{12} is a finite set of CQ$^=$-TO-CQ dependencies from $\mathbf{S}_1$ to $\mathbf{S}_2$. Notice that every st-tgd is a CQ$^=$-TO-CQ dependency. Thus, we lose no generality if we define our chase procedure for the class of CQ$^=$-TO-CQ dependencies.

In order to guarantee the correctness of the chase procedure, we require all dependencies to be in a normal form in which each variable in the premise appears *exactly once* in the relational atoms. Formally, assume that λ is a CQ$^=$-TO-CQ dependency $\varphi(\bar{x}, \bar{y}) \wedge \theta \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})$, where φ, ψ are conjunctions of relational atoms and θ is a conjunction of equalities. Then, λ has the *unique appearance property* if every variable in $(\bar{x}, \bar{y})$ occurs exactly once in $\varphi(\bar{x}, \bar{y})$. Notice that this requirement is without loss of generality, as every CQ$^=$-TO-CQ dependency can be transformed in polynomial time into an equivalent CQ$^=$-TO-CQ dependency having the unique appearance property. More specifically, assume that λ is a CQ$^=$-TO-CQ-dependency $\varphi(\bar{x}, \bar{y}) \wedge \theta \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})$ of the form described in this section. Start the transformation process with $\lambda' = \lambda$. Then, for every variable x in $\bar{x}$, let n be the number of times that x occurs in φ. Choose n fresh variables $x_1, \ldots, x_n$ and replace the ith occurrence of x in φ by x_i, for every $i \in \{1, \ldots, n\}$.

Next, replace all occurrences of x in θ and ψ by the variable x_1, and add to θ the formula $x_2 = x_1 \wedge \cdots \wedge x_n = x_1$. Repeat the procedure for every variable in $\bar{y}$. Clearly, λ' has the unique appearance property, and it is equivalent to λ.

We are now ready to define the chase procedure for conditional instances. The input of this procedure is a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of CQ$^=$-TO-CQ dependencies having the unique appearance property, and a conditional instance $\mathcal{I}$ of $\mathbf{S}_1$. The output of this procedure is a conditional instance $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ of $\mathbf{S}_2$ that is a universal $\mathcal{R}_{\text{cond}}$-solution for $\mathcal{I}$ under $\mathcal{M}$. The algorithm works as follows. It starts by initializing $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ as the empty conditional instance. Then, for every dependency λ in Σ_{12}, the algorithm includes the following elements into $\text{chase}_{\Sigma_{12}}(\mathcal{I})$. Assume that

$$\lambda \ = \ \varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{u}, \bar{v}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}),$$

where (1) $\varphi(\bar{x}, \bar{y}) = R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_k(\bar{x}_k, \bar{y}_k)$, (2) $\bar{x}_i \subseteq \bar{x}$ for every $i \in \{1, \ldots, k\}$, (3) $\bar{y}_i \subseteq \bar{y}$ for every $i \in \{1, \ldots, k\}$, (4) $\psi(\bar{x}, \bar{z}) = T_1(\bar{w}_1, \bar{z}_1) \wedge \cdots \wedge T_\ell(\bar{w}_\ell, \bar{z}_\ell)$, (5) $\bar{w}_j \subseteq \bar{x}$ for every $j \in \{1, \ldots, \ell\}$, (6) $\bar{z}_j \subseteq \bar{z}$ for every $j \in \{1, \ldots, \ell\}$, and (7) $\theta(\bar{u}, \bar{v})$ is a conjunction of equalities such that $\bar{u} \subseteq \bar{x}$ and $\bar{v} \subseteq \bar{y}$. Then, for every variable substitution $f : (\bar{x}, \bar{y}) \rightarrow \text{dom}(\mathcal{I})$, the algorithm verifies whether $(f(\bar{x}_i), f(\bar{y}_i)) \in R_i^{\mathcal{I}}$ for every $i \in \{1, \ldots, k\}$. If this is the case, it chooses a variable substitution $g : \bar{z} \rightarrow \mathbf{N}$ such that $g(\bar{z})$ is a tuple of pairwise distinct fresh null values, and then for every $j \in \{1, \ldots, \ell\}$, it includes the following elements into $\text{chase}_{\Sigma_{12}}(\mathcal{I})$.

—If relation $T_j^{\text{chase}_{\Sigma_{12}}(\mathcal{I})}$ does not contain the tuple $(f(\bar{w}_j), g(\bar{z}_j))$, then it is added, and element-condition $\rho_{T_j}^{\text{chase}_{\Sigma_{12}}(\mathcal{I})}(f(\bar{w}_j), g(\bar{z}_j))$ is initially defined as:

$$\rho_{R_1}^{\mathcal{I}}(f(\bar{x}_1),\ f(\bar{y}_1)) \wedge \cdots \wedge \rho_{R_k}^{\mathcal{I}}(f(\bar{x}_k),\ f(\bar{y}_k)) \wedge \theta(f(\bar{u}),\ f(\bar{v})). \tag{2}$$

—If relation $T_j^{\text{chase}_{\Sigma_{12}}(\mathcal{I})}$ already contains the tuple $(f(\bar{w}_j), g(\bar{z}_j))$, then formula (2) is added as a disjunct of the element-condition $\rho_{T_j}^{\text{chase}_{\Sigma_{12}}(\mathcal{I})}(f(\bar{w}_j), g(\bar{z}_j))$.

It should be noticed that formula $\theta(f(\bar{u}),\ f(\bar{v}))$ in (2) could not be, strictly speaking, an element-condition as it may contain comparisons between constant values (recall that element-conditions allow only comparisons between constants and nulls, and between nulls). For the sake of readability, we will use $\theta(f(\bar{u}),\ f(\bar{v}))$ in (2) as if it were an element-condition, as any comparison between constant values can be easily eliminated to generate an equivalent element-condition: $a = a$ and $a \neq a$ have to be replaced by $n = n$ and $n \neq n$, respectively, where n is a fixed null value, while $a = b$ and $a \neq b$ have to be replaced by $n \neq n$ and $n = n$, respectively, assuming that a and b are distinct constants.

Next we show the correctness of the chase procedure.

THEOREM 4.6. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of* CQ$^=$-TO-CQ *dependencies having the unique appearance property, and $\mathcal{I}$ be a conditional instance of* $\mathbf{S}_1$. *Then,* $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ *is a universal $\mathcal{R}_{cond}$-solution for $\mathcal{I}$ under $\mathcal{M}$.*

PROOF. In this proof, we use the following terminology. Given tuples $\bar{a} = (a_1, \ldots, a_\ell)$, $\bar{b} = (b_1, \ldots, b_\ell)$ and $\bar{c} = (a_{i_1}, \ldots, a_{i_k})$, where $1 \leq i_j \leq \ell$ for every $j \in \{1, \ldots, k\}$ (i.e., $\bar{c} \subseteq \bar{a}$), tuple $\pi_{\bar{c}}(\bar{b})$ is defined as $(b_{i_1}, \ldots, b_{i_k})$. Moreover, $\mathcal{J}$ is used to denote $\text{chase}_{\Sigma_{12}}(\mathcal{I})$. In what follows, we show that $\text{rep}(\mathcal{J}) = \text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$.

—First, we prove that $\text{rep}(\mathcal{J}) \subseteq \text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$. That is, we prove that for every $J \in \text{rep}(\mathcal{J})$, there exists an instance $I \in \text{rep}(\mathcal{I})$ such that $(I, J) \models \Sigma_{12}$.

Let $J \in \text{rep}(\mathcal{J})$, and assume that ν is a null substitution $\nu : \text{nulls}(\mathcal{J}) \rightarrow \mathbf{D}$ such that $\nu(\mathcal{J}) \subseteq J$. Then, let $I = \gamma(\mathcal{I})$, where γ is a substitution for $\mathcal{I}$ defined as follows.

For every $n \in \text{nulls}(\mathcal{J})$, if $n \in \text{nulls}(\mathcal{I})$, then let $\gamma(n) = \nu(n)$, and otherwise let $\gamma(n) = c$, where c is an arbitrary constant (we shall later see that these nulls are not important in the proof). Next, we prove that $(I, J) \models \Sigma_{12}$.

Consider an arbitrary dependency in Σ_{12} of the form $\varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{u}, \bar{v}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, where (1) $\varphi(\bar{x}, \bar{y}) = R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_k(\bar{x}_k, \bar{y}_k)$, (2) $\bar{x}_i \subseteq \bar{x}$ for every $i \in \{1, \ldots, k\}$, (3) $\bar{y}_i \subseteq \bar{y}$ for every $i \in \{1, \ldots, k\}$, (4) $\psi(\bar{x}, \bar{z}) = T_1(\bar{w}_1, \bar{z}_1) \wedge \cdots \wedge T_\ell(\bar{w}_\ell, \bar{z}_\ell)$, (5) $\bar{w}_j \subseteq \bar{x}$ for every $j \in \{1, \ldots, \ell\}$, (6) $\bar{z}_j \subseteq \bar{z}$ for every $j \in \{1, \ldots, \ell\}$, and (7) $\theta(\bar{u}, \bar{v})$ is a conjunction of equalities such that $\bar{u} \subseteq \bar{x}$ and $\bar{v} \subseteq \bar{y}$. Then, assume that $\bar{t}_1, \bar{t}_2$ is a pair of tuples of constants from $\text{dom}(I)$ such that $|\bar{t}_1| = |\bar{x}|$, $|\bar{t}_2| = |\bar{y}|$ and $I \models R_1(\pi_{\bar{x}_1}(\bar{t}_1), \pi_{\bar{y}_1}(\bar{t}_2)) \wedge \cdots \wedge R_k(\pi_{\bar{x}_k}(\bar{t}_1), \pi_{\bar{y}_k}(\bar{t}_2)) \wedge \theta(\pi_{\bar{u}}(\bar{t}_1), \pi_{\bar{v}}(\bar{t}_2))$. Next, we prove that there exists a tuple $\bar{t}_3$ such that $|t_3| = |\bar{z}|$ and $J \models \psi(\bar{t}_1, \bar{t}_3)$, from which we conclude that (I, J) satisfies dependency $\varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{u}, \bar{v}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, as $\bar{t}_1, \bar{t}_2$ are arbitrary tuples from $\text{dom}(I)$.

Given that $I = \gamma(\mathcal{I})$, there exist tuples $\bar{a}$, $\bar{b}$ from $\text{dom}(\mathcal{I})$ such that $\gamma(\bar{a}) = \bar{t}_1$, $\gamma(\bar{b}) = \bar{t}_2$ (recall $\bar{a}$ may contain constants as well as nulls and, thus, $\gamma(\bar{a})$ is obtained by replacing each null e in $\bar{a}$ by its image $\gamma(e)$, and likewise for $\bar{b}$). Thus, given that $I \models R_1(\pi_{\bar{x}_1}(\bar{t}_1), \pi_{\bar{y}_1}(\bar{t}_2)) \wedge \cdots \wedge R_k(\pi_{\bar{x}_k}(\bar{t}_1), \pi_{\bar{y}_k}(\bar{t}_2)) \wedge \theta(\pi_{\bar{u}}(\bar{t}_1), \pi_{\bar{v}}(\bar{t}_2))$, we conclude that $(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{y}_i}(\bar{b})) \in R_i^{\mathcal{I}}$ and $\gamma \models \rho^{\mathcal{I}}_{R_i}(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{y}_i}(\bar{b}))$ for every $i \in \{1, \ldots, k\}$, and we also conclude that $\gamma \models \theta(\pi_{\bar{u}}(\bar{a}), \pi_{\bar{v}}(\bar{b}))$. Next, we prove that $(\bar{a}, \bar{b})$ fires a step of the chase procedure. Define a function f from the variables mentioned in $\bar{x}$, $\bar{y}$ to the values mentioned in $\bar{a}$, $\bar{b}$ as follows.

—For every variable x mentioned in $\bar{x}$, if x occurs in $\bar{x}_i$ $(1 \leq i \leq k)$, then define $f(x)$ as the value assigned to variable x in the tuple $\pi_{\bar{x}_i}(\bar{a})$.
—For every variable y mentioned in $\bar{y}$, if y occurs in $\bar{y}_i$ $(1 \leq i \leq k)$, then define $f(y)$ as the value assigned to variable y in the tuple $\pi_{\bar{y}_i}(\bar{b})$.

It is important to notice that function f is well defined as $\varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{u}, \bar{v}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ has the unique appearance property and, thus, every variable mentioned in $\bar{x}$, $\bar{y}$ occurs exactly once in $\varphi(\bar{x}, \bar{y})$. Moreover, we have by definition of f that $f(\bar{x}) = \bar{a}$ and $f(\bar{y}) = \bar{b}$, from which we conclude that $f(\bar{x}_i) = \pi_{\bar{x}_i}(\bar{a})$ and $f(\bar{y}_i) = \pi_{\bar{y}_i}(\bar{b})$ for every $i \in \{1, \ldots, k\}$. Thus, given that $(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{y}_i}(\bar{b})) \in R_i^{\mathcal{I}}$ for every $i \in \{1, \ldots, k\}$, we have that $(f(\bar{x}_i), f(\bar{y}_i)) \in R_i^{\mathcal{I}}$ for every $i \in \{1, \ldots, k\}$. Hence, tuple $(\bar{a}, \bar{b})$ fires a step of the chase procedure that chooses a variable substitution $g : \bar{z} \to \mathbf{N}$ such that $g(\bar{z})$ is a tuple of pairwise distinct fresh null values, and then includes the facts $T_1(f(\bar{w}_1), g(\bar{z}_1)), \ldots, T_\ell(f(\bar{w}_\ell), g(\bar{z}_\ell))$ into $\mathcal{J}$, and it includes $\rho^{\mathcal{I}}_{R_1}(f(\bar{x}_1), f(\bar{y}_1)) \wedge \cdots \wedge \rho^{\mathcal{I}}_{R_k}(f(\bar{x}_k), f(\bar{y}_k)) \wedge \theta(f(\bar{u}), f(\bar{v}))$ as a disjunct of each element-condition $\rho^{\mathcal{J}}_{T_i}(f(\bar{w}_i), g(\bar{z}_i))$ $(1 \leq i \leq \ell)$.

Let $\bar{n} = g(\bar{z})$ and $\bar{t}_3 = \nu(\bar{n})$. Next, we prove that $J \models \psi(\bar{t}_1, \bar{t}_3)$. Given that $\gamma \models \rho^{\mathcal{I}}_{R_i}(\pi_{\bar{x}_i}(\bar{a}), \pi_{\bar{y}_i}(\bar{b}))$ for every $i \in \{1, \ldots, k\}$, we have that $\gamma \models \rho^{\mathcal{I}}_{R_i}(f(\bar{x}_i), f(\bar{y}_i))$ for every $i \in \{1, \ldots, k\}$ (recall that $f(\bar{x}_i) = \pi_{\bar{x}_i}(\bar{a})$ and $f(\bar{y}_i) = \pi_{\bar{y}_i}(\bar{b})$ for every $i \in \{1, \ldots, k\}$). Thus, given that γ and ν coincides on $\text{nulls}(\mathcal{I}) \cap \text{nulls}(\mathcal{J})$, we conclude that $\nu \models \rho^{\mathcal{I}}_{R_i}(f(\bar{x}_i), f(\bar{y}_i))$ for every $i \in \{1, \ldots, k\}$. Similarly, given that $\gamma \models \theta(\pi_{\bar{u}}(\bar{a}), \pi_{\bar{v}}(\bar{b}))$, we conclude that $\gamma \models \theta(f(\bar{u}), f(\bar{v}))$ and, hence, $\nu \models \theta(f(\bar{u}), f(\bar{v}))$. Thus, given that for every $i \in \{1, \ldots, \ell\}$, it holds that $(f(\bar{w}_i), g(\bar{z}_i)) \in T_i^{\mathcal{J}}$ and $\rho^{\mathcal{I}}_{R_1}(f(\bar{x}_1), f(\bar{y}_1)) \wedge \cdots \wedge \rho^{\mathcal{I}}_{R_k}(f(\bar{x}_k), f(\bar{y}_k)) \wedge \theta(f(\bar{u}), f(\bar{v}))$ is a disjunct in $\rho^{\mathcal{J}}_{T_i}(f(\bar{w}_i), g(\bar{z}_i))$, we conclude that $(\nu(f(\bar{w}_i)), \nu(g(\bar{z}_i))) \in T_i^{\nu(\mathcal{J})}$ for every $i \in \{1, \ldots, \ell\}$. Therefore, given that for every $i \in \{1, \ldots, \ell\}$:

$$\nu(f(\bar{w}_i)) = \nu(\pi_{\bar{w}_i}(\bar{a})) = \pi_{\bar{w}_i}(\nu(\bar{a})) = \pi_{\bar{w}_i}(\gamma(\bar{a})) = \pi_{\bar{w}_i}(\bar{t}_1),$$
$$\nu(g(\bar{z}_i)) = \nu(\pi_{\bar{z}_i}(\bar{n})) = \pi_{\bar{z}_i}(\nu(\bar{n})) = \pi_{\bar{z}_i}(\bar{t}_3),$$

we conclude that $(\pi_{\bar{w}_i}(t_1), \pi_{\bar{z}_i}(t_3)) \in T_i^J$ for every $i \in \{1, \ldots, \ell\}$ since $\nu(\mathcal{J}) \subseteq J$. From this we deduce that $J \models \psi(\bar{t}_1, \bar{t}_3)$, which was to be shown.

—Second, we prove that $\text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I})) \subseteq \text{rep}(\mathcal{J})$. More precisely, we prove that for every $I \in \text{rep}(\mathcal{I})$ and $J \in \text{SOL}_{\mathcal{M}}(I)$, there exists a null substitution $\nu : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu(\mathcal{J}) \subseteq J$, from which we conclude that $J \in \text{rep}(\mathcal{J})$.

Let I, J be instances of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, such that $I \in \text{rep}(\mathcal{I})$ and $J \in \text{SOL}_{\mathcal{M}}(I)$. Then, there exists a null substitution $\gamma : \text{nulls}(\mathcal{I}) \to \mathbf{D}$ such that $\gamma(\mathcal{I}) \subseteq I$. We use γ as follows to define a null substitution $\nu : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu(\mathcal{J}) \subseteq J$. For every element n in $\text{nulls}(\mathcal{I}) \cap \text{nulls}(\mathcal{J})$, let $\nu(n) = \gamma(n)$. For the rest of the null values in $\text{nulls}(\mathcal{J})$, substitution ν is defined according to the steps of the chase in which these nulls are generated. More precisely, let λ be a dependency in Σ_{12} of the form $\varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{u}, \bar{v}) \to \exists \bar{z} \psi(\bar{x}, \bar{z})$, where (1) $\varphi(\bar{x}, \bar{y}) = R_1(\bar{x}_1, \bar{y}_1) \wedge \cdots \wedge R_k(\bar{x}_k, \bar{y}_k)$, (2) $\bar{x}_i \subseteq \bar{x}$ for every $i \in \{1, \ldots, k\}$, (3) $\bar{y}_i \subseteq \bar{y}$ for every $i \in \{1, \ldots, k\}$, (4) $\psi(\bar{x}, \bar{z}) = T_1(\bar{w}_1, \bar{z}_1) \wedge \cdots \wedge T_\ell(\bar{w}_\ell, \bar{z}_\ell)$, (5) $\bar{w}_j \subseteq \bar{x}$ for every $j \in \{1, \ldots, \ell\}$, (6) $\bar{z}_j \subseteq \bar{z}$ for every $j \in \{1, \ldots, \ell\}$, and (7) $\theta(\bar{u}, \bar{v})$ is a conjunction of equalities such that $\bar{u} \subseteq \bar{x}$ and $\bar{v} \subseteq \bar{y}$. Moreover, let $f : (\bar{x}, \bar{y}) \to \text{dom}(\mathcal{I})$ be a variable substitution such that $(f(\bar{x}_i), f(\bar{y}_i)) \in R_i^{\mathcal{I}}$ for every $i \in \{1, \ldots, k\}$, and let $g : \bar{z} \to \mathbf{N}$ be a variable substitution such that $g(\bar{z}) = \bar{n}$ is a tuple of pairwise distinct fresh null values. Then, the value of null substitution ν over the nulls in $\bar{n}$ is defined by considering two cases.

—If γ does not satisfy the element condition $\rho^{\mathcal{I}}_{R_1}(f(\bar{x}_1), f(\bar{y}_1)) \wedge \cdots \wedge \rho^{\mathcal{I}}_{R_k}(f(\bar{x}_k), f(\bar{y}_k)) \wedge \theta(f(\bar{u}), f(\bar{v}))$, then let ν assign to each null in $\bar{n}$ an arbitrary constant (we shall later see that these nulls are not important in the proof).

—Assume that γ satisfies condition $\rho^{\mathcal{I}}_{R_1}(f(\bar{x}_1), f(\bar{y}_1)) \wedge \cdots \wedge \rho^{\mathcal{I}}_{R_k}(f(\bar{x}_k), f(\bar{y}_k)) \wedge \theta(f(\bar{u}), f(\bar{v}))$. Then, by definition of the chase procedure, it must be the case that for each $i \in \{1, \ldots, k\}$, conditional instances $\mathcal{I}$ contains the fact $R_i(f(\bar{x}_i), f(\bar{y}_i))$, and γ satisfies $\rho^{\mathcal{I}}_{R_i}(f(\bar{x}_i), f(\bar{y}_i))$. We thus obtain that $I \models \varphi(\gamma(f(\bar{x})), \gamma(f(\bar{y}))) \wedge \theta(\gamma(f(\bar{u})), \gamma(f(\bar{v})))$. Therefore, given that $J \in \text{SOL}_{\mathcal{M}}(I)$, there must be a tuple $\bar{c}$ (of constants) in $\text{dom}(J)$ of length $|\bar{z}|$ such that $J \models \psi(\gamma(f(\bar{x})), \bar{c})$. Then, define ν so that it maps each null n in $\bar{n}$ to the corresponding element in $\bar{c}$ (i.e., the element from $\bar{c}$ occurring in the same position as n).

Next, we prove that $\nu(\mathcal{J}) \subseteq J$. Assume that for a relation T_p $(1 \leq p \leq \ell)$ and a tuple $\bar{t}$ in $\text{dom}(\mathcal{J})$, it holds that $\nu(\bar{t}) \in T_p^{\nu(\mathcal{J})}$. We need to show that $\nu(\bar{t}) \in T_p^{J}$. Given that $\nu(\bar{t}) \in T_p^{\nu(\mathcal{J})}$, it holds that $\bar{t}$ is an element of $T_p^{\mathcal{J}}$ and $\nu \models \rho^{\mathcal{J}}_{T_p}(\bar{t})$. Thus, by the definition of the chase procedure, we have that there exists a disjunct β in $\rho^{\mathcal{J}}_{T_p}(\bar{t})$ that was generated in one of the steps of the chase, and such that $\nu \models \beta$. Assume that this step corresponds to a dependency λ defined as previously started, in which we use variable substitutions $f : (\bar{x}, \bar{y}) \to \text{dom}(\mathcal{I})$ and $g : \bar{z} \to \mathbf{N}$ (satisfying the same conditions as previously started). Then, $\mathcal{I}$ contains the fact $R_i(f(\bar{x}_i), f(\bar{y}_i))$, for every $i \in \{1, \ldots, k\}$. Moreover, given ν and γ coincide in $\text{nulls}(\mathcal{I}) \cap \text{nulls}(\mathcal{J})$ and $\nu \models \beta$, we have that $\gamma \models \beta$. Therefore, we have that γ satisfies $\rho^{\mathcal{I}}_{R_i}(f(\bar{x}_i), f(\bar{y}_i))$ for every $i \in \{1, \ldots, k\}$, and that γ also satisfies $\theta(f(\bar{u}), f(\bar{v}))$, from which we conclude that $I \models \varphi(\gamma(f(\bar{x})), \gamma(f(\bar{y}))) \wedge \theta(\gamma(f(\bar{u})), \gamma(f(\bar{v})))$. Hence, given that $J \in \text{SOL}_{\mathcal{M}}(I)$, there must be a tuple $\bar{c}$ (of constants) in $\text{dom}(J)$ of length $|\bar{z}|$ such that $J \models \psi(\gamma(f(\bar{x})), \bar{c})$. But we have defined ν to map the nulls in $\bar{t}$ to the element that corresponds to the same position in $\bar{c}$, so that $\nu(\bar{t})$ corresponds exactly to $(\pi_{\bar{w}_p}(\gamma(f(\bar{x}))), \pi_{\bar{z}_p}(\bar{c}))$, from which we deduce that $\nu(\bar{t})$ belongs to T_p^{J} as $J \models \psi(\gamma(f(\bar{x})), \bar{c})$. This concludes the proof of the theorem. □

It turns out that when starting with a positive conditional instance $\mathcal{I}$, one can completely avoid using negation in element-conditions generated by the chase procedure. More precisely, we had argued before that to deal with the comparison between constant values introduced by the chase, we need to replace conditions $a = a$ and $a \neq a$ by $n = n$ and $n \neq n$, respectively, where a is a constant and n is a fixed null value, while $a = b$ and $a \neq b$ have to be replaced by $n \neq n$ and $n = n$, respectively, assuming that a and b are distinct constants. Thus, if we start with a positive conditional instance, then the only inequality that may be generated by the chase is $n \neq n$. To avoid using this condition, the chase procedure can be modified as follows. Before adding an element-condition $\theta(f(\bar{u}), f(\bar{v}))$ in a step of this procedure, we *propagate* the condition $(n \neq n)$ over $\theta(f(\bar{u}), f(\bar{v}))$ by replacing $(n \neq n) \wedge \varphi$ by $(n \neq n)$ and $(n \neq n) \vee \varphi$ by φ, until either $(n \neq n)$ disappears or the result of the process is $(n \neq n)$. Once the propagation process has been finished, the resulting element-condition is used if it is not the element-condition $(n \neq n)$. For example, if we apply this procedure to the element-condition $(n_1 = n_2 \wedge n \neq n)$, then we obtain $(n \neq n)$ and no target tuples are generated in this step of the chase. On the other hand, if we apply it to the element-condition $(n_1 = n_2 \wedge n \neq n) \vee (n_1 = a)$, then we obtain $(n_1 = a)$, and the chase uses $(n_1 = a)$ when constructing the element-conditions of the target tuples.

