Schema mapping illustration using universal examples

Butnariu, F.N.

Award date:
2015

Eindhoven University of Technology
Department of Mathematics and Computer Science
Web Engineering Research Group

Master's Thesis

Schema Mapping Illustration Using Universal Examples

Author: F.N. Butnariu

Supervisors: dr. G.H.L. Fletcher
dr. F. Mandreoli

Assessment Committee: dr. G.H.L. Fletcher
dr. S.P. Luttik
dr. F. Mandreoli
dr. M. Pechenizkiy

August 31, 2015

Abstract

Data exchange is the problem of transforming data from a source database into data that adheres to the schema of a target database. This transformation is formally captured by a schema mapping, which consists of a set of rules that precisely establish the relationship between the source and target database schemas. Here the rules in question are source-to-target tuple generating dependencies (s-t tgds), and the focus is on classes of s-t tgds such as LAV, strict LAV, n-modular, and self-join-free on the source.

Schema mappings intended for realistic databases can be intricate and therefore difficult to understand, but data examples have emerged as a promising means to facilitate schema mapping understanding. Although the semantic description of schema mappings can be infinite in terms of data examples, a particular type of data example known as universal example can reveal finite such semantic descriptions.

This work has investigated the problem of constructing universal examples for schema mappings with a finite semantic description. Although finite, such descriptions can still be large enough in terms of the number of universal examples they include, and a main concern in this work has been the reduction of this space of universal examples. The first contribution of this work is a generator of synthetic universal examples for schema mappings specified by LAV and n-modular s-t tgds. The former class encompasses strict LAV s-t tgds, whereas the latter class also includes s-t tgds that are self-join-free on the source. Universal examples are synthetic in the sense that they are not generated using existing data. In addition, a result in terms of the number of universal examples needed to describe LAV s-t tgds is given. The second contribution is a generator of real universal examples for schema mappings specified by LAV s-t tgds. In this case, universal examples are real in the sense that they are generated using existing data rather than being synthetically constructed.

Contents

Chapter 1

Introduction

This work is set in the context of restructuring data between heterogeneous relational databases. In such a scenario data residing in a source database and conforming to a source schema undergoes a transformation such that it complies with a target schema of a different database. This transformation is known as data exchange.

Data exchange inherently involves a mapping from the schema of a source database to the schema of a target database. A schema mapping defines a formal link between the source schema and the target schema and captures the intended data exchange semantics.

The input and output of performing data exchange according to a schema mapping is an instance of the source data and a corresponding instance of the target data. This pair of instances is known as a data example. Naturally, the source and target instances of a data example comply with the source and target schemas, respectively. Essentially, a data example reveals schema mapping behavior in much the same manner as test input reveals execution paths in a computer program. For this reason, data examples can be used to validate, convey, and improve the quality of schema mappings. In other words, data examples can be employed to verify and understand the data exchange semantics and to convey this understanding to different stakeholders.

Problem Statement

Schema mappings designed for realistic database schemas can be complex and difficult to understand, since the formal specifications of such mappings include intricate rules for migrating data between schemas. This problem, in effect, hinders the understanding of the data exchange semantics. Nevertheless, data examples can facilitate the comprehension of schema mappings, and one particular type of data example, known as universal example, stands out in illustrating schema mapping behavior. The semantic description of a schema mapping can be infinite in terms of data examples, that is, the schema mapping can be semantically identified with an infinite set of data examples. However, at the same time the semantic description of the schema mapping can be finite in terms of universal examples. In fact, there are a number of schema mapping classes that can be semantically identified with finite sets of universal examples.

In this work the construction of universal examples for schema mappings that have a finite semantic description in terms of universal examples is examined. Such universal examples illustrate and clarify the schema mappings. The construction of universal examples is cast as two separate, yet similar, problems. First, given a schema mapping, the construction of synthetic universal examples with artificial data is investigated. Second, given a schema mapping and a source instance, the construction of real universal examples with data originating from the source instance is examined. The schema mapping classes considered are LAV, strict LAV, n-modular, and self-join-free on the source, which are known to have finite semantic descriptions in terms of universal examples.

Contributions

This work introduces two generators of synthetic and real universal examples, respectively, for several classes of schema mappings. The first generator supports LAV, strict LAV, n-modular, and self-join-free on the source schema mappings, whereas the second generator supports LAV and strict LAV schema mappings. Often the space of universal examples that illustrate a schema mapping is large, and a main concern in development of these generators has been the reduction of this space and selection of only representative universal examples. Along the way, we have established a result with respect to the number of universal examples needed to illustrate LAV schema mappings. Furthermore, a prototype implementation of these generators has been built.