From now on, we assume that we work with the modified version of the chase just introduced when considering positive conditional instances. In this case, it is clear that the chase introduces no inequalities in element-conditions and, thus, we have the following.

PROPOSITION 4.7. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of* CQ$^=$-TO-CQ *dependencies, and $\mathcal{I}$ be a positive conditional instance of $\mathbf{S}_1$. Then* $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ *is a positive conditional instance of $\mathbf{S}_2$.*

Theorem 4.3 now follows from this proposition and Theorem 4.6.

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 4.8. *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.*

The previous theorem is a corollary of the following stronger result, that shows that the three distinctive features of the local conditions of positive conditional instances are indeed needed to represent the spaces of solutions of naive instances, which are in turn needed to represent the spaces of solutions of complete instances.

PROPOSITION 4.9

(1) *There exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, and a naive instance $\mathcal{I}$ of $\mathbf{S}_1$ such that for every positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$ that is null-comparison free, it holds that rep($\mathcal{J}$) $\neq \text{SOL}_{\mathcal{M}}$(rep($\mathcal{I}$)).*
(2) *There exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, and a naive instance $\mathcal{I}$ of $\mathbf{S}_1$ such that for every positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$ that is null-constant-comparison free, it holds that rep($\mathcal{J}$) $\neq \text{SOL}_{\mathcal{M}}$(rep($\mathcal{I}$)).*
(3) *There exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds, and a naive instance $\mathcal{I}$ of $\mathbf{S}_1$ such that, for every positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$ that is disjunction free, it holds that rep($\mathcal{J}$) $\neq \text{SOL}_{\mathcal{M}}$(rep($\mathcal{I}$)).*

PROOF

(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:

$$P(x,y) \rightarrow R(x,y),$$
$$P(x,x) \rightarrow T(x).$$

Moreover, let $\mathcal{I}$ be a naive instance of $\mathbf{S}_1$ such that:

$$P^{\mathcal{I}} = \{(n_1,n_2)\},$$

where n_1 and n_2 are distinct null values. Next, we show that $\mathcal{M}$ and $\mathcal{I}$ satisfy the statement of the proposition.

For the sake of contradiction, assume that $\mathcal{J}$ is a positive conditional instance over $\mathbf{S}_2$ such that $\mathcal{J}$ is null-comparison free and $\text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$. Let $c \in \mathbf{D}$ be a constant that is not mentioned in any of the element-conditions of $\mathcal{J}$, and $\nu : \text{nulls}(\mathcal{J}) \rightarrow \mathbf{D}$ be a null substitution such that $\nu(n) = c$ for every $n \in \text{nulls}(\mathcal{J})$.

Let $t \in T^{\mathcal{J}}$. We first prove that $\nu \not\models \rho_T^{\mathcal{J}}(t)$. If $\rho_T^{\mathcal{J}}(t)$ is equal to $\top$, then $\nu(t)$ belongs to T^J for every $J \in \text{rep}(\mathcal{J})$. But then we conclude that $\text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I})) \neq \text{rep}(\mathcal{J})$ since if I_1 is an instance of $\mathbf{S}_1$ such that:

$$P^{I_1} = \{(c_1,c_2)\},$$

where c_1, c_2 are distinct elements from $\mathbf{D}$, and J_1 is an instance of $\mathbf{S}_2$ such that:

$$R^{J_1} = \{(c_1,c_2)\},$$
$$T^{J_1} = \emptyset,$$

then we have that $I_1 \in \text{rep}(\mathcal{I})$, $J_1 \in \text{SOL}_{\mathcal{M}}(I_1)$ and $J_1 \notin \text{rep}(\mathcal{J})$ (since $\nu(t) \notin T^{J_1}$). Thus, we conclude that $\rho_T^{\mathcal{J}}(t)$ is not equal to $\top$, from which we have that $\nu \not\models \rho_T^{\mathcal{J}}(t)$ since $\rho_T^{\mathcal{J}}(t)$ is a positive Boolean combination of conditions of the form $x = d$, where $x \in \mathbf{N}$ and $d \in \mathbf{D}$, and $\nu(x) = c$ with $c \neq d$.

Let $t \in R^{\mathcal{J}}$. Second, we prove that if $\nu \models \rho_R^{\mathcal{J}}(t)$, then $\rho_R^{\mathcal{J}}(t) = \top$ and $\nu(t) = (c,c)$. As above, we note that if $\rho_T^{\mathcal{J}}(t)$ is not equal to $\top$, then we have that $\nu \not\models \rho_T^{\mathcal{J}}(t)$ since $\rho_T^{\mathcal{J}}(t)$ is a positive Boolean combination of conditions of the form $x = d$, where $x \in \mathbf{N}$ and $d \in \mathbf{D}$, and $\nu(x) = c$ with $c \neq d$. Thus, we have that $\rho_R^{\mathcal{J}}(t) = \top$. Now assume, for the sake of contradiction, that $\nu(t) \neq (c,c)$. Moreover, assume without loss of generality that $\nu(t) = (c_1,c_2)$, where $c_1, c_2 \in \mathbf{D}$ and $c \neq c_1$. Then given that $\nu(n) = c$ for every $n \in \text{nulls}(\mathcal{J})$, we conclude that $t = (c_1, x)$, where $x \in (\mathbf{D} \cup \mathbf{N})$. Hence, given that $\rho_R^{\mathcal{J}}(t) = \top$, we have that for every $J \in \text{rep}(\mathcal{J})$, relation R^J includes a tuple of the form

(c_1, d), where $d \in \mathbf{D}$. But then we conclude that $\textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) \neq \text{rep}(\mathcal{J})$ since if I_2 is an instance of $\mathbf{S}_1$ such that:

$$P^{I_2} = \{(c, c)\},$$

and J_2 is an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} R^{J_2} &= \{(c, c)\}, \\ T^{J_2} &= \{c\}, \end{aligned}$$

then we have that $I_2 \in \text{rep}(\mathcal{I})$, $J_2 \in \textsc{Sol}_{\mathcal{M}}(I_2)$ and $J_2 \notin \text{rep}(\mathcal{J})$ (since R^{J_2} does not include a tuple of the form (c_1, d) with $d \in \mathbf{D}$). Thus, we have obtained a contradiction, from which we conclude that $\nu(t) = (c, c)$.

Let $J = \nu(\mathcal{J})$. From the previous two results, we conclude that either J is the empty instance of $\mathbf{S}_2$ or J is an instance of $\mathbf{S}_2$ such that $R^J = \{(c, c)\}$ and $T^J = \emptyset$. By definition of Σ_{12} and given that the empty instance of $\mathbf{S}_1$ is not in $\text{rep}(\mathcal{I})$, we conclude that J cannot be the empty instance of $\mathbf{S}_2$ (since $\textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$ and $J \in \text{rep}(\mathcal{J})$). Hence, we have that $R^J = \{(c, c)\}$ and $T^J = \emptyset$. Let I be an instance of $\mathbf{S}_1$ such that $I \in \text{rep}(\mathcal{I})$ and $J \in \textsc{Sol}_{\mathcal{M}}(I)$ (such an instance exists since $J \in \text{rep}(\mathcal{J})$ and $\textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$). Given that $I \in \text{rep}(\mathcal{I})$ and $R^J = \{(c, c)\}$, we have that $P^I = \{(c, c)\}$. But then we deduce that $c \in T^J$ since (I, J) satisfies dependency $P(x, x) \to T(x)$, which contradicts the fact that $T^J = \emptyset$. This concludes the proof of part (1) of the proposition.

(2) Let $\mathbf{S}_1 = \{A(\cdot), B(\cdot)\}$, $\mathbf{S}_2 = \{S(\cdot), T(\cdot)\}$ and Σ_{12} a set consisting of the following st-tgds:

$$\begin{aligned} A(x) \wedge B(x) &\to S(x), \\ B(x) &\to T(x). \end{aligned}$$

Moreover, let $\mathcal{I}$ be a naive instance over $\mathbf{S}_1$ such that:

$$\begin{aligned} A^{\mathcal{I}} &= \{a\}, \\ B^{\mathcal{I}} &= \{n\}, \end{aligned}$$

where $a \in \mathbf{D}$ and $n \in \mathbf{N}$. Next we show that $\mathcal{M}$ and $\mathcal{I}$ satisfy the statement of the proposition.

For the sake of contradiction, assume that $\mathcal{J}$ is a positive conditional instance over $\mathbf{S}_2$ that is null-constant-comparison free and $\textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$. Let $b \in \mathbf{D}$ such that $a \neq b$ and b is not mentioned in $\mathcal{J}$, I be an instance of $\mathbf{S}_1$ such that:

$$\begin{aligned} A^J &= \{a\}, \\ B^J &= \{b\}, \end{aligned}$$

and J be instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} S^J &= \emptyset, \\ T^J &= \{b\}. \end{aligned}$$

We have that $I \in \text{rep}(\mathcal{I})$ and $J \in \textsc{Sol}_{\mathcal{M}}(I)$. Thus, given that $\textsc{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$, we have that $J \in \text{rep}(\mathcal{J})$. Hence, there exists a substitution $\nu : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu(S^{\mathcal{J}}, \rho_S^{\mathcal{J}}) \subseteq S^J$ and $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) \subseteq T^J$. Let $\nu^\star : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ be a substitution defined as follows:

$$\nu^\star(x) = \begin{cases} a & \nu(x) = b \\ b & \nu(x) = a \\ \nu(x) & \nu(x) \neq a \text{ and } \nu(x) \neq b. \end{cases}$$

Moreover, let $J^\star$ be an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} S^{J^\star} &= \emptyset, \\ T^{J^\star} &= \{a\}. \end{aligned}$$

Then, given that b does not occur in $\mathcal{J}$ and $\mathcal{J}$ does not mention any condition of the form $x = c$ with $x \in \mathbf{N}$ and $c \in \mathbf{D}$, we conclude that $\nu^\star(S^{\mathcal{J}}, \rho_S^{\mathcal{J}}) \subseteq S^{J^\star}$ and $\nu^\star(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) \subseteq T^{J^\star}$ from the fact that $\nu(S^{\mathcal{J}}, \rho_S^{\mathcal{J}}) \subseteq S^J$ and $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) \subseteq T^J$. Therefore, we have that $J^\star \in \text{rep}(\mathcal{J})$, from which we conclude that $J^\star \in \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$ (since $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$). Thus, there exists $I^\star \in \text{rep}(\mathcal{I})$ such that $J^\star \in \text{Sol}_{\mathcal{M}}(I^\star)$. By definition of $\mathcal{I}$, we have that $B^{I^\star} \neq \emptyset$ and, hence, we have that $B^{I^\star} = \{a\}$ since $(I^\star, J^\star)$ satisfies dependency $B(x) \to T(x)$ and $T^{J^\star} = \{a\}$. Then, from the fact that $a \in A^{I^\star}$ (by definition of $\mathcal{I}$), we conclude that a must be in $S^{J^\star}$ since $(I^\star, J^\star)$ satisfies dependency $A(x) \wedge B(x) \to S(x)$. But this contradicts the fact that $S^{J^\star} = \emptyset$, and concludes the proof of part (2) of the proposition.

(3) Let $\mathbf{S}_1 = \{A(\cdot), B(\cdot), C(\cdot)\}$, $\mathbf{S}_2 = \{S(\cdot), T(\cdot)\}$ and Σ_{12} a set consisting of the following tgds:

$$\begin{aligned} A(x) \wedge B(y) \wedge C(y) &\to S(x), \\ B(x) &\to T(x). \end{aligned}$$

Moreover, let $\mathcal{I}$ be a naive instance over $\mathbf{S}_1$ such that:

$$\begin{aligned} A^{\mathcal{I}} &= \{a\}, \\ B^{\mathcal{I}} &= \{n\}, \\ C^{\mathcal{I}} &= \{c_1, c_2\}, \end{aligned}$$

where a, c_1, c_2 are pairwise distinct elements from $\mathbf{D}$ and $n \in \mathbf{N}$. Next, we show that $\mathcal{M}$ and $\mathcal{I}$ satisfy the statement of the proposition.

For the sake of contradiction, assume that $\mathcal{J}$ is a positive conditional instance over $\mathbf{S}_2$ such that $\mathcal{J}$ is disjunction free and $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$. Moreover, without loss of generality, assume that for every $R \in \mathbf{S}_2$ and $t \in R^{\mathcal{J}}$, it holds that either $\rho_R^{\mathcal{J}}(t) = \top$ or $\rho_R^{\mathcal{J}}(t)$ does not mention any condition of the form $x = x$ with $x \in \mathbf{N}$.

It is important to notice that, in the proof, we extensively use the fact that for every $R \in \mathbf{S}_2$ and $t \in R^{\mathcal{J}}$, element-condition $\rho_R^{\mathcal{J}}(t)$ is a conjunction of conditions of the form either $x = y$ with $x, y \in \mathbf{N}$ or $x = c$ with $x \in \mathbf{N}$ and $c \in \mathbf{D}$ (given that $\mathcal{J}$ is disjunction free). Thus, in particular, if a substitution ν does not satisfy a conjunct in $\rho_R^{\mathcal{J}}(t)$, then $\nu \not\models \rho_R^{\mathcal{J}}(t)$. The following four claims will be used in the proof of the proposition.

Claim 4.10

(1) *There exists $t \in T^{\mathcal{J}}$ such that $\rho_T^{\mathcal{J}}(t) = \top$.*
(2) *For every $J \in \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$, it holds that $T^J \neq \emptyset$.*
(3) *For every $J \in \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$, if $a \notin S^J$, then there exists an element $d \in T^J$ such that $d \neq c_1$ and $d \neq c_2$.*

Proof. The ideas in the proof of part (1) of Proposition 4.9 can be used to prove this claim. $\square$

Claim 4.11. *There exists a tuple $t \in S^{\mathcal{J}}$ such that $t = a$.*

PROOF. For the sake of contradiction, assume for every $t \in S^{\mathcal{J}}$, it holds that $t \neq a$. Then let I_1 be an instance of $\mathbf{S}_1$ such that:

$$\begin{aligned} A^{I_1} &= \{a\}, \\ B^{I_1} &= \{c_1\}, \\ C^{I_1} &= \{c_1, c_2\}, \end{aligned}$$

and let J_1 be an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} S^{J_1} &= \{a\}, \\ T^{J_1} &= \{c_1\}. \end{aligned}$$

Given that $I_1 \in \text{rep}(\mathcal{I})$, $(I_1, J_1) \models \Sigma_{12}$ and $\text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$, we conclude that $J_1 \in \text{rep}(\mathcal{J})$. Thus, there exists a substitution $\nu_1 : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu_1(\mathcal{J}) \subseteq J_1$. Let $c \in \mathbf{D}$ be a constant such that $c \neq a$, c is not mentioned in $\mathcal{J}$ and $c \neq \nu_1(x)$ for every $x \in \text{nulls}(\mathcal{J})$, and let $\nu_1^\star : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ be a substitution defined from ν_1 as follows:

$$\nu_1^\star(x) = \begin{cases} c & \nu_1(x) = a \\ \nu_1(x) & \nu_1(x) \neq a. \end{cases}$$

Finally, assume that $J_1^\star = \nu_1^\star(\mathcal{J})$. It is easy to see that for every $R \in \mathbf{S}_2$ and $t \in R^{\mathcal{J}}$, if $\nu_1^\star \models \rho_R^{\mathcal{J}}(t)$, then $\nu_1 \models \rho_R^{\mathcal{J}}(t)$. Thus, given that $T^{J_1} = \{c_1\}$, we conclude by condition (2) in Claim 4.10 and the definition of $\nu_1^\star$ that:

$$T^{J_1^\star} = \{c_1\}.$$

Moreover, given that $\nu_1^\star(x) \neq a$ for every $x \in \text{nulls}(\mathcal{J})$ and $t \neq a$ for every $t \in S^{\mathcal{J}}$, we conclude that $a \notin S^{J_1^\star}$. Since $J_1^\star \in \text{rep}(\mathcal{J})$ and $\text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$, we have that $J_1^\star \in \text{SOL}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$. But this contradicts condition (3) in Claim 4.10 since $a \notin S^{J_1^\star}$ and $T^{J_1^\star} = \{c_1\}$. □

Let E be a set of nulls recursive defined as follows: (1) if $t \in T^{\mathcal{J}}$, $t = n$ with $n \in \mathbf{N}$ and $\rho_T^{\mathcal{J}}(t) = \top$, then $n \in E$; (2) if $t \in T^{\mathcal{J}}$, $t = n$ with $n \in \mathbf{N}$, and every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form $x = y$ with $x, y \in E$, then $n \in E$. Moreover, for every $c \in \mathbf{D}$, let $E(c)$ be a set of nulls recursive defined as follows: (1) if $n \in E$, then $n \in E(c)$; (2) if $t \in T^{\mathcal{J}}$, $t = n$ with $n \in \mathbf{N}$, and every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form either $x = y$ with $x, y \in E(c)$ or $x = c$ with $x \in E(c)$, then $n \in E(c)$.

CLAIM 4.12

(1) *If d_1, d_2 are distinct elements from $\mathbf{D}$, then $E(d_1) \cap E(d_2) = E$.*
(2) *If $c \in \mathbf{D}$ and $\nu : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ is a substitution such that $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c\}$, then for every $x \in E(c)$, it holds that $\nu(x) = c$.*

PROOF

(1) By definition of $E(d_1)$ and $E(d_2)$, we have that $E \subseteq E(d_1) \cap E(d_2)$ and, thus, we only need to show that $E(d_1) \cap E(d_2) \subseteq E$. Next, we show by induction on the structure of $E(d_1)$ that for every $z \in E(d_1)$, if $z \in E(d_2)$, then $z \in E$.

—If $z \in E(d_1)$ because $z \in E$, then the property trivially holds.
—Assume that there exists a tuple $t \in T^{\mathcal{J}}$ such that $t = z$, every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form either $x = y$ with $x, y \in E(d_1)$ or $x = d_1$ with $x \in E(d_1)$. If $z \in E(d_2)$, then given that there is only one tuple t' in $T^{\mathcal{J}}$ such that $t' = z$, we have that every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form either $x = y$ with $x, y \in E(d_2)$ or $x = d_2$ with $x \in E(d_2)$. Thus, given that $d_1 \neq d_2$, we conclude that every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form $x = y$

with $x, y \in (E(d_1) \cap E(d_2))$. Therefore, from the induction hypothesis, we conclude that every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form $x = y$ with $x, y \in E$. Hence, we have that $z \in E$.

(2) We first prove by induction that for every $x \in E$, it holds that $\nu(x) = c$.

—Assume that there exists $t \in T^{\mathcal{J}}$ such that $t = x$ and $\rho_T^{\mathcal{J}}(t) = \top$. Then, we have that $\nu(t) \in \nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$ and, therefore, $\nu(x) = c$ since $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c\}$.

—Assume that there exists a tuple $t \in T^{\mathcal{J}}$ such that $t = x$, $\rho_T^{\mathcal{J}}(t) \neq \top$ and every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form $y = z$ with $y, z \in E$. Then, given that $\nu(u) = c$ for every variable u mentioned in $\rho_T^{\mathcal{J}}(t)$ (by induction hypothesis), we have that $\nu \models \rho_T^{\mathcal{J}}(t)$ and, therefore, $\nu(t) \in \nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$. Hence, we conclude that $\nu(x) = c$ since $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c\}$.

We now prove by induction that for every $x \in E(c)$, it holds that $\nu(x) = c$.

—If $x \in E$, then we have proved that $\nu(x) = c$.

—Assume that there exists a tuple $t \in T^{\mathcal{J}}$ such that $t = x$, every conjunct in $\rho_T^{\mathcal{J}}(t)$ is of the form either $y = z$ with $y, z \in E(c)$ or $y = c$ with $y \in E(c)$. Then given that $\nu(u) = c$ for every variable u mentioned in $\rho_T^{\mathcal{J}}(t)$ (by induction hypothesis), we have that $\nu \models \rho_T^{\mathcal{J}}(t)$ and, therefore, $\nu(t) \in \nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$. Hence, we conclude that $\nu(x) = c$ since $\nu(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c\}$. □

We have shown that there exists a tuple $t \in S^{\mathcal{J}}$ such that $t = a$. Given that $S^{\mathcal{J}}$ is a set, we conclude that such a tuple is unique. In what follows, we denote by t_a the only tuple in $S^{\mathcal{J}}$ such that $t_a = a$.

Claim 4.13

(1) *No conjunct in* $\rho_S^{\mathcal{J}}(t_a)$ *is of the form* $x = d$ *with* $x \in \mathbf{N}$ *and* $d \in \mathbf{D}$.
(2) *If* $x = y$ *is a conjunct in* $\rho_S^{\mathcal{J}}(t_a)$ *with* $x, y \in \mathbf{N}$, *then* $x, y \in E$.

Proof

(1) For the sake of contradiction, assume that $\rho_S^{\mathcal{J}}(t_a)$ mentions a conjunct of the form $u = d$ with $u \in \mathbf{N}$ and $d \in \mathbf{D}$. Moreover, assume, without loss of generality, that $d \neq c_1$. Next, we show that these assumptions lead to a contradiction.

For every $x \in \text{nulls}(\mathcal{J})$, let $c_x \in \mathbf{D}$ be a fresh constant (c_x is not mentioned in $\mathcal{J}$, $c_x \neq a$, $c_x \neq c_1$ and $c_x \neq c_y$ for every $y \in \text{nulls}(\mathcal{J})$ such that $x \neq y$). Moreover, let $\nu_2 : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ be a substitution defined as follows:

$$\nu_2(x) = \begin{cases} c_1 & x \in E(c_1) \\ c_x & \text{otherwise} \end{cases}$$

Finally, let $J_2 = \nu_2(\mathcal{J})$. Next, we show that $\nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c_1\}$. By condition (1) in Claim 4.10, we have that $\nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) \neq \emptyset$. Thus, to show that $\nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c_1\}$, we prove that for every $e \in \mathbf{D}$ such that $e \neq c_1$, it holds that $e \notin \nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$.

Assume, for the sake of contradiction, that there exists $e \in \mathbf{D}$ such that $e \neq c_1$ and $e \in \nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$. Then, there exists a tuple $t_0 \in T^{\mathcal{J}}$ such that $\nu_2 \models \rho_T^{\mathcal{J}}(t_0)$ and either $t_0 = e$ or $t_0 = z$ with $z \in \mathbf{N}$ and $\nu_2(z) = c_z = e$. Given that $\nu_2 \models \rho_T^{\mathcal{J}}(t_0)$, we have by definition of ν_2 that every conjunct in $\rho_T^{\mathcal{J}}(t_0)$ is of the form either $x = y$ with $x, y \in E(c_1)$ or $x = c_1$ with $x \in E(c_1)$. Next, we obtain a contradiction by considering two cases.

—Assume that $t_0 = z$ with $z \in \mathbf{N}$ such that $\nu_2(z) = c_z = e$. Then, given that every conjunct in $\rho_T^{\mathcal{J}}(t_0)$ is of the form either $x = y$ with $x, y \in E(c_1)$ or $x = c_1$ with $x \in E(c_1)$, we conclude that $z \in E(c_1)$. But then, by definition of ν_2, we conclude that $\nu_2(z) = c_1$, which leads to a contradiction since $c_1 \neq c_z$.

—Assume now that $t_0 = e$. Then, consider again instance J_1 defined in the proof of Claim 4.11. Given that $J_1 \in \text{rep}(\mathcal{J})$, there exists a substitution $\nu_1 : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu_1(\mathcal{J}) \subseteq J_1$. Since $\nu_1(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c_1\}$, we have by condition (2) in Claim 4.12 that $\nu_1(x) = c_1$ for every $x \in E(c_1)$. Thus, given that every conjunct in $\rho_T^{\mathcal{J}}(t_0)$ is of the form either $x = y$ with $x, y \in E(c_1)$ or $x = c_1$ with $x \in E(c_1)$, we conclude that $\nu_1 \models \rho_T^{\mathcal{J}}(t_0)$. But then we have that $e \in \nu_1(T^{\mathcal{J}}, \rho_T^{\mathcal{J}})$, which leads to a contradiction since $e \neq c_1$ and $\nu_1(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c_1\}$.

We just proved that $\nu_2(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{c_1\}$ and, thus, $T^{J_2} = \{c_1\}$ since $J_2 = \nu_2(\mathcal{J})$. Given that $\rho_T^{\mathcal{J}}(t_a)$ mentions a conjunct of the form $u = d$ with $u \in \mathbf{N}$ and $d \in \mathbf{D}$, we conclude that $\nu_2 \not\models \rho_S^{\mathcal{J}}(t_a)$ since $\nu_2(x) \neq d$ for every $x \in \text{nulls}(\mathcal{J})$ (recall that $c_1 \neq d$ and for every $x \in \text{nulls}(\mathcal{J})$, it holds that $c_x \neq d$ since d is mentioned in $\mathcal{J}$). Thus, given that t_a is the only tuple in $T^{\mathcal{J}}$ such that $t_a = a$ and $\nu_2(x) \neq a$ for every $x \in \text{nulls}(\mathcal{J})$, we conclude that $a \notin S^{J_2}$. But this contradicts condition (3) of Claim 4.10 since $J_2 \in \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$ (recall that $J_2 \in \text{rep}(\mathcal{J})$ and $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$), $a \notin S^{J_2}$ and $T^{J_2} = \{c_1\}$. This concludes the proof of condition (1) of the claim.

(2) Let ν_2 be the substitution defined in (1), and $J_2 = \nu_2(\mathcal{J})$. We proved in (1) that $T^{J_2} = \{c_1\}$. Thus, from condition (3) in Claim 4.10, we conclude that $a \in S^{J_2}$. Therefore, given that $\nu_2(x) \neq a$ for every $x \in \text{nulls}(\mathcal{J})$ and $S^{J_2} = \nu_2(S^{\mathcal{J}}, \rho_S^{\mathcal{J}})$, we conclude that $\nu_2 \models \rho_S^{\mathcal{J}}(t_a)$. Hence, if $x = y$ is a conjunct in $\rho_S^{\mathcal{J}}(t_a)$ with $x, y \in \mathbf{N}$, then we have that $x, y \in E(c_1)$ (by definition of ν_2). In the same way, it can be proved that if $x = y$ is a conjunct in $\rho_S^{\mathcal{J}}(t_a)$ with $x, y \in \mathbf{N}$, then $x, y \in E(c_2)$. Thus, given that $E(c_1) \cap E(c_2) = E$ (see condition (1) in Claim 4.12), we conclude that if $x = y$ is a conjunct in $\rho_S^{\mathcal{J}}(t_a)$ with $x, y \in \mathbf{N}$, then $x, y \in E$. □

We finally have all the necessary ingredients to prove the proposition. Let $d \in \mathbf{D}$ be a constant such that $d \neq c_1$ and $d \neq c_2$. Moreover, let $I^\star$ be an instance of $\mathbf{S}_1$ such that:

$$\begin{aligned} A^{I^\star} &= \{a\}, \\ B^{I^\star} &= \{d\}, \\ C^{I^\star} &= \{c_1, c_2\}, \end{aligned}$$

and $J^\star$ an instance of $\mathbf{S}_2$ such that:

$$\begin{aligned} S^{J^\star} &= \emptyset, \\ T^{J^\star} &= \{d\}. \end{aligned}$$

Given that $I^\star \in \text{rep}(\mathcal{I})$, $(I^\star, J^\star) \models \Sigma_{12}$ and $\text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I})) = \text{rep}(\mathcal{J})$, we conclude that $J^\star \in \text{rep}(\mathcal{J})$. Thus, there exists a substitution $\nu^\star : \text{nulls}(\mathcal{J}) \to \mathbf{D}$ such that $\nu^\star(\mathcal{J}) \subseteq J^\star$. In fact, by condition (2) of Claim 4.10 and definition of $J^\star$, we have that $\nu^\star(\mathcal{J}) = J^\star$. Thus, we have that $\nu^\star(T^{\mathcal{J}}, \rho_T^{\mathcal{J}}) = \{d\}$ and, therefore, $\nu^\star(x) = d$ for every $x \in E$ (by condition (2) of Claim 4.12 and given that $E \subseteq E(d)$). Hence, we conclude from conditions (1) and (2) of Claim 4.13 that $\nu^\star \models \rho_S^{\mathcal{J}}(t_a)$ and, thus, we have that $a \in \nu^\star(S^{\mathcal{J}}, \rho_S^{\mathcal{J}})$. We infer that $a \in S^{J^\star}$, which leads to a contradiction since $S^{J^\star} = \emptyset$. This concludes the proof of part (3) of the proposition.

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 [Fagin et al. 2005a], 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 the class of mappings specified by st-tgds, universal solutions have been identified as the preferred solutions [Fagin et al. 2005a], and polynomial-time algorithms have been developed to compute these solutions [Fagin et al. 2005a], 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 (recall that $\mathcal{R}_{\text{pos}}$ is the representation system consisting of positive conditional instance). In the following theorem, we show that the chase for conditional instances developed in Section 4 meets this requirements. It is important to notice that the schema mapping is assumed to be fixed in the theorem, which is the usual assumption when studying the complexity of materializing solutions in data exchange [Fagin et al. 2005a].

THEOREM 5.1. *Fix a mapping* $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$*, where* Σ_{12} *is a set of st-tgds. Then, for every positive conditional instance* $\mathcal{I}$ *of* $\mathbf{S}_1$*, one can compute its universal* $\mathcal{R}_{pos}$*-solution* $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ *in polynomial time.*

PROOF. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ be a mapping such that Σ_{12} is a set of st-tgds, and let $\mathcal{I}$ be a positive conditional instance of $\mathbf{S}_1$. First, we note that by using the algorithm described in Section 4.2, the set Σ_{12} can be transformed in polynomial time into a set Σ'_{12} of dependencies in CQ$^{=}$-TO-CQ, in which each dependency λ of Σ'_{12} has the unique appearance property. Consider a dependency λ in Σ'_{12} of form $\varphi(\bar{x}, \bar{y}) \wedge \theta(\bar{x}', \bar{y}') \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and assume that $\bar{x}$ is an n-ary tuple and $\bar{y}$ is a m-ary tuple. It is not difficult to see that the maximum number of chase steps that could be fired for λ is bounded by $|\operatorname{dom}(\mathcal{I})|^{n+m}$, which is polynomial since we are considering Σ_{12} fixed. Moreover, it is easy to see from the definition of the chase procedure presented in section 4.2 that each step can be performed in polynomial time. It follows that $\text{chase}_{\Sigma_{12}}(\mathcal{I})$ can be computed in polynomial time. □

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 Afrati et al. [2008] for the case of naive instances (which is the representation system used in Afrati et al. [2008]). 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 [Vardi 1982].

It was proved in Fagin et al. [2005a] 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 5.2. *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* $\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) = Q(\mathcal{J})$.

Proof. This proposition can be easily proved by considering that $\text{rep}(\mathcal{J}) = \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$, and that $\text{rep}(\mathcal{K}) \subseteq \text{Sol}_{\mathcal{M}}(\text{rep}(\mathcal{I}))$ for each positive conditional instance $\mathcal{K}$ that is an $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$. In fact, we have that:

$$\begin{aligned}
\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) &= \bigcap_{\mathcal{K}\,:\,\mathcal{K} \text{ is an } \mathcal{R}_{\text{pos}}\text{-solution for } \mathcal{I} \text{ under } \mathcal{M}} Q(\mathcal{K}) \\
&= \bigcap_{\mathcal{K}\,:\,\mathcal{K} \text{ is an } \mathcal{R}_{\text{pos}}\text{-solution for } \mathcal{I} \text{ under } \mathcal{M}} \left(\bigcap_{K \in \text{rep}(\mathcal{K})} Q(K) \right) \\
&= \bigcap_{K \in \text{rep}(\mathcal{J})} Q(K) \\
&= Q(\mathcal{J}).
\end{aligned}$$ □

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 Grahne [1991], it is possible to conclude that for every union of conjunctive queries Q, there exists a polynomial-time algorithm that, given a positive conditional instance $\mathcal{J}$, computes $Q(\mathcal{J})$. From these results, we conclude the following.

Theorem 5.3. *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 coincides with the upper bound in Fagin et al. [2005a] 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. [2005a] showed that this 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 Fagin et al. [2005a] for the case of positive conditional instances.

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 $\mathbf{S}_2$ with at most one inequality per disjunct. Then,* CERTAINANSWERS($\mathcal{M}$, Q) *can be solved in polynomial time.*

Before presenting the proof of the theorem, we introduce the necessary terminology, and describe the main strategy. To compute the certain answers, we use a strategy based on some ideas from Fagin et al. [2005a]. The main idea is as follows. Let $\mathcal{I}$ be a conditional instance of schema $\mathbf{S}_1$, and $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ be a mapping such that Σ_{12} is a set of st-tgds. First, notice that, to decide whether a tuple $\bar{t}$ belongs to $\text{certain}_{\mathcal{M}}(Q(\bar{x}), \mathcal{I})$, we can substitute $\bar{t}$ in Q, and check whether $\text{certain}_{\mathcal{M}}(Q(\bar{t}), \mathcal{I})$ is true. For this reason, we shall assume from now on that Q is a Boolean query.

Now, since query Q in the statement of the theorem is a union of conjunctive queries with at most one inequality per disjunct, we know that Q can be written as $Q_1 \vee Q_2$, where Q_1 is a disjunction of conjunctive queries (without inequalities), and Q_2 is a disjunction of conjunctive queries containing one inequality per disjunct. Now consider an arbitrary disjunct Q' of Q_2 of the form $\exists \bar{x}(\varphi(\bar{x}) \wedge x \neq x')$, where $\varphi(\bar{x})$ is a conjunction of relational atoms and variables x, x' are mentioned in $\bar{x}$. Then, the negation of Q' is equivalent to an *equality-generating* dependency, which is a dependency of the form

$$\varphi(\bar{x}) \rightarrow x = x'.$$

Thus, the negation of Q_2 can be represented as a set Σ_{Q_2} of equality-generating dependencies. In a nutshell, in order to compute the certain answers of Q, the idea is to *chase* the universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$ with the set Σ_{Q_2} of equality-generating dependencies, in order to obtain a conditional instance $\mathcal{J}$ that represents all the solutions for $\mathcal{I}$ under $\mathcal{M}$ that satisfy Σ_{Q_2} (and thus, falsify Q_2), and then pose the query Q_1 over the conditional instance $\mathcal{J}$.

Before describing the algorithm in detail, we need to formalize the notion of *chasing* a conditional instance with a set of equality-generating dependencies. Let $\mathcal{I}$ be a positive conditional instance of a schema $\mathbf{S}$, and Σ be a set of equality-generating dependencies over $\mathbf{S}$. Notice that our function rep is open to the addition of new tuples, and thus we cannot hope for a representation that captures *exactly* the set of instances in $\text{rep}(\mathcal{I})$ that satisfies Σ. However, as the following example illustrates, we can always obtain a conditional instance that represents the *minimal models* of the set $\{I \in \text{rep}(\mathcal{I}) \mid I \models \Sigma\}$; that is, a conditional instance $\mathcal{J}$ such that (1) every $I \in \text{rep}(\mathcal{I})$ that satisfies Σ belongs to $\text{rep}(\mathcal{J})$, (2) all instances in $\text{rep}(\mathcal{J})$ belong to $\text{rep}(\mathcal{I})$, and (3) $\nu(\mathcal{J}) \models \Sigma$, for every valid substitution ν of $\mathcal{J}$.

Example 5.5. Let $\mathbf{S}_1 = \{P(\cdot,\cdot), R(\cdot)\}$, and Σ a set that contains only the equality-generating dependency:

$$P(x, y) \wedge R(x) \ \rightarrow \ x = y.$$

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

$$\begin{array}{ll} P(n_1, n_2) & \top \\ R(n_3) & \top. \end{array}$$

Then the conditional instance $\mathcal{J}$ given by:

$$\begin{array}{ll} P(n_1, n_2) & (n_1 \neq n_3) \\ P(n_1, n_1) & (n_1 = n_3) \\ R(n_3) & \top. \end{array}$$

satisfies the conditions (1), (2), and (3) given previously. □

In Example 5.5, we started with a positive conditional instance, and ended up with a conditional instance with an inequality in its element-conditions. Thus, unfortunately, this approach is not useful for querying answering purposes, as it is well known that, in general, querying conditional tables can be intractable [Imielinski and Lipski 1984; Grahne 1991]. Instead, we follow a different approach, and extend the definition of conditional instances allowing them to contain *global conditions* [Grahne 1991]. It should be noticed that we only use conditional instances with global conditions as a tool to prove Theorem 5.4 (we are not introducing them as a new representation system to study).

Formally, given a schema $\mathbf{S}$, a conditional instance $\mathcal{I}$ of $\mathbf{S}$ with global condition is formed by an element-condition $\xi_{\mathcal{I}}$ and an assignment $(R^{\mathcal{I}}, \rho_R^{\mathcal{I}})$ for every relation symbol $R \in \mathbf{S}$, where $R^{\mathcal{I}} \subseteq (\mathbf{D} \cup \mathbf{N})^k$ if k is the arity of R, and $\rho_R^{\mathcal{I}}$ is a function that assigns to each tuple $t \in R^{\mathcal{I}}$ an element-condition $\rho_R^{\mathcal{I}}(t)$. Moreover, we also need to redefine the function rep_{cond} that assigns to every conditional instance in $\mathbf{W}_{\text{cond}}$ a set of instances. Let $\mathcal{I}$ be a conditional instance of a relational schema $\mathbf{S}$ with global element-condition $\xi_{\mathcal{I}}$. Given a null substitution $\nu : \text{nulls}(\mathcal{I}) \to \mathbf{D}$ and $R \in \mathbf{S}$, recall that we 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, $\text{rep}_{\text{cond}}(\mathcal{I})$ is the following set of instances:

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

Example 5.6 (*Example 5.5 continued*). The conditional instance $\mathcal{J}$ with global condition given by

$$\begin{array}{ll} P(n_1, n_2) & \top \\ R(n_3) & \top \end{array}$$

and $\xi_{\mathcal{J}} = ((n_1 \neq n_3) \vee (n_1 = n_2))$, satisfies the conditions (1), (2), and (3) that state that $\mathcal{J}$ captures the *minimal models* of $\mathcal{I}$ and Σ. □

From Example 5.6, one can infer that, if $\mathcal{I}$ is a positive conditional instance and Σ is a set of equality-generating dependencies, then it is possible to represent the minimal models of $\mathcal{I}$ and Σ with a conditional instance in which all element-conditions except for the global condition are positive. Moreover, it can also be shown that the global condition that is needed is not arbitrary, but it is always a conjunction of conditions of form

$$\psi \vee (x = y),$$

where ψ is a positive Boolean combination (only connectives $\wedge$ and $\vee$ are allowed) of formulas of the form $u \neq v$. We shall denote these element-conditions as *Horn* element-conditions. Furthermore, we say that a conditional instance $\mathcal{I}$ of a schema $\mathbf{S}$ with global condition is Horn if (1) $\xi_{\mathcal{I}}$ is either $\top$ or a nonempty conjunction of Horn element-conditions, and (2) for every $R \in \mathbf{S}$, it holds that $\rho_R^{\mathcal{I}}(t)$ is a positive element-condition for every $t \in R^{\mathcal{I}}$ (notice that positive conditional instances $\mathcal{I}$ are a special case of Horn instances in which $\xi_{\mathcal{I}} = \top$).

The reason we prefer to use Horn conditional instances, instead of arbitrary conditional instances (such as the one given in Example 5.5), is made clear with the following result (which proof can be found in the appendix).

PROPOSITION 5.7. *Let $\mathcal{I}$ be a Horn conditional instance of a schema $\mathbf{S}$, and Q a union of conjunctive queries over $\mathbf{S}$. Then $Q(\mathcal{I}) = \bigcap_{I \in rep(\mathcal{I})} Q(I)$ can be computed in polynomial time.*

Thus, regarding query answering, Horn conditional instances are as good as positive conditional instances. Moreover, the following proposition confirms the intuition that Horn conditional instances are enough for representing the minimal models of a positive conditional instance $\mathcal{I}$ and a set of equality-generating dependencies Σ. The proof of the proposition is in the appendix.

PROPOSITION 5.8. *Let $\mathcal{I}$ be a positive conditional instance of a schema* $\mathbf{S}$*, and Σ a set of equality-generating dependencies over* $\mathbf{S}$. *Then, there is a polynomial-time procedure that satisfies the following:*

—*If rep($\mathcal{I}$) $\cap$ $\{I \in \text{INST}(\mathbf{S}) \mid I \models \Sigma\}$ is not empty, then the procedure computes a Horn conditional instance $\mathcal{J}$, such that* (1) *every $I \in$ rep($\mathcal{I}$) that satisfies Σ belongs to rep($\mathcal{J}$),* (2) *all instances in rep($\mathcal{J}$) belong to rep($\mathcal{I}$), and* (3) *$\nu(J) \models \Sigma$, for every valid substitution ν of $\mathcal{J}$.*
—*If rep($\mathcal{I}$) $\cap$ $\{I \in \text{INST}(\mathbf{S}) \mid I \models \Sigma\}$ is empty, then the procedure stops and returns* failure.

We now have all the necessary ingredients to prove Theorem 5.4.

PROOF OF THEOREM 5.4. As explained previously, let $Q = Q_1 \vee Q_2$, where Q_1 is a disjunction of conjunctive queries (without inequalities), and Q_2 is a disjunction of conjunctive queries each conjunctive query with exactly one inequality. Then, the negation of Q_2 is equivalent to the conjunction of a set Σ_{Q_2} of equality-generating dependencies. Similarly, as in Fagin et al. [2005a], we can use the following algorithm to solve the certain answers problem for a conjunctive query Q for a given positive conditional instance $\mathcal{I}$ over a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$.

(1) Let $\mathcal{J} = \text{chase}_{\Sigma_{12}}(\mathcal{I})$. Then compute $\mathcal{K}$ so that it represents the minimal models of $\mathcal{J}$ and Σ_{Q_2}, by using the procedure in Proposition 5.8.
(2) If the procedure returns *failure*, then answer $\underline{\text{true}}$.
(3) If the procedure returns an instance $\mathcal{K}$, then answer $\underline{\text{true}}$ if $Q_1(\mathcal{K}) = \underline{\text{true}}$, and answer $\underline{\text{false}}$ if $Q_1(\mathcal{K}) = \underline{\text{false}}$

We now show that the described procedure correctly computes the certain answer for Q over $\mathcal{I}$. We have the following three cases.

—The procedure returns failure. By Proposition 5.8, we have that there is no solution J in rep($\mathcal{J}$) such that $J \models \Sigma_{Q_2}$. It follows that every solution $J \in$ rep($\mathcal{J}$) satisfies Q_2, and since rep($\mathcal{J}$) = $\text{SOL}_{\mathcal{M}}$(rep($\mathcal{I}$)), we obtain that $\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) = \underline{\text{true}}$.
—The procedure returns $\mathcal{K}$, and it is the case that $Q_1(\mathcal{K}) = \underline{\text{true}}$. Since rep($\mathcal{J}$) $\bigcap \{J \in \text{INST}(\mathbf{S}_2) \mid J \models \Sigma_{Q_2}\} \subseteq$ rep($\mathcal{K}$) and $\mathcal{J}$ is a universal solution, we have that in every solution K for $\mathcal{I}$ under $\mathcal{M}$ that satisfies Q_2 it is the case that $Q_1(K) = \underline{\text{true}}$. Then for every solution J of $\mathcal{I}$ over $\mathcal{M}$, it is either the case that $J \not\models \Sigma_{Q_2}$, or $Q_1(J) = \underline{\text{true}}$, or in other words, either J satisfies Q_2, or J satisfies Q_1. We conclude that $\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) = \underline{\text{true}}$.
—The procedure returns $\mathcal{K}$, and it is the case that $Q_1(\mathcal{K}) = \underline{\text{false}}$. Then, there exists an instance $K \in$ rep($\mathcal{K}$) such that $Q_1(K) = \underline{\text{false}}$. Then, let ν be a valid substitution for $\mathcal{K}$ such that $\nu(\mathcal{K}) \subseteq K$. Since Q_1 is a monotone query, it follows that $Q_1(\nu(\mathcal{K})) = \underline{\text{false}}$. Furthermore, by Proposition 5.8, we have that $\nu(\mathcal{K})$ satisfies Σ_{Q_2}, and thus $\nu(\mathcal{K})$ does not satisfy any of the disjuncts in Q_2. Since, by Proposition 5.8, we know that $\nu(\mathcal{K})$ is a solution for $\mathcal{I}$ under $\mathcal{M}$, we have that $\text{certain}_{\mathcal{M}}(Q, \mathcal{I}) = \underline{\text{false}}$.

That this algorithm runs in polynomial time follows from the facts that (1) $\mathcal{J}$ and $\mathcal{K}$ can be computed in polynomial time (Theorem 5.1 and Proposition 5.8, respectively), (2) computing the certain answers of a union of conjunctive queries over a Horn conditional instance is in polynomial time (Proposition 5.7). □

5.3. Complexity of Recognizing Solutions

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 whether $(I, J) \models \Sigma_{12}$, which gives a straightforward polynomial-time procedure when Σ_{12} is a set of FO sentences. For the case of naive instances, positive conditional instances and conditional instances, this problem becomes more challenging. In this section, we study the complexity of this problem when the inputs are not necessarily complete instances. One of the motivations of studying this problem is to establish a complexity-theoretic separation between positive conditional instances and (general) conditional instances. This gives evidence in favor of using positive conditional instances instead of conditional instances as a representation system for st-tgds.

In some proofs in this section, and also in some proofs in Section 6, we use the notions of *closed-down on the left* and *closed-up on the right* mappings [ten Cate and Kolaitis 2010]. 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 $(I, J') \in \mathcal{M}$. Another notion that we need is *closure under target homomorphisms* [ten Cate and Kolaitis 2010], which generalizes the property of being closed-up on the right. Given instances I and J of the same schema $\mathbf{S}$, a homomorphism h from I to J is a function $h : \text{dom}(I) \to \text{dom}(J)$ such that for every relational symbol R in $\mathbf{S}$ and tuple $t \in R^I$, it holds that $h(t) \in R^J$. Moreover, given a set $K \subseteq \mathbf{D}$, we say that a homomorphism h is the identity over K, if $h(a) = a$ for every element $a \in K$. A mapping $\mathcal{M}$ is closed under target homomorphisms if for every $(I, J) \in \mathcal{M}$ and homomorphism $h : J \to J'$ such that h is the identity over $\text{dom}(I) \cap \text{dom}(J)$, it holds that $(I, J') \in \mathcal{M}$ [ten Cate and Kolaitis 2010]. It is known that every mapping $\mathcal{M}$ specified by a set of st-tgds is closed-down on the left, closed-up on the right, and closed under target homomorphisms [ten Cate and Kolaitis 2010].

Our first result states that checking whether a positive conditional instance is an $\mathcal{R}_{\text{pos}}$-solution of an instance with complete information, which is represented as a naive instance without null values, can be solved in polynomial time.

Proposition 5.9. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then, the problem of verifying, given a naive instance I of $\mathbf{S}_1$ without null values and a positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$, whether $\mathcal{J}$ is an $\mathcal{R}_{pos}$-solution for I under $\mathcal{M}$ can be solved in polynomial time.*

Proof. Let $\mathcal{J}$ be a positive conditional instance and I a naive-instance without null values. Consider the instance $\nu(\mathcal{J})$ where ν is a substitution that replaces every null value in $\mathcal{J}$ by a fresh constant (appearing neither in I nor in $\mathcal{J}$). Notice that ν falsifies all element conditions in $\mathcal{J}$ except for those that are equivalent to $\top$. Thus, every tuple in $\nu(\mathcal{J})$ is a tuple of the form $\nu(\bar{t})$ such that $\bar{t}$ has element condition equivalent to $\top$ in $\mathcal{J}$. We claim that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of I under $\mathcal{M}$ if and only if $(I, \nu(\mathcal{J})) \models \Sigma_{12}$.

Assume first that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of I under $\mathcal{M}$. We need to prove that $(I, \nu(\mathcal{J})) \models \Sigma_{12}$. Since $\nu(\mathcal{J}) \in \text{rep}(\mathcal{J})$, we know that there exists an instance $K \in \text{rep}(I)$ such that $(K, \nu(\mathcal{J})) \models \Sigma_{12}$. Now given that $K \in \text{rep}(I)$, and since I is a complete instance we know that $I \subseteq K$. Thus, we have that $I \subseteq K$ and $(K, \nu(\mathcal{J})) \models \Sigma_{12}$, and then since $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ is given by st-tgds we know that it is closed-down on the left, thus obtaining $(I, \nu(\mathcal{J})) \models \Sigma_{12}$. This completes this part of the proof.

Assume now that $(I, \nu(\mathcal{J})) \models \Sigma_{12}$. We need to prove that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of I under $\mathcal{M}$. For this, we prove that, for every $J \in \text{rep}(\mathcal{J})$, there exists $K \in \text{rep}(I)$ such that $(K, J) \models \Sigma_{12}$. Let $J \in \text{rep}(\mathcal{J})$, then there exists a substitution ξ such that $\xi(\mathcal{J}) \subseteq J$. We next prove that there exists an instance $K \in \text{rep}(I)$ such that $(K, \xi(\mathcal{J})) \models \Sigma_{12}$. Notice that since $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ is specified by st-tgds, then it is closed-up on the right, and thus from $(K, \xi(\mathcal{J})) \models \Sigma_{12}$ we obtain that $(K, J) \models \Sigma_{12}$. Consider now a function h with domain $\text{dom}(\nu(\mathcal{J}))$ such that $h(\nu(x)) = \xi(x)$. Since ν assigns a fresh constant to every null in $\mathcal{J}$ (and is the identity over the constants in $\mathcal{J}$), we have that function h is well defined. Moreover, we know that for every tuple $\nu(\bar{t})$ in $\nu(\mathcal{J})$ it holds that $\bar{t}$ has element condition equivalent to $\top$ in $\mathcal{J}$, and thus, for every $\nu(\bar{t})$ in $\nu(\mathcal{J})$ the tuple $\xi(\bar{t})$ is in $\xi(\mathcal{J})$. Furthermore, by the definition of h, we obtain that $h(\nu(\bar{t})) = \xi(\bar{t})$ for every $\nu(\bar{t})$ in $\nu(\mathcal{J})$, implying that $h(\nu(\mathcal{J})) \subseteq \xi(\mathcal{J})$. Thus, notice that h is a function from $\nu(\mathcal{J})$ to $\xi(\mathcal{J})$ that is the identity over all the constants mentioned in $\mathcal{J}$ (and maps the fresh constants in $\nu(\mathcal{J})$ to other constants in $\xi(\mathcal{J})$). In particular, h is a function that is the identity over all the values in I. Thus, since $(I, \nu(\mathcal{J})) \models \Sigma_{12}$ and since mappings given by st-tgds are closed under target homomorphisms, we obtain that $(I, h(\nu(\mathcal{J}))) \models \Sigma_{12}$, and thus, $(I, \xi(\mathcal{J})) \models \Sigma_{12}$.

Finally, notice that for every element condition α in $\mathcal{J}$ we can check in polynomial time whether α is equivalent to $\top$ by just replacing equalities of the form $n = n$ by $\underline{\text{true}}$, equalities of the form $n = m$ by $\underline{\text{false}}$, and then checking if the evaluation of the resulting formula is $\underline{\text{true}}$ (see Lemma A.2 in the appendix). Thus, in order to check whether $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of I, we first construct $\nu(\mathcal{J})$ in polynomial time, and then test if $(I, \nu(\mathcal{J})) \models \Sigma_{12}$ which can be done in polynomial time since Σ_{12} is fixed. □

Our next result shows that the problem considered in Proposition 5.9 becomes coNP-complete if $\mathcal{J}$ is a conditional instance (instead of a positive conditional instance).

PROPOSITION 5.10. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then, the problem of verifying, given a naive instance I of $\mathbf{S}_1$ without null values and a conditional instance $\mathcal{J}$ of $\mathbf{S}_2$, whether $\mathcal{J}$ is an $\mathcal{R}_{pos}$-solution for I under $\mathcal{M}$ is coNP-complete.*

PROOF. We first show that the problem is in coNP. Thus, assume that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of I. Then, there exists an instance K and a substitution μ such that $\mu(\mathcal{J}) \subseteq K$ and $(I, K) \not\models \Sigma_{12}$. Notice that this implies that $(I, \mu(\mathcal{J})) \not\models \Sigma_{12}$ since, otherwise, we would have that $(I, K) \models \Sigma_{12}$ given that mappings specified by st-tgds are closed-up on the right. Thus, in order to check that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of I, it is enough to check that there exists a substitution μ such that $(I, \mu(\mathcal{J})) \not\models \Sigma_{12}$. Then, we use the substitution μ as a witness and we check $(I, \mu(\mathcal{J})) \not\models \Sigma_{12}$ in polynomial time since Σ_{12} is fixed. We have shown that the problem of checking whether $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of I is in NP, which completes this part of the proof.

Now to prove that the problem is coNP-hard we use a reduction from the DNF-TAUTOLOGY problem. This problem has as input a propositional formula φ of the form $D_1 \vee D_2 \vee \cdots \vee D_k$, each D_i a conjunction of literals, and the question is whether every truth assignment satisfies φ. DNF-TAUTOLOGY is coNP-hard. To see this, notice that given a formula φ in DNF, we can construct in polynomial time a formula ψ in CNF that is equivalent to $\neg\varphi$. front of literals, and then replace formulas of the form $\neg\neg x$ by x with x a propositional variable). Thus, we have that $\varphi \in$ DNF-TAUTOLOGY if and only if $\psi \notin$ CNF-SAT, and CNF-SAT is a well-known NP-complete problem.

Now, consider the fixed schema mapping $\mathcal{M} = (\{R(\cdot)\}, \{S(\cdot)\}, \{R(x) \rightarrow S(x)\})$. Let φ be a propositional formula in DNF with variables $x_1, x_2, \ldots, x_n$. When constructing the instances, we consider $x_1, \ldots, x_n$ as null values. Now, consider a complete instance I with a single fact $R(1)$ and the conditional instance $\mathcal{J}$ with a fact $S(1)$ and an element condition α associated to $S(1)$ such that α is constructed from φ by replacing every propositional variable x in φ by $x = 1$, and every negation of a propositional variable $\neg x$ by $x \neq 1$. For example, if φ is the formula $(x \wedge y \wedge \neg z) \vee (\neg x \wedge y \wedge z)$, then α is the element condition

$$(x = 1 \wedge y = 1 \wedge z \neq 1) \vee (x \neq 1 \wedge y = 1 \wedge z = 1).$$

We now prove that $\varphi \in$ DNF-TAUTOLOGY if and only if $\mathcal{J}$ is an $\mathcal{R}_{\text{cond}}$-solution of I. We prove the contrapositive. Thus, assume first that $\varphi \notin$ DNF-TAUTOLOGY, then there exists a truth assignment σ such that $\sigma \not\models \varphi$. Notice that σ can be seen as a value assignment for the nulls in $\mathcal{J}$. Moreover, it holds that $\sigma(x) = 1$ if and only if $\sigma \models x = 1$, and $\sigma(x) = 0$ is and only if $\sigma \models x \neq 1$. Therefore, σ as an assignment for $\mathcal{J}$ is such that $\sigma \not\models \alpha$. Thus, the instance $\sigma(\mathcal{J})$ is the empty instance and then $(I, \sigma(\mathcal{J})) \not\models \Sigma_{12}$. Since $\sigma(\mathcal{J}) \in \text{rep}(\mathcal{J})$, we have that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of I. Assume now that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of I. Then, there exists an instance $K \in \text{rep}(\mathcal{J})$ such that $(I, K) \not\models \Sigma_{12}$. Given that $(I, K) \not\models \Sigma_{12}$ we have that the fact $S(1)$ does not belong to K. Moreover, from $K \in \text{rep}(\mathcal{J})$, we know that there exists an assignment μ such that $\mu(\mathcal{J}) \subseteq K$. From these two facts plus the construction of $\mathcal{J}$, we obtain that there exists an assignment μ such that $\mu \not\models \alpha$. Now construct a truth assignment for φ as follows. For every variable x in φ, if $\mu(x) = 1$, then $\sigma(x) = 1$, and if $\mu(x) \neq 1$, then $\sigma(x) = 0$. Then, we have that $\sigma(x) = 1$ if and only if $\mu \models x = 1$, and $\sigma(x) = 0$ if and only if $\mu \models x \neq 1$. Thus, since $\mu \not\models \alpha$ we have that $\sigma \not\models \varphi$, which implies that $\varphi \notin$ DNF-TAUTOLOGY. □

We now consider the general problem in which both source and target instances may be general elements in a representation system $(\mathbf{W}, \text{rep})$. In Section 7.1, we also use this formulation when considering the representation system of knowledge bases. The general formulation is as follows. Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{R} = (\mathbf{W}, \text{rep})$ a representation system. We want to 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 $\mathcal{M}$?

The next result implies that for the representation systems $\mathcal{R}_{\text{pos}}$ and $\mathcal{R}_{\text{cond}}$, the complexity of both general problems coincides and is Π_2^P-complete.

THEOREM 5.11. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of st-tgds. Then* CHECKSOLUTION($\mathcal{M}, \mathcal{R}_{cond}$) *and* CHECKSOLUTION($\mathcal{M}, \mathcal{R}_{pos}$) *are both Π_2^P-complete.*[2]

PROOF. We first prove that CHECKSOLUTION($\mathcal{M}, \mathcal{R}_{\text{cond}}$) is in Π_2^P. Given a pair $(\mathcal{I}, \mathcal{J})$ such that both $\mathcal{I}$ and $\mathcal{J}$ are conditional instances, we have that $\mathcal{J}$ is an $\mathcal{R}_{\text{cond}}$-solution of $\mathcal{I}$ if and only if for every $J \in \text{rep}(\mathcal{J})$ there exists an instance $I \in \text{rep}(\mathcal{I})$ such that $(I, J) \models \Sigma_{12}$. We show next that this test can be done in Π_2^P. We first prove that the following property is in NP:

$$\text{Given an instance } L \text{ of schema } \mathbf{S}_2, \text{ check whether there exists an instance } K \in \text{rep}(\mathcal{I}) \text{ such that } (K, L) \models \Sigma_{12}. \tag{3}$$

Notice that $K \in \text{rep}(\mathcal{I})$ if and only if there exists a valuation ν such that $\nu(\mathcal{I}) \subseteq K$. Moreover, if $(K, L) \models \Sigma_{12}$, then $(\nu(\mathcal{I}), L) \models \Sigma_{12}$ given that $\nu(\mathcal{I}) \subseteq K$ and Σ_{12} is a set of st-tgds (which implies that $\mathcal{M}$ is closed-down on the left). Thus, in order to check property (3), we can guess the valuation ν and then, in polynomial time, test whether $(\nu(\mathcal{I}), L) \models \Sigma_{12}$. This proves that (3) is in NP. We now use (3) to show that CHECKSOLUTION($\mathcal{M}, \mathcal{R}_{\text{cond}}$) is in Π_2^P. Thus, assume that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of $\mathcal{I}$. Then, there exists an instance of $\mathbf{S}_2$, say J^*, such that $J^* \in \text{rep}(\mathcal{J})$ and for every

[2]In the conference version of this article [Arenas et al. 2011], we incorrectly claimed that CHECKSOLUTION($\mathcal{M}, \mathcal{R}_{\text{pos}}$) was in NP. We thank Pablo Barceló for pointing out to us that there was an error in the proof of that claim.

$I \in \text{rep}(\mathcal{I})$ it holds that $(I, J^*) \not\models \Sigma_{12}$. Given that $J^* \in \text{rep}(\mathcal{J})$ we know that there exists a valuation μ such that $\mu(\mathcal{J}) \subseteq J^*$. Moreover, if, for some I^*, we have that $(I^*, \mu(\mathcal{J})) \models \Sigma_{12}$, then we would have that $(I^*, J^*) \models \Sigma_{12}$, since Σ_{12} is a set of st-tgds and, consequently, defines a mapping that is closed-up on the right. Thus, we conclude that, for every $I \in \text{rep}(\mathcal{I})$, it holds that $(I, \mu(\mathcal{J})) \not\models \Sigma_{12}$. Thus, to check that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of $\mathcal{I}$, it is enough to guess a valuation ν and then check that, for every $I \in \text{rep}(\mathcal{I})$, it holds that $(I, \nu(\mathcal{J})) \not\models \Sigma_{12}$. We know that this last check can be done with an NP oracle (by property (3)), and then checking that $\mathcal{J}$ is not an $\mathcal{R}_{\text{cond}}$-solution of $\mathcal{I}$ can be done in Σ_2^P. This completes the proof that CHECKSOLUTION($\mathcal{M}$, $\mathcal{R}_{\text{cond}}$) is in Π_2^P.

We now show that CHECKSOLUTION($\mathcal{M}$, $\mathcal{R}_{\text{pos}}$) is Π_2^P-hard. To do this, we use a reduction from the validity problem of quantified propositional formulas of the form $\forall\bar{x}\exists\bar{y}\varphi$, where φ is a propositional formula in 3CNF that mentions variables from $\bar{x}$ and $\bar{y}$.

Consider a schema $\mathbf{S}_1$ with a binary predicate T and four ternary predicate C_1, C_2, C_3, and C_4, and a schema $\mathbf{S}_2$ which is a copy of $\mathbf{S}_1$, that is, $\mathbf{S}_2$ is composed of $\hat{T}$, $\hat{C}_1$, $\hat{C}_2$, $\hat{C}_3$, and $\hat{C}_4$. Consider also the copying setting Σ_{12} composed of formulas $T(x, y) \rightarrow \hat{T}(x, y)$ and $C_i(x, y, z) \rightarrow \hat{C}_i(x, y, z)$ for $i = 1, 2, 3, 4$. Now, let Φ be a formula of the form

$$\forall x_1 \cdots \forall x_k \exists y_1 \cdots \exists y_\ell \; \varphi,$$

where φ a propositional formula in 3CNF. We construct an instance $(\mathcal{I}, \mathcal{J})$ of CHECKSOLUTION($\mathcal{M}$, $\mathcal{R}_{\text{pos}}$) by using variables $x_1, \ldots, x_k$ and $y_1, \ldots, y_\ell$ as null values in $\mathcal{I}$ and $\mathcal{J}$, and values $0, 1, 2, \ldots, k$ as constants.

To encode the satisfaction of a propositional formula in 3CNF, we use predicates C_i's. The idea is that C_1 is used to encode clauses of the form $(u \vee v \vee w)$, C_2 to encode clauses of the form $(u \vee v \vee \neg w)$, C_3 to encode clauses of the form $(u \vee \neg v \vee \neg w)$, and C_4 to encode clauses of the form $(\neg u \vee \neg v \vee \neg w)$, with u, v, and w propositional variables.

To construct instance $\mathcal{J}$, we consider all the possible tuples composed of values 0 and 1 for all predicates $\hat{C}_1$, $\hat{C}_2$, $\hat{C}_3$, and $\hat{C}_4$, except for $\hat{C}_1(0, 0, 0)$, $\hat{C}_2(0, 0, 1)$, $\hat{C}_3(0, 1, 1)$ and $\hat{C}_4(1, 1, 1)$. Moreover, all the tuples added to predicates $\hat{C}_i$ have element-condition $\top$. Notice that the only tuples that do not appear in the $\hat{C}_i$ predicates are those that represent assignments that falsify the clauses. Regarding predicate $\hat{T}$, for every $i \in \{1, \ldots, k\}$, we add the tuple $\hat{T}(i, x_i)$ to $\mathcal{J}$ with element-condition $\top$.

Now, to construct $\mathcal{I}$, we consider the element-condition α given by the expression

$$\alpha : \quad (x_1 = 0 \vee x_1 = 1) \wedge (x_2 = 0 \vee x_2 = 1) \wedge \cdots \wedge (x_k = 0 \vee x_k = 1).$$

Then, for every clause C in φ, we add a tuple with the propositional variables of C to the corresponding C_i predicate in $\mathcal{I}$, every tuple with element-condition α. For example, if $(x_1 \vee y_3 \vee \neg x_4)$ and $(\neg y_2 \vee \neg x_1 \vee \neg x_2)$ are clauses in φ, we add tuples $C_2(x_1, y_3, x_4)$ and $C_4(y_2, x_1, x_2)$ to $\mathcal{I}$, with element condition α. Finally, similarly as for $\mathcal{J}$, for every $i \in \{1, \ldots, k\}$, we add the tuple $T(i, x_i)$ to $\mathcal{I}$ with element-condition $\top$. We show next that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of $\mathcal{I}$ if and only if Φ is valid.

Assume first that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of $\mathcal{I}$. Thus, we have that, for every $K \in \text{rep}(\mathcal{J})$, there must exists a substitution μ such that $(\mu(\mathcal{I}), K) \models \Sigma_{12}$. Let ν be a substitution such that ν assigns value 0 or 1 to the variables $x_1, \ldots, x_k$. Then, since $\nu(\mathcal{J}) \in \text{rep}(\mathcal{J})$, we know that there exists a substitution μ such that $(\mu(\mathcal{I}), \nu(\mathcal{J})) \models \Sigma_{12}$. First, notice that since for every $i \in \{1, \ldots, k\}$ the tuple $T(i, x_i)$ has element-condition $\top$, then $T(i, \mu(x_i))$ belongs to $\mu(\mathcal{I})$. Moreover, since the only tuple in $\nu(\mathcal{J})$ that has the constant value i in the first component of $\hat{T}$ is $\hat{T}(i, \nu(x_i))$ and $(\mu(\mathcal{I}), \nu(\mathcal{J}))$ satisfies the dependency $T(x, y) \rightarrow \hat{T}(x, y)$, we obtain that $\mu(x_i) = \nu(x_i)$ for every $i \in \{1, \ldots, k\}$. Then, we have that $\mu \models \alpha$ and thus, for every tuple $C_i(u, v, w) \in \mathcal{I}$ with $i = 1, 2, 3, 4$, we have that $C_i(\mu(u), \mu(v), \mu(w))$ is in $\mu(\mathcal{I})$. Notice that, in $\nu(\mathcal{J})$, we have all the combinations of values 0 and 1 for all predicates $\hat{C}_1, \hat{C}_2, \hat{C}_3$, and $\hat{C}_4$, except for $\hat{C}_1(0,0,0)$, $\hat{C}_2(0,0,1)$, $\hat{C}_3(0,1,1)$ and $\hat{C}_4(1,1,1)$. Thus, since $(\mu(\mathcal{I}), \nu(\mathcal{J}))$ satisfies the dependency $C_i(x,y,z) \rightarrow \hat{C}_i(x,y,z)$ for $i = 1,2,3,4$, it is easy to see that the valuation σ_μ of variables $x_1,\ldots,x_k$ and $y_1,\ldots,y_\ell$ induced by the substitutions μ is such that $\sigma_\mu \models \varphi$. Thus, we have shown that, for every possible assignment ν of variables $x_1,\ldots,x_k$, there exists an extension μ of ν to variables $y_1,\ldots,y_\ell$ such that $\mu \models \varphi$. This implies that Φ is valid.

Now assume that Φ is valid. We show next that, for every possible substitution ν, there exists a substitution μ such that $(\mu(\mathcal{I}), \nu(\mathcal{J})) \models \Sigma_{12}$. Assume first that substitution ν assigns to some null x_i a value different from 0 and 1. Now consider the substitution μ such that $\mu(x_i) = \nu(x_i)$ for every $i \in \{1,\ldots,k\}$ and such that $\mu(y_i)$ is an arbitrary value for $i \in \{1,\ldots,\ell\}$. Notice that $\mu \not\models \alpha$ and then relations C_1, C_2, C_3, and C_4 are empty in $\mu(\mathcal{I})$. Moreover, relation T in $\mu(\mathcal{I})$ is a copy of relation $\hat{T}$ in $\nu(\mathcal{J})$, from which we conclude that $(\mu(\mathcal{I}), \nu(\mathcal{J})) \models \Sigma_{12}$. Finally, assume that ν is a substitution that assigns to every null $x_1,\ldots,x_k$ a value in $\{0,1\}$. Since Φ is valid, we know that we can find a valuation μ of the variables $y_1,\ldots,y_\ell$ such that $(\nu,\mu) \models \varphi$. Then, consider μ' obtained by extending μ such that $\mu'(x_i) = \nu(x_i)$ for every $i \in \{1,\ldots,k\}$. It is not difficult to see that $(\mu'(\mathcal{I}), \nu(\mathcal{J})) \models \Sigma_{12}$. We have shown that, for every possible substitution ν, there exists a substitution μ such that $(\mu(\mathcal{I}), \nu(\mathcal{J})) \models \Sigma_{12}$. Finally, since st-tgds are closed-up on the right, we obtain that for every possible substitution ν and instance K such that $\nu(\mathcal{J}) \subseteq K$, there exists a substitution μ such that $(\mu(\mathcal{I}), K) \models \Sigma_{12}$. This implies that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution of $\mathcal{I}$, completing the proof. □

It should be noticed that the lower bounds in Theorem 5.11 cannot be directly obtained from the results in Abiteboul et al. [1991], since, in that paper, conditional instances allow *global conditions* that we do not consider in this proof.

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. [2005a], 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 [Bernstein 2003; Bernstein and Melnik 2007]. Two of the most fundamental operations in this area are the *composition* and *inversion* of schema mappings. The problem of composing schema mappings was solved in Fagin et al. [2005b] for the class of mappings specified by st-tgds. More precisely, Fagin et al. [2005b] proposed 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 [Fagin et al. 2005b]. 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 [Fagin 2007; Fagin et al. 2007, 2009; Arenas et al. 2009a, 2009b]. 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 [Bernstein and Melnik 2007]. 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. [2009], where the notion of inversion proposed in Arenas et al. [2009b] 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 considered together with the composition of schema mappings, that is, for the case of SO-tgds. Indeed, SO tgds do not always admit an inverse under the notions of inversion defined in Fagin [2007], Fagin et al. [2007], and Arenas et al. [2009a, 2009b], and it is not clear whether the notion of inversion introduced in Fagin et al. [2009] 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 Arenas et al. [2009b], if positive conditional instances are allowed in source and target schemas. It remains as an open problem to show whether this inverse can always be specified with an SO tgd, or with an extension of this language.

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 [Fagin et al. 2005b]), which 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:

$$\exists f \left(\forall x\, (E(x) \rightarrow R(x, f(x))) \;\wedge\; \forall x\, (E(x) \wedge x = f(x) \rightarrow T(x))\right). \tag{4}$$

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 $\mathbf{S}_2$,[3] 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 Fagin et al. [2005b] 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 $\mathcal{M}$.*

PROOF. Given a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, we know from Fagin et al. [2005b] that there exists a finite sequence of mappings $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ specified by st-tgds, such that $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$. Since $\mathcal{M}$ is fixed, $\mathcal{M}_1, \ldots, \mathcal{M}_k$ are also fixed. Now given a positive conditional instance $\mathcal{I}$, from Theorem 5.1 we know that we can construct in polynomial time a sequence $\mathcal{I}_1, \ldots, \mathcal{I}_k$ such that $\mathcal{I}_1$ is a universal $\mathcal{R}_{\text{pos}}$-solution of $\mathcal{I}$ under $\mathcal{M}_1$ and $\mathcal{I}_i$ is a universal $\mathcal{R}_{\text{pos}}$-solution of $\mathcal{I}_{i-1}$ under $\mathcal{M}_i$ for $i \in \{2, \ldots, k\}$. Therefore, given that $\mathcal{M} = \mathcal{M}_1 \circ \cdots \circ \mathcal{M}_k$, we have that $\mathcal{I}_k$ is a positive conditional instance that is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$. □

[3]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).

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. In fact, the approach used to prove Theorem 6.1 cannot be used to prove similar results within the data exchange setting proposed by Fagin et al. [2005a], 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 before doing this, we extend the notion of schema mapping in this section to include positive conditional instances, a necessary step in the study of these operators.

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 article.

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 $(I_1, I_2) \in \text{PC}(\mathcal{M})$.*

Recall that the notions of closed-down on the left mapping and closed-up on the right mapping were defined in Section 5.3. It is important to notice that every mapping specified by a set of st-tgds satisfies these conditions, as well as every mapping specified by an SO tgd.

PROOF OF PROPOSITION 6.3. Assume that $\mathcal{M}$ is a closed-down on the left and closed-up on the right mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, and let I_1, I_2 be a pair of instances of $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively. Next, we show that $(I_1, I_2) \in \mathcal{M}$ if and only if $(I_1, I_2) \in \text{PC}(\mathcal{M})$.

($\Rightarrow$) Assume that $(I_1, I_2) \in \mathcal{M}$. Given that $\mathcal{M}$ is closed-up on the right, we have that for every instance I_2' of $\mathbf{S}_2$ such that $I_2 \subseteq I_2'$, it holds that $(I_1, I_2') \in \mathcal{M}$. Thus, we conclude by definition of I_2 that $\text{rep}_{\text{pos}}(I_2) \subseteq \text{SOL}_{\mathcal{M}}(I_1)$ and, therefore, $\text{rep}_{\text{pos}}(I_2) \subseteq \text{SOL}_{\mathcal{M}}(\text{rep}_{\text{pos}}(I_1))$ since $I_1 \in \text{rep}_{\text{pos}}(I_1)$. Hence, we have that I_2 is an $\mathcal{R}_{\text{pos}}$-solution of I_1 under $\mathcal{M}$ and, thus, $(I_1, I_2) \in \text{PC}(\mathcal{M})$.

($\Leftarrow$) Assume that $(I_1, I_2) \in \text{PC}(\mathcal{M})$. Then, we have that I_2 is an $\mathcal{R}_{\text{pos}}$-solution of I_1 under $\mathcal{M}$ and, thus, $\text{rep}_{\text{pos}}(I_2) \subseteq \text{SOL}_{\mathcal{M}}(\text{rep}_{\text{pos}}(I_1))$. Given that $I_2 \in \text{rep}_{\text{pos}}(I_2)$, we conclude that $I_2 \in \text{SOL}_{\mathcal{M}}(\text{rep}_{\text{pos}}(I_1))$. Hence, there exists an instance I_1' of $\mathbf{S}_1$ such that $I_1' \in \text{rep}_{\text{pos}}(I_1)$ and $(I_1', I_2) \in \mathcal{M}$. But then by definition of I_1, we have that $I_1 \subseteq I_1'$, from which we conclude that $(I_1, I_2) \in \mathcal{M}$ since $\mathcal{M}$ is closed-down on the left. □

6.3. Composition in the Presence of Positive Conditional Instances

In Fagin et al. [2005b], SO tgds were introduced to deal with the problem of composing schema mappings. In fact, it was proved in Fagin et al. [2005b] 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,* $\text{PC}(\mathcal{M}_{12} \circ \mathcal{M}_{23}) = \text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$.

PROOF. Assume that $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$ and $\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \sigma_{23})$, where σ_{12} y σ_{23} are SO tgds. Next, we show that $\text{PC}(\mathcal{M}_{12} \circ \mathcal{M}_{23}) = \text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$.

($\Leftarrow$) Assume that $(\mathcal{I}_1, \mathcal{I}_3) \in \text{PC}(\mathcal{M}_{12}) \circ \text{PC}(\mathcal{M}_{23})$. Then, there exists a positive conditional instance $\mathcal{I}_2$ of $\mathbf{S}_2$ such that $(\mathcal{I}_1, \mathcal{I}_2) \in \text{PC}(\mathcal{M}_{12})$ and $(\mathcal{I}_2, \mathcal{I}_3) \in \text{PC}(\mathcal{M}_{23})$. Thus, we have that:

$$\text{rep}_{\text{pos}}(\mathcal{I}_2) \subseteq \text{SOL}_{\mathcal{M}_{12}}(\text{rep}_{\text{pos}}(\mathcal{I}_1)), \tag{5}$$

$$\text{rep}_{\text{pos}}(\mathcal{I}_3) \subseteq \text{SOL}_{\mathcal{M}_{13}}(\text{rep}_{\text{pos}}(\mathcal{I}_2)). \tag{6}$$

Next we show that this implies that $\text{rep}_{\text{pos}}(\mathcal{I}_3) \subseteq \text{SOL}_{\mathcal{M}_{12} \circ \mathcal{M}_{23}}(\text{rep}_{\text{pos}}(\mathcal{I}_1))$, from which we conclude that $(\mathcal{I}_1, \mathcal{I}_3) \in \text{PC}(\mathcal{M}_{12} \circ \mathcal{M}_{23})$.

Assume that $I_3 \in \text{rep}_{\text{pos}}(\mathcal{I}_3)$. Then, we have by (6) that there exists $I_2 \in \text{rep}_{\text{pos}}(\mathcal{I}_2)$ such that $(I_2, I_3) \in \mathcal{M}_{23}$. Moreover, given that $I_2 \in \text{rep}_{\text{pos}}(\mathcal{I}_2)$, we conclude by (5) that there exists $I_1 \in \text{rep}_{\text{pos}}(\mathcal{I}_1)$ such that $(I_1, I_2) \in \mathcal{M}_{12}$. Therefore, we have that $I_1 \in \text{rep}_{\text{pos}}(\mathcal{I}_1)$ and $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$, from which we deduce that $I_3 \in \text{SOL}_{\mathcal{M}_{12} \circ \mathcal{M}_{23}}(\text{rep}_{\text{pos}}(\mathcal{I}_1))$. This conclude the proof of the first part of the lemma.

($\Rightarrow$) Assume that $(\mathcal{I}_1, \mathcal{I}_3) \in \text{PC}(\mathcal{M}_{12} \circ \mathcal{M}_{23})$. Then, we have that:

$$\text{rep}_{\text{pos}}(\mathcal{I}_3) \subseteq \text{SOL}_{\mathcal{M}_{12} \circ \mathcal{M}_{23}}(\text{rep}_{\text{pos}}(\mathcal{I}_1)). \tag{7}$$

Given that positive conditional instances form a strong representation system for the class of mappings specified by SO tgds (see Corollary 6.2), we know that there exists a positive conditional instance $\mathcal{I}_2$ of $\mathbf{S}_2$ such that:

$$\text{rep}_{\text{pos}}(\mathcal{I}_2) = \text{Sol}_{\mathcal{M}_{12}}(\text{rep}_{\text{pos}}(\mathcal{I}_1)). \quad (8)$$

Next we show that (7) and (8) imply that $\text{rep}_{\text{pos}}(\mathcal{I}_3) \subseteq \text{Sol}_{\mathcal{M}_{23}}(\text{rep}_{\text{pos}}(\mathcal{I}_2))$, from which we conclude that $(\mathcal{I}_2, \mathcal{I}_3) \in \text{Pc}(\mathcal{M}_{23})$ and, hence, that $(\mathcal{I}_1, \mathcal{I}_3) \in \text{Pc}(\mathcal{M}_{12}) \circ \text{Pc}(\mathcal{M}_{23})$ since $(\mathcal{I}_1, \mathcal{I}_2) \in \text{Pc}(\mathcal{M}_{12})$.

Let $I_3 \in \text{rep}_{\text{pos}}(\mathcal{I}_3)$. Then, we have by (7) that there exists $I_1 \in \text{rep}_{\text{pos}}(\mathcal{I}_1)$ such that $(I_1, I_3) \in \mathcal{M}_{12} \circ \mathcal{M}_{23}$. Thus, we know that there exists an instance I_2 of $\mathbf{S}_2$ such that $(I_1, I_2) \in \mathcal{M}_{12}$ and $(I_2, I_3) \in \mathcal{M}_{23}$. But not only that, we also know that $I_2 \in \text{Sol}_{\mathcal{M}_{12}}(\text{rep}_{\text{pos}}(\mathcal{I}_1))$ since $I_1 \in \text{rep}_{\text{pos}}(\mathcal{I}_1)$. Hence, we have by (8) that $I_2 \in \text{rep}_{\text{pos}}(\mathcal{I}_2)$. We have proved that $(I_2, I_3) \in \mathcal{M}_{23}$ and $I_2 \in \text{rep}_{\text{pos}}(\mathcal{I}_2)$, from which we conclude that $I_3 \in \text{Sol}_{\mathcal{M}_{23}}(\text{rep}_{\text{pos}}(\mathcal{I}_2))$, which was to be shown. This concludes the proof of the lemma. □

From the results in Fagin et al. [2005b] and Lemma 6.4, it is straightforward to prove the following 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 $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ 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 Arenas et al. [2009b]. In that article, 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 Arenas et al. [2009b]. Given that, in general, there may exist many possible recoveries for a given mapping, an order relation on recoveries is introduced in Arenas et al. [2009b] 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 $\mathcal{P}_{12} \circ \mathcal{P}_{21} \subseteq \mathcal{P}_{12} \circ \mathcal{P}'_{21}$. Note that 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 is proved in Arenas et al. [2009b] that every mapping specified by a finite set of st-tgds admits a maximum recovery, in the scenario where only usual instances with complete information are allowed in the source, and only naive instances, not positive conditional instances, are allowed in the target. Universal solutions are essential to prove this result [Arenas et al. 2009b], as they can be used to distinguish between source instances having different spaces of solutions under a mapping given by st-tgds. That is, if $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of st-tgds, and I_1, I_2 are instances of $\mathbf{S}_1$, then it holds that $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$ if and only if I_1, I_2 have the same universal solutions under $\mathcal{M}$. In the case of SO tgds, it has been shown that there exist mappings specified by these dependencies that do not admit maximum recoveries [Arenas et al. 2009a, 2013], in the same scenario as for the case of st-tgds. In the following example, we present the main difficulty encountered when trying to construct a maximum recovery for a mapping specified by a SO tgd, which is the fact that universal solutions cannot be used to distinguish between instances having different spaces of solutions in this case.

Example 6.6. Let $\mathbf{S}_1$ be a schema consisting of unary relations A, B, and $\mathbf{S}_2$ be a schema consisting of a binary relation R and a unary relation S. Moreover, assume that $\mathcal{M}$ is a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ specified by the following SO tgd:

$$\sigma_{12} = \exists f\, (\forall x(A(x) \rightarrow R(x, f(x))) \wedge \forall x(A(x) \wedge x = f(x) \rightarrow S(x)) \wedge \forall x(B(x) \rightarrow R(x, f(x)))).$$

Consider instances $I_1 = \{A(a)\}$ and $I_2 = \{B(a)\}$ of schema $\mathbf{S}_1$, where a is an arbitrary constant from $\mathbf{D}$, and instance $J = \{R(a, a)\}$ of $\mathbf{S}_2$. Under the semantics for SO tgds proposed in Fagin et al. [2005b], we have that J is not a solution for I_1 under $\mathcal{M}$, while J is a solution for I_2 under $\mathcal{M}$. To see why the former holds, assume for the sake of contradiction that (I_1, J) satisfies σ_{12}. Then, there exists an interpretation f^0 of function symbol f such that (I_1, J) satisfies the three conjunct of σ_{12} under this interpretation. Given that $A(a)$ holds in I_1, $J = \{R(a, a)\}$ and (I_1, J) satisfies conjunct $\forall x(A(x) \rightarrow R(x, f(x)))$, we conclude that $f^0(a) = a$. Thus, given that (I_1, J) satisfies conjunct $\forall x(A(x) \wedge x = f(x) \rightarrow S(x))$, we conclude that $S(a)$ holds in J, which leads to a contradiction. On the other hand, under the semantics for SO tgds proposed in Fagin et al. [2005b], instances I_1, I_2 have the same universal solutions. To see why this is the case, notice that instance $J^\star = \{R(a, n)\}$ of $\mathbf{S}_2$, where n is an arbitrary null value from $\mathbf{N}$, is not only a solution but also a universal solution for both I_1 and I_2 under $\mathcal{M}$.

Given that I_1, I_2 have different spaces of solutions under $\mathcal{M}$, one would like to somehow distinguish between them when computing an inverse for $\mathcal{M}$. As pointed out before, for the case of the maximum recovery, universal solutions have been used as the main tool for distinguishing between instances such as I_1 and I_2. However, I_1, I_2 have the same universal solutions in this case, so universal solutions are of no utility and $\mathcal{M}$ does not admit a maximum recovery. □

Formalizations of the argument presented in Example 6.6 can be found in [Arenas et al. 2009a, 2009b, 2013]. In fact, it has been proved in Arenas et al. [2009a, 2013] that there exist mappings specified by SO tgds that admit neither a Fagin-inverse [Fagin 2007] nor a quasi-inverse [Fagin et al. 2007] nor a maximum recovery [Arenas et al. 2009b]. Moreover, it has also been shown that SO tgds do not admit an inverse under the notion of CQ-maximum recovery studied in [Arenas et al., 2009a, 2013]. Thus, given that it is not clear whether the notion of inversion introduced in Fagin et al. [2009] is appropriate for the language of SO tgds, one can conclude that, 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 6.7. *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.*

Before proving the theorem, we show in the following example how the use of positive conditional instances help in solving the difficulty presented in Example 6.6.

Example 6.8. Let $\mathcal{M}$, I_1 and I_2 be defined as in Example 6.6. As pointed out in Section 6.2, instances I_1 and I_2 can be represented as the following positive conditional instances:

$$\begin{array}{lll} \mathcal{I}_1 : A & (a) & \top \\ \mathcal{I}_2 : B & (a) & \top. \end{array}$$

Then, by using exactly the same argument as in Example 6.6, one can prove that $\mathcal{I}_1$, $\mathcal{I}_2$ have different spaces of $\mathcal{R}_{\text{pos}}$-solutions under $\text{Pc}(\mathcal{M})$. However, as opposed to Example 6.6, the spaces of universal $\mathcal{R}_{\text{pos}}$-solutions for $\mathcal{I}_1$ and $\mathcal{I}_2$ under $\text{Pc}(\mathcal{M})$ are different. On the one side, positive conditional instance:

$$\mathcal{J}^\star : R(a, n) \quad \top,$$

which represents instance $J^\star$ mentioned in Example 6.6, is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}_2$ under $\text{Pc}(\mathcal{M})$, but it is not even an $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}_1$ under $\text{Pc}(\mathcal{M})$. To see why this is the case, notice that if $J = \{R(a, a)\}$, then $J \in \text{rep}_{\text{pos}}(\mathcal{J}^\star)$ and $J \notin \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I}_1))$. On the other side, the following positive conditional instance is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}_1$ under $\text{Pc}(\mathcal{M})$, which is not a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}_2$ under $\text{Pc}(\mathcal{M})$:

$$\begin{array}{ll} R(a, n) & \top \\ S(a) & n = a. \end{array}$$

Thus, in this case, universal $\mathcal{R}_{\text{pos}}$-solutions can be used to distinguish between instances $\mathcal{I}_1$ and $\mathcal{I}_2$. And, more generally, in this case universal $\mathcal{R}_{\text{pos}}$-solutions can be used to construct a maximum recovery for $\mathcal{M}$, as shown next. □

To prove Theorem 6.7, we need to extend some of the results in Arenas et al. [2009b] to the case of Pc-mappings. Let $\mathcal{P}$ be a Pc-mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$ and $\mathcal{I}$ a positive conditional instance of $\mathbf{S}_1$. Then $\text{Sol}_{\mathcal{P}}(\mathcal{I})$ is defined as the set $\{\mathcal{J} \mid (\mathcal{I}, \mathcal{J}) \in \mathcal{P}\}$. A positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$ is said to be a *witness for* $\mathcal{I}$ *under* $\mathcal{P}$ if for every positive conditional instance $\mathcal{I}'$ of $\mathbf{S}_1$, if $\mathcal{J} \in \text{Sol}_{\mathcal{P}}(\mathcal{I}')$, then $\text{Sol}_{\mathcal{P}}(\mathcal{I}) \subseteq \text{Sol}_{\mathcal{P}}(\mathcal{I}')$. Moreover, a positive conditional instance $\mathcal{J}$ is said to a witness solution for a positive conditional instance $\mathcal{I}$ under a Pc-mapping $\mathcal{P}$ if $\mathcal{J}$ is witness for $\mathcal{I}$ under $\mathcal{P}$ and $\mathcal{J} \in \text{Sol}_{\mathcal{P}}(\mathcal{I})$. Next, we use the notion of witness solution to provide a necessary and sufficient condition for the existence of a maximum recovery for a Pc-mapping.

LEMMA 6.9. *Let $\mathcal{P}$ be a* Pc*-mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$. Then, $\mathcal{P}$ has a maximum recovery iff for every positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$, there exists a witness solution for $\mathcal{I}$ under $\mathcal{P}$.*

The proof of Lemma 6.9 is in the Appendix. We now prove Theorem 6.7.

PROOF OF THEOREM 6.7. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \sigma_{12})$, where σ_{12} is an SO tgd. Next, we use Lemma 6.9 to prove that $\text{Pc}(\mathcal{M})$ admits a maximum recovery. Let $\mathcal{I}$ be a positive conditional instance of $\mathbf{S}_1$. We know from Corollary 6.2 that there exists a positive conditional instance $\mathcal{J}$ of $\mathbf{S}_2$ such that $\mathcal{J}$ is a universal $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{J}$ under $\mathcal{M}$, that is,

$$\text{rep}_{\text{pos}}(\mathcal{J}) = \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I})). \tag{9}$$

Next, we show that $\mathcal{J}$ is a witness for $\mathcal{I}$ under $\text{Pc}(\mathcal{M})$. Assume that $\mathcal{J} \in \text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}')$, where $\mathcal{I}'$ is a positive conditional instance of $\mathbf{S}_1$. Then we have that $\mathcal{J}$ is an $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}'$ under $\mathcal{M}$, that is,

$$\text{rep}_{\text{pos}}(\mathcal{J}) \subseteq \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I}')). \tag{10}$$

From (9) and (10), we conclude that:

$$\text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I})) \subseteq \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I}')). \tag{11}$$

If $\mathcal{J}' \in \text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I})$, then $\mathcal{J}'$ is an $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}$ under $\mathcal{M}$, that is, $\text{rep}_{\text{pos}}(\mathcal{J}') \subseteq \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I}))$. Thus, we have from (11) that $\text{rep}_{\text{pos}}(\mathcal{J}') \subseteq \text{Sol}_{\mathcal{M}}(\text{rep}_{\text{pos}}(\mathcal{I}'))$ and, hence, $\mathcal{J}'$ is an $\mathcal{R}_{\text{pos}}$-solution for $\mathcal{I}'$ under $\mathcal{M}$. Hence, we conclude that $\text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}) \subseteq \text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}')$.

We conclude from these paragraphs that for every positive conditional instance $\mathcal{I}'$ of $\mathbf{S}_1$ such that $\mathcal{J} \in \text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}')$, it holds that $\text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}) \subseteq \text{Sol}_{\text{Pc}(\mathcal{M})}(\mathcal{I}')$. Thus, we have that $\mathcal{J}$ is a witness for $\mathcal{I}$ under $\text{Pc}(\mathcal{M})$.

We have shown that every positive conditional instance $\mathcal{I}$ of $\mathbf{S}_1$ has a witness solution under $\text{Pc}(\mathcal{M})$, which implies by Lemma 6.9 that $\text{Pc}(\mathcal{M})$ admits a maximum recovery. □

7. KNOWLEDGE BASES

In this section, we apply our framework for representation systems to the case of *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*).

Next, we formalize the notion of knowledge base used in this article, 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 (J, Σ_2) a *knowledge-base solution* of (I, Σ_1) under $\mathcal{M}$.

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 $\mathbf{S}_1$:

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

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 $\mathbf{S}_2 = \{F'(\cdot, \cdot), GP'(\cdot, \cdot)\}$ by using the following set Σ_{12} of st-tgds that copies F into F' and GP into GP'

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

and let $\mathcal{M}$ be the mapping specified by Σ_{12}. According to our definition for exchanging knowledge-bases, it can be proved that for (I, Σ_1) described previously, the knowledge base (J, Σ_2) with $J = \{F'(a, b), F'(b, d), GP'(c, d)\}$ and Σ_2 given by

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

is a knowledge-base solution of (I, Σ_1), as $\text{Mod}(J, \Sigma_2) \subseteq \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$. Notice that, even though mapping $\mathcal{M}$ states that all data from GP needs to be copied to GP', the knowledge-base solution is not storing all this information explicitly. For example, the fact $GP'(a, d)$ is not in J. Instead, the knowledge-base is taking advantage of the explicit data stored in F' and the implicit knowledge in Σ_2. In particular, the facts $F'(a, b)$ and $F'(b, d)$ in J, together with the rule $F'(x, y) \wedge F'(y, z) \rightarrow GP'(x, z)$ in Σ_2, ensure that every model in $\text{Mod}(J, \Sigma_2)$ will contain the fact $GP'(a, d)$. On the other hand, $GP'(c, d)$ has to be explicitly included in J, since it comes from the atom $GP(c, d)$ that is inferred from predicate M in $\mathbf{S}_1$, for which we don't have any information in $\mathbf{S}_2$.

Another possible knowledge-base solution for (I, Σ_1) under $\mathcal{M}$ is $(J', \emptyset)$ with $J' = \{F'(a, b), F'(b, d), GP'(a, d), GP'(c, d)\}$. In this case, no implicit knowledge is needed as all data is explicitly included in J'. $\square$

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. The example also shows that in general one would have many possibilities when deciding what to make explicit and what to keep implicit when exchanging knowledge bases. Thus, one of the main questions that need to be answered in the knowledge-base context is what is a good knowledge-base solution. We study this fundamental question, as well as the problem of actually materializing a solution, in Section 8.

Besides this question, many algorithmic problems arise in the context of knowledge bases and schema mappings. Here, 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 $\mathcal{M}$. But first, we introduce some notation that will be extensively used in the rest of this article.

Chasing Complete Instances with Full tgds. Recall that, in Section 4.2, we introduced a special chase procedure for conditional instances. Instead of defining the chase for complete instances as a particular case of the chase introduced in Section 4.2, we define here the procedure for the specific case of complete instances and full tgds [Beeri and Vardi 1984], as it is considerably simpler.

Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be disjoint schemas, and 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 instance of $\mathbf{S}_1$ obtained by the following process. Begin with instance I. Then, for every full tgd of the form

$$\varphi(\bar{x}) \rightarrow R(\bar{x})$$

in Σ_1 and every tuple $\bar{a}$ of values in $\text{dom}(I)$, if $I \models \varphi(\bar{a})$ and $R(\bar{a})$ is not a fact in I, then add $R(\bar{a})$ to I, and repeat this process until no new fact can be added to I. This procedure always terminates, and runs in polynomial time if the set Σ_1 is fixed [Beeri and Vardi 1984]. 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 $\mathbf{S}_1 \cup \mathbf{S}_2$. Whenever Σ_{12} is a set of full st-tgds, we denote by $\text{chase}_{\Sigma_{12}}(I)$ the resulting instance $J^\star$ (which is the standard notation in the data exchange context [Fagin et al. 2005a]). 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 decision 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 *tgd knowledge bases* and *full-tgd 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 for knowledge bases using (non-full) tgds, and for a fixed schema mapping $\mathcal{M}$ specified by a single copy st-tgd, that is, a full st-tgd of the form $S(x_1, \dots, x_k) \to T(x_1, \dots, x_k)$, where $x_1, \dots, x_k$ are pairwise distinct variables.

THEOREM 7.2. *There exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, with Σ_{12} consisting of a single copy st-tgd, for which the problem* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{tgd}})$ *is undecidable.*

PROOF. We use a reduction from the embedding problem for finite semigroups [Kolaitis et al. 2006]. Consider a pair $\mathbf{A} = (A, g)$ where A is a finite set and $g : A \times A \to A$ is a partial associative function. Then, $\mathbf{A}$ is embeddable in a finite semigroup if there exists $\mathbf{B} = (B, f)$ such that $A \subseteq B$ and $f : B \times B \to B$ is a total associative function that extends g. The following problem is known to be undecidable [Kolaitis et al. 2006]: given an arbitrary $\mathbf{A} = (A, g)$, checking whether it is embeddable in a finite semigroup.

We first define a fixed mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ such that $\mathbf{S}_1 = \{C(\cdot,\cdot), E(\cdot,\cdot), N(\cdot,\cdot), G(\cdot,\cdot,\cdot), F(\cdot)\}$, $\mathbf{S}_2 = \{R(\cdot)\}$, and $\Sigma_{12} = \{\forall x (F(x) \to R(x))\}$. Now let $\mathbf{A} = (A, g)$ with $A = \{a_1, \dots, a_n\}$ a finite set and $g : A \times A \to A$ a partial associative function over A. We explain now how we construct an instance of CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{tgd}})$ from $\mathbf{A}$.

We start by considering an instance I such that $C^I = \{(i, a_i) \mid i \in \{1, \dots, n\}\}$, $E^I = \{(a_i, a_i) \mid i \in \{1, \dots, n\}\}$, $N^I = \{(i, j) \mid i \neq j,\ i, j \in \{1, \dots, n\}\}$, $G^I = \{(a_i, a_j, a_k) \mid g(a_i, a_j) = a_k\}$ and $F^I = \emptyset$. We construct now a set Σ_1 of tgds such that there exists an instance $K \in \text{MOD}(I, \Sigma)$ with $F^K = \emptyset$ if and only if $\mathbf{A}$ is embeddable in a finite semigroup. In Σ_1, we use formulas ensuring that E is interpreted as an equivalence relation over the elements mentioned in G. That is, we include the following dependencies:

$$\text{dom}_G(x) \to E(x, x) \tag{12}$$

$$E(x, y) \to E(y, x) \tag{13}$$

$$E(x, y) \wedge E(y, z) \to E(x, z), \tag{14}$$

where $\text{dom}_G(x)$ is defined as $\exists u \exists v (G(x, u, v) \vee G(u, x, v) \vee G(u, v, x))$. Notice that although (12) is not syntactically a tgd (it has disjunctions in its premise), it is equivalent to a set of tgds. We also ensure that this equivalence relation is consistent with the initial elements in the set A. We do this by using predicate C and predicate N with the following tgd:

$$E(x, y) \wedge C(u, x) \wedge C(v, y) \wedge N(u, v) \ \rightarrow\ \exists w\ F(w). \tag{15}$$

This tgd essentially states that if x and y are different values of A and the equivalence relation assigns both values to the same equivalence class, then relation F cannot be empty. Before continuing with the construction of Σ_1, we note that for every instance K that contains I, satisfies the tgds (12)–(15), and interprets F as the emptyset ($F^K = \emptyset$), it holds that E^K contains an equivalence relation over the elements mentioned in G^K such that every element in A belongs to a different equivalence class.

Now we include a tgd in Σ_1 that ensures that G represents a function that is consistent with the equivalence classes:

$$G(x, y, z) \wedge E(x, x') \wedge E(y, y') \wedge E(z, z') \ \rightarrow\ G(x', y', z') \tag{16}$$

$$G(x, y, z) \wedge G(x', y', z') \wedge E(x, x') \wedge E(y, y') \ \rightarrow\ E(z, z') \tag{17}$$

Finally, we need to include a tgd that ensures that G is total and associative:

$$\text{dom}_G(x) \wedge \text{dom}_G(y) \ \rightarrow\ \exists z\ G(x, y, z) \tag{18}$$

$$G(x, y, u) \wedge G(u, z, v) \wedge G(y, z, w) \ \rightarrow\ G(x, w, v). \tag{19}$$

Set Σ_1 consists of tgds (12)–(19).

Claim 7.3. **A** *is embeddable in a finite semigroup if and only if there exists an instance* $K \in \text{Mod}(I, \Sigma_1)$ *such that* $F^K = \emptyset$.

Proof

($\Rightarrow$) Assume that **A** is embeddable in a finite semigroup, and let $\mathbf{B} = (B, f)$ be a finite semigroup such that $A \subseteq B$ and $f : B \times B \rightarrow B$ is a total and associative function that extends g. Then, let K be an instance of $\mathbf{S}_1$ such that E^K is the identity over B, G^K is exactly the set of triples $(a, b, f(a, b))$, and all the other relations are interpreted as in I. It is easy to see that $I \subseteq K$, $K \models \Sigma_1$ and $F^K = \emptyset$ (given that $F^I = \emptyset$).

($\Leftarrow$) Assume that $I \subseteq K$, $K \models \Sigma_1$ and $F^K = \emptyset$. Then, we can construct a finite semigroup $\mathbf{B} = (B, f)$ embedding **A** as follows. Set B is defined by choosing a representative of every equivalence class induced by E^K, with the additional restriction that for every $a \in A$, element a is chosen as the representative of its class. Moreover, function f over $B \times B$ is defined as: $f(a, b) = c$ if and only if $(a, b, c) \in G^K$. Then, by the properties discussed previously, we have that $A \subseteq B$ and function f is total and associative, which was to be shown. □

To complete the reduction, consider an instance J of $\mathbf{S}_2$ such that $R^J = \emptyset$, and let $\Sigma_2 = \emptyset$. Next, we prove that **A** is embeddable in a finite semigroup if and only if (J, Σ_2) is a knowledge base solution of (I, Σ_1) under $\mathcal{M}$. We first notice that if **A** is embeddable in a finite semigroup, then, by Claim 7.3, we have that there exists an instance $K \in \text{Mod}(I, \Sigma_1)$ such that $F^K = \emptyset$. Then, by definition of Σ_{12}, we conclude that $J \in \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$, which implies that $\text{Mod}(J, \Sigma_1) \subseteq \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$ by definition of J and Σ_2. Thus, we have that (J, Σ_2) is a knowledge base solution of (I, Σ_1) under $\mathcal{M}$. For the opposite direction, assume that (J, Σ_2) is a knowledge base solution of (I, Σ_1) under $\mathcal{M}$. Then, given that $J \in \text{Mod}(J, \Sigma_2)$, we have that $J \in \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$. Hence, there exists $K \in \text{Mod}(I, \Sigma_1)$ such that (K, J) satisfies Σ_{12}. But then, given that $R^J = \emptyset$, we have by definition of Σ_{12} that $F^K = \emptyset$, which implies, by Claim 7.3, that **A** is embeddable in a finite semigroup. This concludes the proof of the theorem.

Notice that the preceding undecidability result holds even if the dependencies of the knowledge bases are assumed to be fixed. Thus, Theorem 7.2 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. More precisely, we start by showing that this problem is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, 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 [Wagner 1987].

THEOREM 7.4. *For every mapping $\mathcal{M}$ specified by a finite set of st-tgds,* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ *is in* $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. *Moreover, there exists a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of full st-tgds, such that* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ *is* $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$*-complete.*

Before proving Theorem 7.4, we prove a lemma that characterizes knowledge base solutions in terms of the chase procedure.

LEMMA 7.5. *Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a finite set of st-tgds, and (I, Σ_1), (J, Σ_2) be knowledge bases over $\mathbf{S}_1$ and $\mathbf{S}_2$, respectively, where Σ_1 and Σ_2 are finite sets of full tgds. Then, (J, Σ_2) is a knowledge-base solution of (I, Σ_1) iff* $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$.

PROOF. First, notice that since Σ_1 is a set of full-tgds, then $\text{chase}_{\Sigma_1}(I) \in \text{MOD}(I, \Sigma_1)$. Moreover, for every $K \in \text{MOD}(I, \Sigma_1)$, it holds that $\text{chase}_{\Sigma_1}(I) \subseteq K$. Similarly, $\text{chase}_{\Sigma_2}(J) \in \text{MOD}(J, \Sigma_2)$ and for every $L \in \text{MOD}(J, \Sigma_2)$ we have that $\text{chase}_{\Sigma_2}(J) \subseteq L$.

Now, to prove one of the directions of the lemma, assume that $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$. We have to prove that for every $L \in \text{MOD}(J, \Sigma_2)$ there exists a $K \in \text{MOD}(I, \Sigma_1)$ such that $(K, L) \models \Sigma_{12}$. Now, let $L \in \text{MOD}(J, \Sigma_2)$. We know that $\text{chase}_{\Sigma_2}(J) \subseteq L$ and thus, since $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ and Σ_{12} is a set of st-tgds, we have that $(\text{chase}_{\Sigma_1}(I), L) \models \Sigma_{12}$. We have shown that, for every $L \in \text{MOD}(J, \Sigma_2)$, there exists a $K \in \text{MOD}(I, \Sigma_1)$ such that $(K, L) \models \Sigma_{12}$, and thus, (J, Σ_2) is a knowledge-base solution of (I, Σ_1).

Towards the opposite direction, assume that (J, Σ_2) is a knowledge-base solution of (I, Σ_1) under the exchange setting $(\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$. Since $\text{chase}_{\Sigma_2}(J) \in \text{MOD}(J, \Sigma_2)$ we know that there exists an instance $K \in \text{MOD}(I, \Sigma_1)$ such that $(K, \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$. Thus, since $\text{chase}_{\Sigma_1}(I) \subseteq K$ and Σ_{12} is a set of st-tgds, we conclude that $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ which was to be shown. □

We now proceed to the proof of Theorem 7.4.

PROOF OF THEOREM 7.4. To show the membership in $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ we use a characterization of this class proved by Wagner [1990] and Buss and Hay [1991]. Let $\mathrm{P}^{\|\mathrm{NP}}$ be the class of problems that can be accepted in polynomial time by a deterministic Turing machine that can make a polynomial number of *parallel queries* to an NP oracle. More specifically, for an input w the machine first computes a polynomial number of inputs for the NP oracle and then, after receiving all the answers, the machine decides in polynomial time whether w belongs to the language (see Wagner [1990]). The important feature here is that no query to the oracle depend on the answer to another query. It was proved in Wagner [1990] and Buss and Hay [1991] that $\mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}^{\|\mathrm{NP}}$. Thus, it is enough to prove the membership of CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ in $\mathrm{P}^{\|\mathrm{NP}}$.

We begin our proof with a technical claim. Let $\mathbf{S}$ be a fixed schema and R an n-ary relational symbol in $\mathbf{S}$ (notice that since $\mathbf{S}$ is fixed, then n is also fixed). We claim that given an instance I, a set Σ of full tgds over $\mathbf{S}$, and an n-ary tuple $\bar{a}$, testing whether $\text{chase}_{\Sigma}(I) \models R(\bar{a})$ is in NP. To establish the membership in NP, we can use as witness the complete sequence of chase steps that produce the atom $R(\bar{a})$. Every chase step is composed of a full tgd $\forall \bar{x}(\exists \bar{y} \varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x})) \in \Sigma$, and a homomorphism h from the atoms in $\varphi(\bar{x}, \bar{y})$ to the instance partially computed until that step in a chase of I. It can be checked in polynomial time that every such step is a valid chase step. We argue now that the witness is of polynomial size in the size of I and Σ. Let $k_{\mathbf{S}}$ be the sum of the arities of the relational symbols in $\mathbf{S}$. Notice that since the schema $\mathbf{S}$ is fixed, $k_{\mathbf{S}}$ is also fixed. Moreover, the number of facts in $\text{chase}_\Sigma(I)$ is bounded by $|\operatorname{dom}(I)|^{k_{\mathbf{S}}}$, every fact of size at most $k_{\mathbf{S}}$, and since every chase step generates at least one new fact, we conclude that the number of chase steps is polynomial in the size of I. Finally, since the witness for every step is of size polynomial in the size of I and Σ, we conclude that the complete witness is of size polynomial in the size of I and Σ.

Now let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ be a fixed mapping with Σ_{12} a set of st-tgds. We need to prove that the problem CheckSolution($\mathcal{M}, \mathcal{K}_{\text{full-tgd}}$) is in $\text{P}^{\|\text{NP}}$. Consider a Turing machine M with an NP oracle that, given a knowledge base (I, Σ_1) over $\mathbf{S}_1$ and a knowledge base (J, Σ_2) over $\mathbf{S}_2$, checks whether (J, Σ_2) is a knowledge base solution of (I, Σ_1) under $\mathcal{M}$ as follows. For every n-ary relational symbol R in $\mathbf{S}_1$ and every n-ary tuple $\bar{a}$ of elements in $\operatorname{dom}(I)$, the machine uses the NP oracle to test whether $\text{chase}_{\Sigma_1}(I) \models R(\bar{a})$. After M has obtained all the answers from the oracle, it has a complete picture of what relational atoms are satisfied by $\text{chase}_{\Sigma_1}(I)$ and, thus, it can effectively construct $\text{chase}_{\Sigma_1}(I)$. Notice that since $\mathbf{S}_1$ is fixed, the machine needs only a polynomial number of queries to the NP oracle to construct $\text{chase}_{\Sigma_1}(I)$. Also notice that all these queries can be made *in parallel* since none of them depend on the answer of another query. Similarly, the machine makes a polynomial number of parallel queries to the oracle to construct $\text{chase}_{\Sigma_2}(J)$. The machine then tests whether $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ which, by Lemma 7.5, is enough to test whether (J, Σ_2) is a knowledge base solution of (I, Σ_1) under $(\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$. Notice that since Σ_{12} is fixed, testing $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ can be done in polynomial time with respect to the size of $\text{chase}_{\Sigma_1}(I)$ and $\text{chase}_{\Sigma_2}(J)$. Moreover, since $\mathbf{S}_1$ and $\mathbf{S}_2$ are fixed, $\text{chase}_{\Sigma_1}(I)$ is of size polynomial in the size of I and Σ_1, and $\text{chase}_{\Sigma_2}(J)$ is of size polynomial in the size of J and Σ_2. Thus, all the process can be done in polynomial time by using a polynomial number of parallel queries to an NP oracle. This completes the proof of membership to $\text{P}^{\|\text{NP}}$.

We prove now that CheckSolution($\mathcal{M}, \mathcal{K}_{\text{full-tgd}}$) is $\text{P}^{\text{NP}[O(\log n)]}$-hard for some fixed mapping $\mathcal{M}$, by reducing MaxOddClique to our problem. An instance of MaxOddClique is an undirected graph G and the question is whether the size of the maximum clique in G is an odd number. It follows from the results by Wagner [1987] that MaxOddClique is $\text{P}^{\text{NP}[O(\log n)]}$-hard.

We describe first the fixed schema mapping that we use in the reduction. The schema $\mathbf{S}_1$ is composed of the relations $Succ(\cdot, \cdot)$, $First(\cdot)$, $Clique(\cdot)$, and $E(\cdot, \cdot)$. Similarly, $\mathbf{S}_2$ is composed of $Succ'(\cdot, \cdot)$, $First'(\cdot)$, $Clique'(\cdot)$, and $E'(\cdot, \cdot)$. The set Σ_{12} contains a single full tgd:

$$\forall x \forall y \; \big(Clique(x) \wedge Succ(x, y) \; \rightarrow \; Clique'(y)\big). \tag{20}$$

Before continuing with the reduction, let us explain the intuition of the interpretation of each relation symbol and the intended meaning of the tgd in Σ_{12}. In both schemas, relation E is used to store a copy of a graph G, and relation $Succ$ is used to store a successor relation. Then, we use a set of dependencies Σ_1 over $\mathbf{S}_1$ to ensure that if G has a clique of even size ℓ, then every interpretation of the knowledge base over $\mathbf{S}_1$ satisfy the atom $Clique(\ell)$. Similarly, we use a set Σ_2 over $\mathbf{S}_2$ to ensure that if G has a clique of odd size m, then every interpretation of the knowledge base over $\mathbf{S}_2$ satisfy the atom $Clique(m)$. Finally, the full st-tgd in Σ_{12} is used to check that if there is a clique of even size ℓ in G, then there must exists a clique of size $\ell + 1$, thus ensuring that the maximum size of a clique in G is an odd number. In what follows, we formalize this intuition.

Given a graph G with n nodes, we construct an input to CheckSolution($\mathcal{M}$, $\mathcal{K}_{\text{full-tgd}}$) as follows. First, instance I is such that $(a, b) \in E^I$ if and only if (a, b) is an edge in G. Moreover, we include in $Succ^I$ a successor relation over $\{1, \ldots, n+1\}$, that is, $Succ^I = \{(i, i+1) \mid i \in \{1, \ldots, n\}\}$, and $First^I = \{1\}$. Finally, we set $Clique^I$ to be the empty set. Similarly, for instance J, we have that $E'^J = \{(a, b) \mid (a, b)$ is an edge in $G\}$, $Succ'^J = \{(i, i+1) \mid i \in \{1, \ldots, n\}\}$, $First'^J = \{1\}$. In the case of J, we have that $Clique'^J = \{1\}$. Now, to describe the sets Σ_1 and Σ_2 we need to define some preliminary formulas. Let $\delta_k(x)$ for $k \in \{1, \ldots, n\}$ be a formula over $\mathbf{S}_1$ defined inductively as follows:

$$\begin{aligned} \delta_1(x) &:= First(x) \\ \delta_k(x) &:= \exists y\big(\delta_{k-1}(y) \wedge Succ(y, x)\big). \end{aligned}$$

That is, $\delta_k(x)$ states that x is the element at position k in the successor relation. Let κ_m for $m \in \{2, \ldots, n\}$ be the following sentence over $\mathbf{S}_1$:

$$\exists x_1 \cdots \exists x_m \bigwedge_{\substack{i,j \in \{1,\ldots,m\} \\ i \neq j}} E(x_i, x_j).$$

That is, κ_m states that there is a clique of size m in the graph given by relation E. Similarly, we construct formulas $\delta'_k(x)$ and κ'_m over $\mathbf{S}_2$ by replacing *First*, *Succ*, and E by *First'*, *Succ'*, and *E'*, respectively. Now, by using the previous formulas, we construct Σ_1 and Σ_2 as follows. For every even number $\ell \in \{2, \ldots, n\}$, we include the sentence:

$$\forall x\big(\big(\kappa_\ell \wedge \delta_\ell(x)\big) \rightarrow Clique(x)\big) \tag{21}$$

in Σ_1. Similarly, for every odd number $m \in \{2, \ldots, n\}$, we include in Σ_2 the sentence:

$$\forall x\big(\big(\kappa'_m \wedge \delta'_m(x)\big) \rightarrow Clique'(x)\big). \tag{22}$$

We show now that (J, Σ_2) is a knowledge base solution of (I, Σ_1) under ($\mathbf{S}_1$, $\mathbf{S}_2$, Σ_{12}) if and only if the size of the maximum clique in G is odd.

First, notice that by the construction of I and Σ_1, we have that for every even number $\ell \in \{2, \ldots, n\}$, it holds that $\text{chase}_{\Sigma_1}(I) \models Clique(\ell)$ if and only if G has a clique of size ℓ. Moreover, by the construction of J and Σ_2 we have that for every odd number $m \in \{1, 2, \ldots, n\}$, it holds that $\text{chase}_{\Sigma_2}(J) \models$ *Clique'*(m) if and only if G has a clique of size m. We use these properties in the rest of the proof.

Assume that the maximum size of a clique in G is an odd number m. We show that ($\text{chase}_{\Sigma_1}(I)$, $\text{chase}_{\Sigma_2}(J)$) satisfies Σ_{12}, which, by Lemma 7.5, shows that (J, Σ_2) is a knowledge-base solution of (I, Σ_1). Suppose that $\text{chase}_{\Sigma_1}(I) \models Clique(i) \wedge Succ(i, j)$ for some i, j. Then, by the discussion in the previous paragraph and since the maximum size of a clique in G is the odd number m, we know that i is an even number such that $i < m$. Moreover, by the interpretation of *Succ*, we have that $j = i + 1$. Then, j is an odd number such that $j \leq m$, which implies that $\text{chase}_{\Sigma_2}(J) \models Clique(j)$. This shows that ($\text{chase}_{\Sigma_1}(I)$, $\text{chase}_{\Sigma_2}(J)$) $\models \Sigma_{12}$.

Assume now that the size of the maximum clique in G is an even number ℓ. Then, we know that $\text{chase}_{\Sigma_1}(I) \models Clique(\ell) \wedge Succ(\ell, \ell+1)$. Now, since ℓ is the maximum size of a clique in G, we know that $\text{chase}_{\Sigma_2}(J) \not\models Clique'(\ell+1)$. Thus, ($\text{chase}_{\Sigma_1}(I)$, $\text{chase}_{\Sigma_2}(J)$) $\not\models$ Σ_{12}, and by Lemma 7.5, we have that (J, Σ_2) is not a knowledge base solution of (I, Σ_1) under $\mathcal{M}$. This completes the reduction. □

We continue our study 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, shedding light on how this complexity depends on these parameters. More precisely, we assume in the former case 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 but considering Σ_2 fixed.

THEOREM 7.6. *Let* $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, *where* Σ_{12} *is a set of st-tgds. Then,* CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$: (1) *can be solved in polynomial time if both source implicit knowledge and target implicit knowledge are fixed,* (2) *is NP-complete if source implicit knowledge is fixed, and* (3) *is coNP-complete if target implicit knowledge is fixed.*

PROOF. Property (1) of the theorem is a direct consequence of the fact that the chase for fixed sets of full tgds is polynomial.

Now we prove (2). By Lemma 7.5, we know that (J, Σ_2) is a knowledge base solution of (I, Σ_1) if and only if $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$. Thus, to check that (J, Σ_2) is a knowledge base solution of (I, Σ_1), we can proceed as follows. We precompute $\text{chase}_{\Sigma_1}(I)$ (which can be done in polynomial time since Σ_1 is fixed). Then, for every formula of the form $\forall \bar{x}(\exists \bar{y}\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\psi(\bar{x}, \bar{z}))$ in Σ_{12}, and for every pair of tuples $\bar{a}$ and $\bar{b}$ of elements such that $\text{chase}_{\Sigma_1}(I)$ satisfies $\varphi(\bar{a}, \bar{b})$, we guess a tuple $\bar{c}$ and a chase sequence that from J and Σ_2 produces the atoms in $\psi(\bar{a}, \bar{c})$. Given that $(\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ is fixed, the complete witness is of size polynomial in the size of I, J and Σ_2, and it can be used to check in polynomial time that $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$, thus proving the membership in NP.

To prove that the problem is NP-hard, we use a reduction from the 3-COLORABILITY problem. Consider the schemas $\mathbf{S}_1 = \{R(\cdot)\}$, and $\mathbf{S}_2 = \{P_1(\cdot), P_2(\cdot), E(\cdot, \cdot)\}$, the set of formulas $\Sigma_{12} = \{\forall x(R(x) \rightarrow P_2(x))\}$, and let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$. Given a graph G, we construct an instance of CHECKSOLUTION$(\mathcal{M}, \mathcal{K}_{\text{full-tgd}})$ as follows. Consider the instance I of $\mathbf{S}_1$ such that $R^I = \{c\}$, and $\Sigma_1 = \emptyset$. Notice that Σ_1 is fixed. Now, instance J is such that $E^J = \{(i, j) \mid i, j \in \{1, 2, 3\} \text{ and } i \neq j\}$, $P_1^J = \{c\}$, and $P_2^J = \emptyset$. To construct Σ_2 we do the following. Assume that G has n nodes, and let f be a one-to-one function from the nodes of G to the set of variables $\{x_1, \dots, x_n\}$. Then, we consider the sentence φ_G defined by:

$$\exists x_1 \cdots \exists x_n \bigwedge_{(a,b) \text{ edge in } G} E(f(a), f(b)).$$

That is, φ_G is a sentence that describes the edges in G. It is straightforward to see that $J \models \varphi_G$ if and only if G is 3-colorable. Now, we include in Σ_2 the tgd:

$$\forall x\big((\varphi_G \wedge P_1(x)) \rightarrow P_2(x)\big).$$

We prove next that (J, Σ_2) is a knowledge base solution of (I, Σ_1) if and only if G is 3-colorable. Notice that $\Sigma_1 = \emptyset$, then, by Lemma 7.5, it is enough to prove that $(I, \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ if and only if G is 3-colorable. Thus, assume that G is 3-colorable. Notice that, in this case, $J \models \varphi_G$, thus, since $J \models P_1(c)$ we have that $\text{chase}_{\Sigma_2}(J) \models P_2(c)$ which implies that $(I, \text{chase}_{\Sigma_2}(J)) \models \Sigma_{12}$ completing one direction. Towards the opposite direction, assume that G is not 3-colorable. Thus, we have that $P_2^{\text{chase}_{\Sigma_2}(J)} = \emptyset$, which implies that $(I, \text{chase}_{\Sigma_2}(J)) \not\models \Sigma_{12}$. This completes the proof of property (2).

We conclude with the proof of property (3). To prove the membership in coNP, assume that (J, Σ_2) is not a knowledge base solution of (I, Σ_1). Then, we know that there is a st-tgd σ in Σ_{12} of the form $\forall \bar{x}(\exists \bar{y}\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\psi(\bar{x}, \bar{z}))$, such that $(\text{chase}_{\Sigma_1}(I), \text{chase}_{\Sigma_2}(J)) \not\models \sigma$. Thus, we can guess σ and tuples $\bar{a}$ and $\bar{b}$, and then check that $\text{chase}_{\Sigma_1}(I) \models \varphi(\bar{a}, \bar{b})$ and $\text{chase}_{\Sigma_2}(J) \not\models \exists \bar{z}\psi(\bar{a}, \bar{z})$. By the discussion in the second paragraph of the proof of Theorem 7.4, and since $\mathbf{S}_1$ and Σ_1 are fixed, we know that checking that $\text{chase}_{\Sigma_1}(I) \models$

$\varphi(\bar{a}, \bar{b})$ can be done in NP. Moreover, since Σ_2 is fixed, checking that $\text{chase}_{\Sigma_2}(J) \not\models \exists\bar{z}\psi(\bar{a}, \bar{z})$ can be done in polynomial time by computing $\text{chase}_{\Sigma_2}(J)$ and then evaluating $\exists\bar{z}\psi(\bar{a}, \bar{z})$. We have shown that checking that (J, Σ_2) is not a knowledge base solution of (I, Σ_1) is in NP, which was to be shown.

To prove that the problem is coNP-hard, we use a reduction from the complement of the 3-COLORABILITY problem. Consider the schemas $\mathbf{S}_1 = \{E(\cdot,\cdot), P_1(\cdot), P_2(\cdot)\}$, and $\mathbf{S}_2 = \{R(\cdot)\}$, the set of formulas $\Sigma_{12} = \{\forall x(P_2(x) \rightarrow R(x))\}$, and let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$. Given a graph G, we construct from G an instance of CHECKSOLUTION($\mathcal{M}, \mathcal{K}_{\text{full-tgd}}$) as follows. Consider the instance J of $\mathbf{S}_2$ such that $R^J = \emptyset$, and $\Sigma_2 = \emptyset$. Notice that Σ_2 is fixed. Now, instance I is such that $E^I = \{(i, j) \mid i, j \in \{1, 2, 3\} \text{ and } i \neq j\}$, $P_1^I = \{c\}$ and $P_2^I = \emptyset$. To construct Σ_1, consider the formula φ_G defined in the proof of part (2). Then, we have that $I \models \varphi_G$ if and only if G is 3-colorable. Now we include in Σ_1 the tgd $\forall x\big((\varphi_G \wedge P_1(x)) \rightarrow P_2(x)\big)$.

We prove next that (J, Σ_2) is a knowledge base solution of (I, Σ_1) if and only if G is not 3-colorable. Notice that, since I and J are ground, and $\Sigma_2 = \emptyset$, then, by Lemma 7.5, it is enough to prove that $(\text{chase}_{\Sigma_1}(I), J) \models \Sigma_{12}$ if and only if G is not 3-colorable. Thus, assume that G is 3-colorable. Notice that, in this case, $I \models \varphi_G$, thus, since $I \models P_1(c)$ we have that $\text{chase}_{\Sigma_1}(I) \models P_2(c)$. Also notice that $J \not\models R(c)$, thus implying that $(\text{chase}_{\Sigma_1}(I), J) \not\models \Sigma_{12}$ completing one direction. Towards the opposite direction, assume that G is not 3-colorable. Thus, we have that $P_2^{\text{chase}_{\Sigma_1}(I)} = \emptyset$, which implies that $(\text{chase}_{\Sigma_1}(I), J) \models \Sigma_{12}$. This completes the proof. □

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* [Deutsch and Tannen 2003; Fagin et al. 2005a]. 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. The first question to answer is what is a *good* knowledge base to materialize. This question turns out to be considerably more complex that in the relational case in which no explicit knowledge is present [Fagin et al. 2005a]. We show how our general framework for defining solutions in the presence of representation systems allows us to select several possible good solutions in the knowledge-base case. 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. It is important to notice that the problem of what is the better notion of solution in the knowledge-base case is not settled yet, and several other works have explored this subject based on our formalization [Arenas et al. 2011, 2012a, 2012b].

Given the undecidability results about knowledge bases specified by (non-full) tgds, proved in Section 7.1, we focus our investigation on full tgds. Note that this case includes some of the motivating scenarios for our investigation, such as RDFS graphs [Hayes 2004].

8.1. Universal $\mathcal{K}$-Solutions

Let $\mathcal{K} = (\mathbf{K}, \text{Mod})$ be the representation system of knowledge bases, where $\mathbf{K}$ denotes all possible knowledge bases, and Mod denotes the *models* of a knowledge base, as defined in Section 7. 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

$$\text{Mod}(J, \Sigma_2) = \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1)). \tag{23}$$

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 straightforward 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 $\mathcal{M}$.*

Proof. The proof follows directly from the fact that $(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)), \emptyset)$ is a universal $\mathcal{K}$-solution of (I, Σ_1). Let $k_{\mathbf{S}_1}$ be the sum of the arities of the relational symbols in $\mathbf{S}_1$, and similarly, $k_{\mathbf{S}_2}$ the sum of the arities of the relational symbols in $\mathbf{S}_2$. Notice that the number of facts in $\text{chase}_{\Sigma_1}(I)$ is bounded by $|\,\text{dom}(I)|^{k_{\mathbf{S}_1}}$, every fact of at most size $k_{\mathbf{S}_1}$, which is polynomial since $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ is fixed. Similarly, the number of facts in $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ is bounded by $(|\,\text{dom}(I)|^{k_{\mathbf{S}_1}})^{k_{\mathbf{S}_2}}$, every fact of at most size $k_{\mathbf{S}_2}$, which is also polynomial. □

Moreover, it immediately follows from Eq. (23) 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 in the target schema of the implicit knowledge.

Example 8.2. Let (I, Σ_1), (J, Σ_2) and $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ be as in Example 7.1. In that example, (J, Σ_2) can be considered as a good solution for (I, Σ_1) under $\mathcal{M}$ since it make non-trivial use of the implicit knowledge in the target. 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 for this 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 $K' \in \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$, while $\text{Mod}(J, \Sigma_2)$ 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 $F'(a, b) \in K'$ and $F'(b, e) \in K'$, but $GP'(a, e) \notin K'$). □

In what follows, we introduce a new class of good $\mathcal{K}$-solutions that captures the solution 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 Eq. (23) 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. Thus, this 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:

$$\text{Mod}(J, \Sigma_2) \equiv_{\text{Min}} \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1)).$$

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 as it can be characterized completely in terms of the standard chase procedure.

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

Proof. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$. First, notice that, by the classical properties of the chase with full-tgds, we have that $\text{Min}(\text{Mod}(I, \Sigma_1)) = \{\text{chase}_{\Sigma_1}(I)\}$. Similarly, $\text{Min}(\text{Mod}(J, \Sigma_2)) = \{\text{chase}_{\Sigma_2}(J)\}$. Next, we compute $\text{Min}(\text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1)))$. Let $K \in \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$. Then, we know that there exists $L \in \text{Mod}(I, \Sigma_1)$ such that $(L, K) \in \mathcal{M}$. Since $\text{chase}_{\Sigma_1}(I) \subseteq L$ and $\mathcal{M}$ is closed-down on the left, we have that $(\text{chase}_{\Sigma_1}(I), K) \in \mathcal{M}$. This implies that $\text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1)) = \text{Sol}_{\mathcal{M}}(\text{chase}_{\Sigma_1}(I))$. Moreover, from classical data exchange results and since Σ_{12} is a set of full-tgds, we know that for every instance I' of $\mathbf{S}_1$ and every instance $L \in \text{Sol}_{\mathcal{M}}(I')$ it holds that $\text{chase}_{\Sigma_{12}}(I') \subseteq L$. Thus, by letting $I' = \text{chase}_{\Sigma_1}(I)$, we obtain that for every $L \in \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$ it holds that $\text{chase}_{\Sigma_{12}}(I') = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) \subseteq L$. This implies that $\text{Min}(\text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))) = \{\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))\}$.

We have shown so far that $\text{Min}(\text{Mod}(J, \Sigma_2)) = \{\text{chase}_{\Sigma_2}(J)\}$, and that $\text{Min}(\text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))) = \{\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))\}$. Thus, we obtain that $\text{Mod}(J, \Sigma_2) \equiv_{\text{Min}} \text{Sol}_{\mathcal{M}}(\text{Mod}(I, \Sigma_1))$ holds if and only if $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) = \text{chase}_{\Sigma_2}(J)$. This completes the proof of the proposition. □

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:

$$\begin{aligned} I' = \{&F(a, b), M(c, b), F(b, d), P(a, b), \\ &P(c, b), P(b, d), GP(a, d), GP(c, d)\}. \end{aligned}$$

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, since we have 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). $\square$

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 the implicit knowledge in the target to depend 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 also consider 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. Before presenting any result of this section, we need to introduce some terminology. We say that a set Σ is an *arbitrary set of dependencies* over a schema $\mathbf{S}$ when we only assume that Σ defines a set of instances of $\mathbf{S}$. Abusing notation, we use $I \models \Sigma$ to denote the fact that I is an instance in the set defined by Σ. For arbitrary sets of dependencies Σ and Σ', we say that Σ implies Σ' if every instance in the set defined by Σ is also in the set defined by Σ'. Notice that this definition is consistent with the definition of logical implication of sentences in, e.g., first-order logic. In fact, for a set of FO-sentences Σ and an arbitrary set of dependencies Σ', we have that Σ implies Σ' if every instance that satisfies the FO-sentences in Σ is an instance in the set defined by Σ'. We use the usual notation $\Sigma \models \Sigma'$ to indicate that Σ implies Σ' for arbitrary sets of dependencies.

With this notation, we can introduce one of the main notions used in this section. Let $\mathcal{M}$ be a mapping from a schema $\mathbf{S}_1$ to a schema $\mathbf{S}_2$, 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 arbitrary set of dependencies Σ_2' that is safe for Σ_1 and $\mathcal{M}$, it holds that Σ_2 implies Σ_2' ($\Sigma_2 \models \Sigma_2'$).

In this section, we first present an algorithm that is able to compute in polynomial time optimal-safe sets. Unfortunately, the output of this algorithm is a set of second-order logic (SO) sentences. A natural question is whether one can improve this algorithm to provide, e.g., a set of FO-sentences. We show that this cannot be done as FO is not expressive enough to specify optimal-safe sets. Finally, we provide an algorithm that although it does not provide optimal-safe sets, it can compute nontrivial safe sets for acyclic knowledge bases.

To present our first algorithm, we need to introduce some terminology. In what follows, we use a procedure COMPOSE that given pairwise disjoint schemas $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, a set Σ_1 of logical sentences over $\mathbf{S}_1 \cup \mathbf{S}_2$ and a set Σ_2 of sentences over $\mathbf{S}_2 \cup \mathbf{S}_3$, computes a set Σ of sentences over $\mathbf{S}_1 \cup \mathbf{S}_3$ such that $(I, J) \models \Sigma$ if and only if there exists K such that $(I, K) \models \Sigma_1$ and $(K, J) \models \Sigma_2$. As pointed out in Nash et al. [2005], there exists a straightforward implementation of COMPOSE when Σ_1 and Σ_2 are sets of FO sentences; if $\Sigma_1 = \{\sigma_1, \dots, \sigma_n\}$, $\Sigma_2 = \{\gamma_1, \dots, \gamma_m\}$ are set of FO-sentences, and $\mathbf{S}_2 = \{S_1, \dots, S_k\}$, then a set Σ consisting of second-order formula $\exists S_1 \cdots \exists S_k\, (\sigma_1 \wedge \cdots \wedge \sigma_n \wedge \gamma_1 \wedge \cdots \wedge \gamma_m)$ satisfies this condition. It should be noticed that second-order quantification is unavoidable to express the composition of mappings specified by FO sentences, even for the case of st-tgds [Fagin et al. 2005b]. In what follows, we use COMPOSE as a black box, which could have been implemented by considering the idea shown above and the techniques presented in Fagin et al. [2005b] and Nash et al. [2005]. We use COMPOSE in Step 4 of the following algorithm to create an optimal-safe set.

Example 8.7. Let $\mathbf{S}_1 = \{E(\cdot,\cdot), A(\cdot), B(\cdot), P(\cdot)\}$ and $\mathbf{S}_2 = \{F(\cdot,\cdot), R(\cdot)\}$, and consider the following sets of dependencies:

$$\begin{aligned}\Sigma_1 &= \{E(x,y) \rightarrow A(x) \wedge B(y),\ A(x) \wedge B(x) \rightarrow P(x)\}\\ \Sigma_{12} &= \{E(x,y) \rightarrow F(x,y),\ P(x) \rightarrow R(x)\}\end{aligned}$$

Next, we show how algorithm OPTIMALSAFE computes the output Σ_2, given $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and Σ_1 as input. Given that Σ_{12} consists of *copying dependencies*, the sets $\mathcal{C}_F$ and $\mathcal{C}_R$ generated in Step 1 are very simple:

$$\begin{aligned}\mathcal{C}_F &= \{\exists z_1 \exists z_2 (E(z_1, z_2) \wedge z_1 = x \wedge z_2 = y)\},\\ \mathcal{C}_R &= \{\exists z (P(z) \wedge z = x)\}.\end{aligned}$$

ALGORITHM: OPTIMALSAFE($\mathcal{M}$, Σ_1)

```
Input:  $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ with $\Sigma_{12}$ a set of full-tgds from $\mathbf{S}_1$ to $\mathbf{S}_2$, and a set $\Sigma_1$ of full-tgds
        over $\mathbf{S}_1$.
Output: A set $\Sigma_2$ of SO-formulas over $\mathbf{S}_2$ that is optimal-safe for $\Sigma_1$ and $\mathcal{M}$.

1. For every $n$-ary relational symbol $R$ of $\mathbf{S}_2$ construct a set of FO sentences $\mathcal{C}_R$ as follows:
   (a) Begin with $\mathcal{C}_R = \emptyset$ and let $\bar{x}$ be an $n$-tuple of distinct variables not mentioned in $\Sigma_1$.
   (b) For every dependency $\varphi(\bar{z}) \rightarrow R(\bar{z})$ in $\Sigma_1$ with $\bar{z}$ an $n$-tuple of not necessarily distinct
       variables, add formula $\exists \bar{z}(\varphi(\bar{z}) \wedge \bar{z} = \bar{x})$ to $\mathcal{C}_R$.
2. Construct a set of formulas $\Sigma_{12}^-$ over $\mathbf{S}_2 \cup \mathbf{S}_1$ as follows. Let $\Sigma_{12}^- = \emptyset$. For every $n$-ary
   relational symbol $R$ of $\mathbf{S}_2$, let $\bar{x}$ be an $n$-tuple of distinct variables and
   (a) if $\mathcal{C}_R \neq \emptyset$, then add to $\Sigma_{12}^-$ the formula $R(\bar{x}) \rightarrow \alpha(\bar{x})$, where $\alpha(\bar{x})$ is the disjunction of
       the formulas in $\mathcal{C}_R$.
   (b) if $\mathcal{C}_R = \emptyset$, then add to $\Sigma_{12}^-$ the formula $\forall \bar{x}(\neg R(\bar{x}))$.
3. Let $\hat{\mathbf{S}}_2$ be a copy of $\mathbf{S}_2$ defined as $\{\hat{R} \mid R \in \mathbf{S}_2\}$, and $\Sigma'_{12}$ the set obtained from $\Sigma_{12}$ by
   replacing every symbol $R$ of $\mathbf{S}_2$ by its copy $\hat{R}$.
4. Let $\Sigma'$ be the set of SO-formulas over $\mathbf{S}_2 \cup \hat{\mathbf{S}}_2$ that is obtained as the output of COMPOSE($\Sigma_{12}^- \cup$
   $\Sigma_1$, $\Sigma'_{12}$).
5. Let $\Sigma_2$ be the set of formulas over $\mathbf{S}_2$ obtained from $\Sigma'$ by replacing every symbol $\hat{R}$ of $\hat{\mathbf{S}}_2$
   by $R$. Return $\Sigma_2$.
```

Notice that $\mathcal{C}_F$ is equivalent to $\{E(x, y)\}$ and $\mathcal{C}_R$ equivalent to $\{P(x)\}$, thus Σ_{12}^- is given by the following set of dependencies from $\mathbf{S}_2$ to $\mathbf{S}_1$

$$\Sigma_{12}^- = \{F(x, y) \to E(x, y),\ R(x) \to P(x)\}$$

Now, Σ'_{12} is given by the set of dependencies $\{E(x, y) \to \hat{F}(x, y),\ P(x) \to \hat{R}(x)\}$, and we need to compose $\Sigma_{12}^- \cup \Sigma_1$ with Σ'_{12}. By using the general strategy for composing dependencies (mentioned in the paragraph before the algorithm), we obtain that $\textsc{Compose}(\Sigma_{12}^- \cup \Sigma_1, \Sigma'_{12})$ can be expressed as the following SO-formula over $\mathbf{S}_2 \cup \hat{\mathbf{S}}_2$:

$$\begin{aligned}\exists E \exists A \exists B \exists P \big(&\forall x \forall y \big(F(x, y) \to E(x, y)\big) \ \wedge\ \forall x \big(R(x) \to P(x)\big)\\ &\wedge \forall x \forall y \big(E(x, y) \to A(x) \wedge B(y)\big) \ \wedge\ \forall x \big(A(x) \wedge B(x) \to P(x)\big)\\ &\wedge \forall x \forall y \big(E(x, y) \to \hat{F}(x, y)\big) \ \wedge\ \forall x \big(P(x) \to \hat{R}(x)\big)\big)\end{aligned}$$

Finally, the algorithm replaces $\hat{F}(x, y)$ and $\hat{R}(x)$ by $F(x, y)$ and $R(x)$, respectively, in this formula. Notice that, after this replacement, we obtain the dependencies $F(x, y) \to E(x, y)$ and $E(x, y) \to F(x, y)$ as conjuncts in the formula, which implies that we can eliminate the second-order quantification over relation E. Similarly, we can eliminate the second-order quantification over P and, thus, the output Σ_2 of the algorithm is (equivalent to) the following SO-formula over $\mathbf{S}_2 = \{F(\cdot, \cdot), R(\cdot)\}$:

$$\exists A \exists B \big(\forall x \forall y \big(F(x, y) \to A(x) \wedge B(y)\big) \ \wedge\ \forall x \big(A(x) \wedge B(x) \to R(x)\big)\big).$$

It can be proved that Σ_2 is optimal-safe for Σ_1 and Σ_{12} (see Theorem 8.9). □

Before proving that OptimalSafe is correct, we provide a characterization of safe sets that is instrumental in the proof of correctness.

Proposition 8.8. *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 $\mathbf{S}_1$. An arbitrary set of dependencies Σ_2 over $\mathbf{S}_2$ is safe for Σ_1 and $\mathcal{M}$ iff $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) \models \Sigma_2$ for every instance I of $\mathbf{S}_1$.*

Proof

($\Leftarrow$) Let I be a source instances, and let $J^* = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ we show now that (J^*, Σ_2) is a minimal knowledge-base solution of (I, Σ_1) under $\mathcal{M}$. From the proof of Proposition 8.4, we know that $\text{Min}(\textsc{Sol}_{\mathcal{M}}(\textsc{Mod}(I, \Sigma_1))) = \{J^*\}$. Then, we only need to show that $\text{Min}(\textsc{Mod}(J^*, \Sigma_2)) = \{J^*\}$. Notice that $\textsc{Mod}(J^*, \Sigma_2) = \{K \mid J^* \subseteq K$ and $K \models \Sigma_2\}$. Then, since $J^* \models \Sigma_2$, we have that $J^* \in \textsc{Mod}(J^*, \Sigma_2)$ and, for every $K \in \textsc{Mod}(J^*, \Sigma_2)$, it holds that $J^* \subseteq K$, which implies that J^* is the only minimal instance in $\textsc{Mod}(J^*, \Sigma_2)$.

($\Rightarrow$) To obtain a contradiction, assume that there exists an instance I such that $J^* = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) \not\models \Sigma_2$. We show next that for every K, (K, Σ_2) is not a minimal knowledge-base solution of (I, Σ_1). From the proof of Proposition 8.4, we know that $\text{Min}(\textsc{Sol}_{\mathcal{M}}(\textsc{Mod}(I, \Sigma_1))) = \{J^*\}$. But $\text{Min}(\textsc{Mod}(K, \Sigma_2)) \neq \{J^*\}$ since $J^* \not\models \Sigma_2$ and thus $J^* \notin \textsc{Mod}(K, \Sigma_2)$. Thus, we obtain that (K, Σ_2) is not a minimal knowledge-base solution of (I, Σ_1). □

We are now ready to prove the correctness of OptimalSafe.

Theorem 8.9. *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, with input $\mathcal{M}$ and Σ_1, algorithm* OptimalSafe *computes a set Σ_2 of second-order logic sentences that is optimal-safe for Σ_1 and $\mathcal{M}$. Moreover, algorithm* OptimalSafe *runs in polynomial time.*

PROOF. We first show that Σ_2 is safe. From Proposition 8.8, it is enough to show that for every I the instance $J^* = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ is such that $J^* \models \Sigma_2$. Let $\hat{J}^*$ be the instance of $\hat{\mathbf{S}}_2$ obtained from J^* by replacing every symbol $R \in \mathbf{S}_2$ by its copy $\hat{R}$. By the construction of set Σ_2 in the algorithm, to show that $J^* \models \Sigma_2$, it is enough to show that $(J^*, \hat{J}^*) \models \Sigma'$, where Σ' is the set constructed in Step 4. Thus, we need to show that there exists an instance K such that $(J^*, K) \models \Sigma_{12}^- \cup \Sigma_1$ and $(K, \hat{J}^*) \models \Sigma'_{12}$. We claim that $K = \text{chase}_{\Sigma_1}(I)$ satisfies these conditions. First, notice that $\text{chase}_{\Sigma_1}(I) \models \Sigma_1$. Moreover, from the construction of Σ'_{12} we know that $(\text{chase}_{\Sigma_1}(I), \hat{J}^*) \models \Sigma'_{12}$, since $J^* = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$. Thus, we only need to to prove that $(J^*, \text{chase}_{\Sigma_1}(I)) \models \Sigma_{12}^-$. Or, since $\text{chase}_{\Sigma_1}(I) = K$, then we need to show that $(\text{chase}_{\Sigma_{12}}(K), K) \models \Sigma_{12}^-$. Choose an arbitrary sentence σ from Σ_{12}^-. We have the following two options.

(1) The sentence σ is of form $\forall \bar{x}(\neg R(\bar{x}))$. Then, we know that R does not appear in the conclusion of any dependency in Σ_{12}, and thus clearly, $\text{chase}_{\Sigma_{12}}(K) \models \sigma$, and $(\text{chase}_{\Sigma_{12}}(K), K) \models \sigma$.
(2) Sentence σ is a dependency of the form $R(\bar{x}) \rightarrow \alpha(\bar{x})$. To show that $(\text{chase}_{\Sigma_{12}}(K), K) \models \sigma$, assume that $\text{chase}_{\Sigma_{12}}(K) \models R(\bar{a})$ for some tuple $\bar{a}$. We need to show that $K \models \alpha(\bar{a})$. Since $\text{chase}_{\Sigma_{12}}(K) \models R(\bar{a})$, then the fact must have been added in some chase step, thus we know that there exists a dependency $\varphi(\bar{z}) \rightarrow R(\bar{z})$ in Σ_{12} such that $K \models \varphi(\bar{a})$. Therefore, by the construction of $\alpha(\bar{x})$, we know that it contains a disjunct of the form $\exists \bar{z}(\varphi(\bar{z}) \wedge \bar{z} = \bar{x})$. Finally, since $K \models \exists \bar{z}(\varphi(\bar{z}) \wedge \bar{z} = \bar{a})$ we have that $K \models \alpha(\bar{a})$ which was to be shown.

We then have that $(\text{chase}_{\Sigma_{12}}(K), K) \models \Sigma_{12}^-$. Thus, we have shown that for every instance I it holds that $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ satisfies every formula in Σ_2, which proves that Σ_2 is safe.

We now show that if Σ'_2 is a safe set of formulas, then Σ_2 logically implies Σ'_2. We first show the following property: if $K \models \Sigma_2$, then there exists an instance L such that $K = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(L))$. Assume that $K \models \Sigma_2$, and let $\hat{K}$ be the instance obtained from K by replacing every relation $R \in \mathbf{S}_2$ by its copy $\hat{R}$. Then, we know that $(K, \hat{K}) \models \Sigma'$ where Σ' is the set constructed in Step 4. Therefore, there exists an instance L of $\mathbf{S}_1$ such that $(K, L) \models \Sigma_{12}^- \cup \Sigma_1$ and $(L, \hat{K}) \models \Sigma'_{12}$. We show next that $K = \text{chase}_{\Sigma_{12}}(L)$. First, from $(L, \hat{K}) \models \Sigma'_{12}$ we know that $\text{chase}_{\Sigma_{12}}(L) \subseteq K$. Next, we show that $K \subseteq \text{chase}_{\Sigma_{12}}(L)$. Assume that there is a relation R over $\mathbf{S}_2$ and a tuple $\bar{a}$ such that $K \models R(\bar{a})$. Notice that if R is not in the conclusion of a dependency in Σ_{12} then the formula $\forall \bar{x}(\neg R(\bar{x}))$ is in Σ_{12}^-, which contradicts the fact that $(K, L) \models \Sigma_{12}^-$. Then, we know that R is in the conclusion of a dependency in Σ_{12} and, moreover, Σ_{12}^- contains a dependency of the form $R(\bar{x}) \rightarrow \alpha(\bar{x})$ (constructed in Step 2 of the algorithm). Then, $L \models \alpha(\bar{a})$, which implies that there is a disjunct in $\alpha(\bar{x})$ of the form $\exists \bar{z}(\varphi(\bar{z}) \wedge \bar{z} = \bar{x})$ such that $L \models \exists \bar{z}(\varphi(\bar{z}) \wedge \bar{z} = \bar{a})$, and then $L \models \varphi(\bar{a})$. Moreover, we also know that $\varphi(\bar{z}) \rightarrow R(\bar{z})$ is a dependency in Σ_{12}. Finally, since $L \models \varphi(\bar{a})$, we have that $\text{chase}_{\Sigma_{12}}(L) \models R(\bar{a})$. We have shown that, if $K \models R(\bar{a})$, then $\text{chase}_{\Sigma_{12}}(L) \models R(\bar{a})$. From this, we conclude that $K \subseteq \text{chase}_{\Sigma_{12}}(L)$, which was to be shown. Finally, since $L \models \Sigma_1$, we have that $\text{chase}_{\Sigma_1}(L) = L$. Thus, we have shown that, if $K \models \Sigma_2$, then there exists an instance L such that $K = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(L))$.

Now let Σ'_2 be a safe set. By using the property shown in the previous paragraph, we can show that, if $K \models \Sigma_2$, then $K \models \Sigma'_2$, and then Σ_2 logically implies Σ'_2. Then assume that $K \models \Sigma_2$. We know that there exists an instance L such that $K = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(L))$. Thus, since Σ'_2 is safe by Proposition 8.8, we obtain that $K = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(L)) \models \Sigma'_2$. $\square$

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 8.10. *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 $\mathcal{M}$.*

Proof. Let $\mathbf{S}_1 = \{E(\cdot,\cdot), F(\cdot,\cdot), A(\cdot,\cdot), C(\cdot), W(x,y)\}$ and consider the set Σ_1 of full-tgds over Σ_1 given by:

$$\begin{aligned} E(x,y) \wedge A(x,u) \wedge A(x,v) &\rightarrow F(x,y) \\ E(x,y) \wedge A(x,u) \wedge A(x,u) &\rightarrow W(x,y) \\ A(x,u) &\rightarrow C(u). \end{aligned}$$

Let $\mathbf{S}_2 = \{F'(\cdot,\cdot), W'(\cdot,\cdot), C'(\cdot)\}$ and consider the mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ where Σ_{12} is the set of dependencies $\{F(x,y) \rightarrow F'(x,y), W(x,y) \rightarrow W'(x,y), C(x) \rightarrow C'(x)\}$.

Now assume that Σ_2 is safe for Σ_1 and $\mathcal{M}$. Further assume that if Σ_2' is also safe, then Σ_2 logically implies Σ_2'. Then, by Theorem 8.9, we know that Σ_2 is logically equivalent to the output of Algorithm 1. Thus, from the proof of Theorem 8.9, we know that an instance $J \models \Sigma_2$ if and only if there exists an instance I such that $J = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$.

We prove now that Σ_2 is not FO-definable. To obtain a contradiction, assume that Σ_2 contains only FO sentences, and let ψ be the conjunction of the sentences in Σ_2. Now let u, v, w be variables that are not mentioned in ψ and consider the formula ψ' obtained from ψ by:

—replacing every relational atom $C'(x)$ by the formula $x = u \vee x = v \vee x = w$, and
—replacing every relational atom $W'(x,y)$ by the formula $x \neq x \wedge y \neq y$.

Now let α be the FO sentence:

$$\exists u \exists v \exists w \big(u \neq v \wedge u \neq w \wedge v \neq w \wedge \psi'\big).$$

Notice that α is an FO sentence over the schema $\mathbf{S}' = \{F'(\cdot,\cdot)\}$. Let K be an arbitrary instance of $\mathbf{S}'$ and J an instance of $\mathbf{S}_2$ such that $F'^J = F'^K$, $W'^J = \emptyset$ and $C'^J = \{a,b,c\}$ (with a,b,c arbitrary different values). Notice that $K \models \alpha$ if and only if $J \models \psi$. To see this, just notice that restricted to the class of instances of $\mathbf{S}_2$ that interprets W' as the empty set and C' as a set with three different elements a, b, and c, the formula $W'(x,y)$ is equivalent to $x \neq x \wedge y \neq y$, and the formula $C'(x)$ is equivalent to $(x = a \vee x = b \vee x = c)$. Moreover, given the definition of Σ_1 and Σ_{12} it is straightforward to show that an instance of $\mathbf{S}_2$ that interprets W' as the empty set and C' as a set with three different elements a, b, and c satisfies ψ if and only if the interpretation of F' is 3-colorable. Thus, by this discussion, we have that α defines 3-colorability, which is a contradiction since 3-colorability is not expressible in FO (in fact, it is not expressible even in the infinitary logic $\mathcal{L}^{\omega}_{\infty\omega}$ [Dawar 1998], which is more expressive than FO). $\square$

Theorem 8.10 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 that is specified by a set of full tgds with inequalities in their premises.

Formally, 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-known 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 $\beta(\bar{x}) \to R(\bar{x})$ 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. We use the unfolding of an acyclic set Σ of full tgds as an intermediate step in our algorithm (as well as in the proof of Lemma 8.13).

To present our algorithm, we need to introduce some additional 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 Arenas et al. [2010] 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 Arenas et al. [2010]). Moreover, from the results in Arenas et al. [2010], 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 Fagin et al. [2005b], 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.

Theorem 8.11. *Given a mapping $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where Σ_{12} is a set of full st-tgds, and an acyclic set Σ_1 of full tgds over $\mathbf{S}_1$, algorithm* 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 $\mathcal{M}$.*

Proof. According to the definition, to prove that Σ_2 is safe, we need to show that for every instance I of $\mathbf{S}_1$ the instance $J^* = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ is such that $J^* \models \Sigma_2$. Let $\hat{J}^*$ be the instance of $\hat{\mathbf{S}}_2$ obtained from J^* by replacing every symbol $R \in \mathbf{S}_2$ by its copy $\hat{R}$. By the construction of set Σ_2 in the algorithm, to show that $J^* \models \Sigma_2$ it is enough to show that $(J^*, \hat{J}^*) \models \Sigma''$, where Σ'' is the set constructed in Step 5. Thus, we need to show that there exists an instance K of $\mathbf{S}_1$ such that $(J^*, K) \models \Sigma'$ and $(K, \hat{J}^*) \models \Sigma'_{12}$, where Σ' is the set constructed in Step 2 and Σ'_{12} is the set constructed in Step 3. We claim that the instance $K = \text{chase}_{\Sigma_1}(I)$ over $\mathbf{S}_1$ satisfies these conditions.

First, notice that $(K, \hat{J}^*) \models \Sigma'_{12}$ since $J^* = \text{chase}_{\Sigma_{12}}(K)$. Thus, we only need to to prove that $(J^*, K) \models \Sigma'$. Let $\gamma(\bar{x}) \to R(\bar{x})$ be a dependency in Σ', and assume that $J^* \models \gamma(\bar{a})$ for some tuple $\bar{a}$. Then, by the construction of Σ', we know that $J^* \models \beta(\bar{a})$ where $\beta(\bar{x})$ is a target rewriting of a formula $\alpha(\bar{x})$ under $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ such that $\alpha(\bar{x}) \to R(\bar{x})$ is a dependency in Σ_1^{+}. Now, it is well known that if $\mathcal{M}$ is specified by full-tgds and Q is a target query in $\text{UCQ}^{=,\neq}$, then, for every instance L of $\mathbf{S}_1$, it holds that $\text{certain}(Q, \text{Sol}_{\mathcal{M}}(L)) = Q(\text{chase}_{\Sigma_{12}}(L))$. Thus, since $\beta(\bar{x})$ is a target rewriting of $\alpha(\bar{x})$ and $J^{\star} = \text{chase}_{\Sigma_{12}}(K) \models \beta(\bar{a})$, then $K \models \alpha(\bar{a})$. Finally, since $K = \text{chase}_{\Sigma_1}(I)$, we have that, $K \models \Sigma_1^{+}$, obtaining that $K \models R(\bar{a})$ since $\alpha(\bar{x}) \to R(\bar{x})$ is in Σ_1^{+}. $\square$

```
ALGORITHM: FullSafe($\mathcal{M}$, $\Sigma_1$)
Input: $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where $\Sigma_{12}$ is a set of full st-tgds, and an acyclic set $\Sigma_1$ of full tgds
  over $\mathbf{S}_1$.
Output: A set $\Sigma_2$ of full tgds with inequalities over $\mathbf{S}_2$ that is safe for $\Sigma_1$ and $\mathcal{M}$.
  1. Construct a set of formulas $\Sigma_1^+$ by unfolding $\Sigma_1$.
  2. Construct a set $\Sigma'$ of full st-tgds with inequalities from $\mathbf{S}_2$ to $\mathbf{S}_1$ as follows. Begin with
     $\Sigma' = \emptyset$. For every tgd $\alpha(\bar{x}) \to R(\bar{x})$ in $\Sigma_1^+$ do the following:
     2.1. If $\alpha(\bar{x})$ is target rewritable under $\mathcal{M}$, then let $\beta(\bar{x})$ be the query in UCQ$^{=,\neq}$ over $\mathbf{S}_2$
          that is the output of TRew($\mathcal{M}, \alpha(\bar{x})$). For every disjunct $\gamma(\bar{x})$ in $\beta(\bar{x})$, add to $\Sigma'$ the
          dependency $\gamma(\bar{x}) \to R(\bar{x})$ (and eliminate equalities by using variable substitutions).
  3. Let $\hat{\mathbf{S}}_2$ be a copy of $\mathbf{S}_2$ defined as $\{\hat{R} \mid R \in \mathbf{S}_2\}$, and $\Sigma'_{12}$ the set of full st-tgds from $\mathbf{S}_1$ to $\hat{\mathbf{S}}_2$
     obtained from $\Sigma_{12}$ by replacing every $R \in \mathbf{S}_2$ by $\hat{R}$.
  4. Let $\Sigma''$ 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 ComposeFull($\Sigma'$, $\Sigma'_{12}$).
  5. Let $\Sigma_2$ be the set of formulas over $\mathbf{S}_2$ obtained from $\Sigma''$ by replacing every symbol $\hat{R} \in \hat{\mathbf{S}}_2$
     by $R$. Return $\Sigma_2$.
```

Example 8.12. 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

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

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:

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

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

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

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

$$\exists y (F'(x, y) \wedge F'(y, z)) \ \rightarrow \ GP'(x, z) \tag{24}$$

is in the output of FullSafe($\mathcal{M}$, Σ_1). In fact, it can be proved that the set Σ_2 returned by FullSafe($\mathcal{M}$, Σ_1) is logically equivalent to the set consisting of dependency (24).

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 [Hayes 2004; Patel-Schneider et al. 2004], where, for a specific domain, the rules in a knowledge base remains unchanged, while the explicit data changes from one repository to another.

In this section, we present an algorithm that, given $\mathcal{M}$, Σ_1 and Σ_2 as previously stated, returns a set Σ_1' of full tgds satisfying conditions (C1) and (C2). Our algorithm works specifically for the case in which Σ_1 is an acyclic set of full tgds. The key property that allows us to develop this algorithm is stated in the following lemma. The lemma ensures that if Σ_1 is an acyclic set of full tgds, then the problem of verifying whether conditions (C1) and (C2) hold for every instance I_1 of $\mathbf{S}_1$ is decidable in exponential time.

LEMMA 8.13. *There exists an exponential-time algorithm that, given $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ with Σ_{12} a set of full st-tgds, an acyclic set of full tgds Σ_1 over $\mathbf{S}_1$, a set of full tgds Σ_1' over $\mathbf{S}_1$ such that $\Sigma_1 \models \Sigma_1'$, and a set of full tgds with inequalities Σ_2 over $\mathbf{S}_2$ that is safe for Σ_1 and $\mathcal{M}$, verifies whether*

$$\text{chase}_{\Sigma_2}\left(\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1'}(I)\right)\right) = \text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) \qquad (25)$$

holds for every instance I of $\mathbf{S}_1$.

PROOF. We first show that, for every I, it holds that

$$\text{chase}_{\Sigma_2}\left(\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1'}(I)\right)\right) \subseteq \text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right).$$

First, notice that $\text{chase}_{\Sigma_1'}(I) \subseteq \text{chase}_{\Sigma_1}(I)$ since Σ_1 and Σ_1' are sets of full-tgds and $\Sigma_1 \models \Sigma_1'$. Thus, we also have that

$$\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1'}(I)\right) \subseteq \text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right).$$

Finally, since Σ_2 is safe then $\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right) \models \Sigma_2$ and we have that

$$\begin{aligned}&\text{chase}_{\Sigma_2}\left(\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1'}(I)\right)\right)\\ &\qquad \subseteq \text{chase}_{\Sigma_2}\left(\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right)\right) = \text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right).\end{aligned}$$

Thus, to prove the lemma, we only need to show how to test that the inclusion

$$\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1}(I)\right) \subseteq \text{chase}_{\Sigma_2}\left(\text{chase}_{\Sigma_{12}}\left(\text{chase}_{\Sigma_1'}(I)\right)\right) \qquad (26)$$

holds for every instance I. In what follows we describe a procedure to test (26).

Let Σ_1^+ be the set that results from unfolding Σ_1 (recall the notion of unfolding that we use in Algorithm FULLSAFE). Now consider a schema $\hat{\mathbf{S}}_1$ that is a copy of $\mathbf{S}_1$, and let Σ_{copy} be the set of *copying* dependencies from $\hat{\mathbf{S}}_1$ to $\mathbf{S}_1$, that is, $\Sigma_{\text{copy}} = \{\hat{R}(\bar{x}) \to R(\bar{x}) \mid R \in \mathbf{S}_1\}$. Let Σ_1^* be the set obtained from Σ_1^+ by replacing every relational symbol R in the left-hand side of a dependency by its copy $\hat{R}$, and let Δ_1 be the set of dependencies $\Sigma_1^* \cup \Sigma_{\text{copy}}$. Notice that Δ_1 is a set of dependencies from $\hat{\mathbf{S}}_1$ to $\mathbf{S}_1$. By the properties of the unfolding, it is not difficult to see that the set Δ_1 satisfies the following. Let I be an arbitrary instance of $\mathbf{S}_1$, and $\hat{I}$ the instance obtained from I by replacing every relation symbol

R of $\mathbf{S}_1$ by its copy $\hat{R}$. Then, it holds that $\text{chase}_{\Delta_1}(\hat{I}) = \text{chase}_{\Sigma_1}(I)$. Now, let Γ be a set of dependencies that is logically equivalent to the composition of Δ_1 with Σ_{12}. That is, $(I, J) \models \Gamma$ if and only if there exists K such that $(I, K) \models \Sigma_1^*$ and $(K, J) \models \Sigma_{12}$.

Notice that Δ_1 is of size exponential in Σ_1, since Δ_1 contains all the dependencies in Σ_1^* which is constructed from the unfolding Σ_1^+ of Σ_1. In particular, Δ_1 has an exponential number of dependencies, each dependency of polynomial size with respect to the size of Σ_1. Moreover, if we use the typical algorithm for composing full-tgds presented in Fagin et al. [2005b] to construct Γ, then the size of Γ is exponential in Σ_{12}. More precisely, let n_{Δ_1} be the number of dependencies in Δ_1, $n_{\Sigma_{12}}$ the number of dependencies in Σ_{12}, and k_{Δ_1} be the maximum number of relational atoms in the left-hand side of a dependency in Δ_1. Then, Γ has a number of dependencies that is proportional to $(k_{\Delta_1})^{n_{\Sigma_{12}}} \times n_{\Delta_1}$, each dependency of polynomial size with respect to the size of Σ_1 and Σ_{12}. Thus, Γ has an exponential number of dependencies with respect to the size of Σ_1 and Σ_{12}, each dependency of polynomial size with respect to the size of Σ_1 and Σ_{12}.

In what follows, we assume some familiarity with Datalog programs (we refer the reader to Abiteboul et al. [1995] and Ceri et al. [1989] for a detailed introduction to Datalog). Let $\mathcal{D}$ be the Datalog program obtained by considering all the tgds in Γ as Datalog rules. Notice that $\mathcal{D}$ has $\hat{\mathbf{S}}_1$ as the *extensional database schema* and $\mathbf{S}_2$ as the *intentional schema*. Moreover, $\mathcal{D}$ is a program in which every rule is of the form $\alpha(\bar{x}) \rightarrow R(\bar{x})$ such that R does not appear in the body of any rule in $\mathcal{D}$. Now, it is clear by the construction of $\mathcal{D}$ that if we start with an instance $\hat{I}$ of $\hat{\mathbf{S}}_1$ as the extensional database, then the evaluation of $\mathcal{D}$ over $\hat{I}$ is exactly $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$.

Similarly, consider the set of full-tgds given by

$$\Gamma' \;=\; \Sigma_{\text{copy}} \cup \Sigma_1' \cup \Sigma_{12} \cup \Sigma_2.$$

Then, let $\mathcal{D}'$ be the Datalog program obtained from Γ'. Then, $\mathcal{D}'$ is a possibly recursive Datalog program with inequalities (since Σ_2 can be recursive and has inequalities) in which $\hat{\mathbf{S}}_1$ is the extensional database schema and $\mathbf{S}_1 \cup \mathbf{S}_2$ is the intentional database schema. In this case, we have that, if we start with an instance $\hat{I}$ of $\hat{\mathbf{S}}_1$ as the extensional database, then the evaluation of $\mathcal{D}'$ over $\hat{I}$ is exactly $\text{chase}_{\Sigma_1'}(I) \cup \text{chase}_{\Sigma_2}(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1'}(I)))$.

Now it is easy to argue that (26) holds for every I if and only if the program $\mathcal{D}$ is contained in $\mathcal{D}'$. First, assume that $\mathcal{D}$ is contained in $\mathcal{D}'$. Thus, for every $\hat{I}$ over $\hat{\mathbf{S}}_1$, we have that $\mathcal{D}(\hat{I}) \subseteq \mathcal{D}'(\hat{I})$, which implies that $\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I))$ is contained in $\text{chase}_{\Sigma_1'}(I) \cup \text{chase}_{\Sigma_2}(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1'}(I)))$, and thus

$$\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1}(I)) \subseteq \text{chase}_{\Sigma_2}(\text{chase}_{\Sigma_{12}}(\text{chase}_{\Sigma_1'}(I)))$$

since $\text{chase}_{\Sigma_1'}(I)$ is an instance over $\mathbf{S}_1$ which is not mentioned in $\mathcal{D}$. On the other hand, if (26) holds for every I, then $\mathcal{D}(\hat{I})$ is contained in $\mathcal{D}'(\hat{I})$ for every $\hat{I}$ in $\hat{\mathbf{S}}_1$ since $\mathcal{D}$ does not mention schema $\mathbf{S}_1$.

We have reduced the problem of testing whether (26) holds to the problem of testing the containment of a non-recursive Datalog program into a Datalog program with inequalities. It is well known that deciding whether a nonrecursive Datalog program is contained in a Datalog program is decidable in EXPTIME [Sagiv 1988]. Moreover, the nonrecursive Datalog program $\mathcal{D}$ that we need to check for containment is such that if a relation name R occurs in the right-hand side of a rule in $\mathcal{D}$, then it does not occur in the left-hand side of any rule in $\mathcal{D}$, which ensures that one can check containment of the program by considering one rule at a time [Sagiv 1988]. More precisely, it follows from the results of Sagiv [1988] that, to decide that $\mathcal{D}$ is contained in $\mathcal{D}'$, it suffices to test, for every rule r in $\mathcal{D}$, whether the program consisting of this single rule is contained in $\mathcal{D}'$. Now, since $\mathcal{D}'$ is a Datalog program with inequalities the technique for checking containment developed in Sagiv [1988] need to be adapted. This can be done using the techniques presented in Ullman [1997] as follows. To check that a rule r is contained in $\mathcal{D}'$, we first create a set $\mathcal{R}$ containing all the rules obtained from r by making equal some of the variables occurring in r (and applying the corresponding variable replacements). For instance, if r is the rule $A(x_1, x_2, x_3, x_4) \rightarrow B(x_1, x_3)$, then $\mathcal{R}$ contains, among others, the rule $A(x_1, x_1, x_3, x_4) \rightarrow B(x_1, x_3)$ obtained from r by applying the equality $x_1 = x_2$, and also the rule $A(x_1, x_2, x_2, x_1) \rightarrow B(x_1, x_2)$ obtained from r by applying the equalities $x_1 = x_4$ and $x_2 = x_3$. Then, in order to check that r is contained in $\mathcal{D}'$, one can apply the technique described in Sagiv [1988] and Ullman [1997] to every rule r' in $\mathcal{R}$: we evaluate the program $\mathcal{D}'$ over the database instance composed of the *frozen body* of r', that is, the atoms in the body of r' considering all variables as if they were distinct constant values, and then check whether the resulting instance contains the *frozen head* of r' [Sagiv 1988; Ullman 1997]. Notice that the number of rules in the set $\mathcal{R}$ is at most exponential in the size of r, and every rule in $\mathcal{R}$ is of size linear in the size of r. Thus, the complete process of checking if r is contained in $\mathcal{D}'$ can be done in exponential time with respect to the size of r and $\mathcal{D}'$.

Finally, since the number of rules in $\mathcal{D}$ is exponential in the size of Σ_1 and Σ_{12}, each rule of size polynomial, and $\mathcal{D}'$ is of size proportional to Σ_1', Σ_{12} and Σ_2, the whole process can be done in time exponential with respect to the size of Σ_1, Σ_{12}, Σ_1', and Σ_2. This completes the proof of the lemma. □

We are now ready to present an algorithm that given $\mathcal{M}$, Σ_1 and Σ_2, returns a minimal set Σ_1' of full tgds satisfying conditions (C1) and (C2). In this algorithm, we use some of the terminology introduced in Section 8.3.1 for the definition of procedure FullSafe.

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.

```
ALGORITHM: Minimize($\mathcal{M}$, $\Sigma_1$, $\Sigma_2$)
  Input:  $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$, where $\Sigma_{12}$ is a set of full st-tgds, an acyclic set $\Sigma_1$ of full tgds,
          and a set $\Sigma_2$ of full tgds with inequalities that is safe for $\Sigma_1$ and $\mathcal{M}$.
  Output: A minimal set $\Sigma_1'$ that satisfies conditions (C1) and (C2) for every instance $I_1$ of $\mathbf{S}_1$.
  1. Let $\Sigma_1^+$ be the set obtained by unfolding $\Sigma_1$, 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 $\sigma$ from $\Gamma$ and repeat Step 2.
  3. Let $\Sigma_1' = \Gamma$, and return $\Sigma_1'$.
```

Theorem 8.14. *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($\mathcal{M}$, Σ_1)*, and Σ_1' the output of* Minimize($\mathcal{M}$, Σ_1, Σ_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 for (I, Σ_1) under $\mathcal{M}$.*

Proof. This theorem is a direct consequence of Theorem 8.11, Proposition 8.4, and Lemma 8.13, which establishes the decidability of the key property used in algorithm Minimize. □

Example 8.15. Let $\mathcal{M} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and Σ_1 be as in Example 7.1. From Example 8.12, we know that the output of FullSafe($\mathcal{M}$, Σ_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($\mathcal{M}$, Σ_1, Σ_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 8.14 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. We would also like to continue studying the notion of chase that we provided in this paper (to deal with conditional instances), and see whether it can be modified to handle other models that feature relational databases with annotations of the tuples, such as the semirings for provenance presented in Green et al. [2007].

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Libaray.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous references for their many helpful comment.

REFERENCES

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.

ABITEBOUL, S., KANELLAKIS, P. C., AND GRAHNE, G. 1991. On the representation and querying of sets of possible worlds. *Theoret. Comput. Sci. 78*, 1, 158–187.

AFRATI, F., LI, C., AND PAVLAKI, V. 2008. Data exchange: Query answering for incomplete data sources. In *Proceedings of InfoScale*.

ANTOVA, L., KOCH, C., AND OLTEANU, D. 2007. 10^{10^6} worlds and beyond: Efficient representation and processing of incomplete information. In *Proceedings of ICDE*. 606–615.

ARENAS, M., BOTOEVA, E., AND CALVANESE, D. 2011. Knowledge base exchange. In *Description Logics*.

ARENAS, M., BOTOEVA, E., CALVANESE, D., RYZHIKOV, V., AND SHERKHONOV, E. 2012a. Exchanging description logic knowledge bases. In *Proceedings of KR*.

ARENAS, M., BOTOEVA, E., CALVANESE, D., RYZHIKOV, V., AND SHERKHONOV, E. 2012b. Representability in DL-lite$_R$ knowledge base exchange. In *Description Logics*.

ARENAS, M., PÉREZ, J., AND REUTTER, J. L. 2011. Data exchange beyond complete data. In *Proceedings of PODS*. 83–94.

ARENAS, M., PÉREZ, J., REUTTER, J. L., AND RIVEROS, C. 2009a. Inverting schema mappings: bridging the gap between theory and practice. *Proc. VLDB 2*, 1, 1018–1029.

ARENAS, M., PÉREZ, J., REUTTER, J. L., AND RIVEROS, C. 2010. Foundations of schema mapping management. In *Proceedings of PODS*. 227–238.

ARENAS, M., PÉREZ, J., REUTTER, J. L., AND RIVEROS, C. 2013. The language of plain SO-TGDS: Composition, inversion and structural properties. *J. Comput. System Sci. 79*, 6, 763–784.

ARENAS, M., PÉREZ, J., AND RIVEROS, C. 2009b. The recovery of a schema mapping: Bringing exchanged data back. *Trans. Datab. Syst. 34*, 4.

BEERI, C. AND VARDI, M. 1984. A proof procedure for data dependencies. *J. ACM 31*, 4, 718–741.

BERNSTEIN, P. 2003. Applying model management to classical meta data problems. In *Proceedings of CIDR*.

BERNSTEIN, P. AND MELNIK, S. 2007. Model management 2.0: manipulating richer mappings. In *Proceedings of SIGMOD*. 1–12.

BUSS, S. R. AND HAY, L. 1991. On truth-table reducibility to SAT. *Inf. Comput. 91*, 1, 86–102.

CERI, S., GOTTLOB, G., AND TANCA, L. 1989. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng. 1*, 1, 146–166.

DAWAR, A. 1998. A restricted second order logic for finite structures. *Inf. Comput. 143*, 2, 154–174.

DEUTSCH, A. AND TANNEN, V. 2003. Reformulation of XML queries and constraints. In *Proceedings of ICDT*. 225–241.

FAGIN, R. 2007. Inverting schema mappings. *Trans. Datab. Syst. 32*, 4.

FAGIN, R., KOLAITIS, P. G., MILLER, R., AND POPA, L. 2005a. Data exchange: semantics and query answering. *Theor. Comput. Sci. 336*, 1, 89–124.

FAGIN, R., KOLAITIS, P. G., POPA, L., AND TAN, W.-C. 2005b. Composing schema mappings: Second-order dependencies to the rescue. *Trans. Datab. Syst. 30*, 4, 994–1055.

FAGIN, R., KOLAITIS, P. G., POPA, L., AND TAN, W.-C. 2007. Quasi-inverses of schema mappings. In *Proceedings of PODS*. 123–132.

FAGIN, R., KOLAITIS, P. G., POPA, L., AND TAN, W.-C. 2009. Reverse data exchange: coping with nulls. In *Proceedings of PODS*. 23–32.

GRAHNE, G. 1991. *The Problem of Incomplete Information in Relational Databases*. Springer.

GRAHNE, G. AND ONET, A. 2011. Closed world chasing. In *Proceedings of LID*. 7–14.

GREEN, T. J., KARVOUNARAKIS, G., AND TANNEN, V. 2007. Provenance semirings. In *Proceedings of PODS*. 31–40.

HAYES, P. February 2004. RDF Semantics, W3C Recommendation. http://www.w3.org/TR/rdf-mt.

IMIELINSKI, T. AND LIPSKI, W. 1984. Incomplete information in relational databases. *J. ACM 31*, 4, 761–791.

KOLAITIS, P. G., PANTTAJA, J., AND TAN, W.-C. 2006. The complexity of data exchange. In *Proceedings of PODS*. 30–39.

LIBKIN, L. 2006. Data exchange and incomplete information. In *Proceedings of PODS*. 60–69.

LIBKIN, L. AND SIRANGELO, C. 2008. Data exchange and schema mappings in open and closed worlds. In *Proceedings of PODS*. 139–148.

NASH, A., BERNSTEIN, P., AND MELNIK, S. 2005. Composition of mappings given by embedded dependencies. In *Proceedings of PODS*. 172–183.

PATEL-SCHNEIDER, P., HAYES, P., AND HORROCKS, I. February 2004. OWL Web Ontology Language, W3C Recommendation. http://www.w3.org/TR/owl-semantics/.

SAGIV, Y. 1988. Optimizing datalog programs. In *Foundations of Deductive Databases and Logic Programming,* Morgan Kaufmann.

TEN CATE, B. AND KOLAITIS, P. G. 2010. Structural characterizations of schema-mapping languages. *Comm. ACM 53*, 1, 101–110.

ULLMAN, J. D. 1997. Information integration using logical views. In *Proceedings of ICDT*. 19–40.

VARDI, M. 1982. The complexity of relational query languages. In *Proceedings of STOC*. 137–146.

WAGNER, K. W. 1987. More complicated questions about maxima and minima, and some closures of NP. *Theoret. Comput. Sci. 51*, 53–80.

WAGNER, K. W. 1990. Bounded Query Classes. *SIAM J. Comput. 19*, 5, 833–846.

Received December 2011; revised August 2012; accepted May 2013