Thesis outline

The second chapter gives background information on the problem investigated here by formally introducing concepts used in subsequent chapters and by reporting on the related work done with respect to schema mapping illustration using universal examples.

The third chapter examines the problem of constructing illustrative but synthetic universal examples for several schema mapping classes such as LAV, strict LAV, n-modular, and self-join-free on the source. The universal examples are synthetic in the sense that they are constructed using artificial data.

The fourth chapter introduces two approaches to tackle the problem of constructing real universal examples for LAV and strict LAV schema mappings. First, an eager approach attempts to generate all universal examples that illustrate a given schema mapping in one go. Second, a lazy approach constructs one universal example at a time as the need for it arises. In both approaches the universal examples are generated with data retrieved from a given source database as opposed to the construction of synthetic universal examples.

The fifth chapter discusses a prototype implementation of the approaches to generate synthetic and real universal examples. The prototype implementation has been used to investigate a number of case studies that are also reported in this chapter.

The sixth chapter concludes the thesis by summarizing the contributions and giving directions for future work.

Acknowledgements

First and foremost, I would like to express my gratitude to Dr. Fletcher for his invaluable and continuous guidance throughout the course of this work. I also wish to thank Dr. Mandreoli from University of Modena and Reggio Emilia for her useful comments on an early draft of this thesis. Last but not least, I would like to thank Dr. Luttik and Dr. Pechenizkiy for reviewing the thesis.

Chapter 2

Background

2.1 Preliminaries

In this section some preliminary notions are given. First, basic concepts of database theory such as schema and instance are defined. Second, standard constraints used in data exchange known as source-to-target tuple generating dependencies are defined, and two types of such constraints are described, namely LAV and strict LAV. Third, the data exchange problem is defined along with its key notions of schema mapping and solution. Schema mappings consist of the standard constraints mentioned earlier, and solutions are revealed to have a particular type called universal solution. Furthermore, the concept of data example is defined along with its specific type known as universal example. Lastly, in addition to LAV and strict LAV, two other types of constraints are described, namely n-modular and self-join-free on the source.

The following definitions are given in the context of the relational model. Unless stated otherwise, these definitions are from [2].

2.1.1 Schemas and instances

Relation A *relation* is a finite set of n-tuples, each of which represents a relationship between n values [8].

Schema A *schema* $\mathbf{R}$ is a finite sequence $(R_1, ..., R_k)$ of relation symbols, each of a fixed arity.

Instance An *instance* I over $\mathbf{R}$ is a sequence $(R_1, ..., R_k)$, where each R_i is a relation, for $1 \leq i \leq k$. R_i denotes both the relation symbol and the relation that interprets the symbol. The *active domain* of an instance I, denoted by $adom(I)$, is the set of all values occurring in I.

Fact A *fact* of an instance I over $\mathbf{R}$ is an expression $P(v_1, ..., v_n)$, where P is a relation symbol in $\mathbf{R}$ and $v_1, ..., v_n$ are values such that tuple $(v_1, ..., v_n)$ belongs to relation P.

2.1.2 Source-to-target tuple generating dependencies

Atom An *atom* over $\mathbf{R}$ is an expression $P(x_1, ..., x_n)$, where P is a relation symbol in $\mathbf{R}$ and $x_1, ..., x_n$ are variables, not necessarily distinct.

S-t tgd A *s-t tgd (source-to-target tuple generating dependency)* or *GLAV (global-and-local-as-view) constraint* is a first-order sentence of the form

$$\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})),$$

where $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables, $\varphi(\mathbf{x})$ is a conjunction of atoms over a source schema $\mathbf{S}$ such that each variable in $\mathbf{x}$ occurs in at least one atom in $\varphi(\mathbf{x})$, and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over a target schema $\mathbf{T}$ with variables in $\mathbf{x}$ and $\mathbf{y}$. For notational simplicity, the universal quantifiers in front of the s-t tgds will be dropped in the rest of this thesis.

LAV s-t tgd A *LAV (local-as-view) s-t tgd* is a first-order sentence in which the left-hand side is a single atom, i.e. it is of the form

$$Q(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}),$$

where $Q(\mathbf{x})$ is an atom over a source schema $\mathbf{S}$.

Strict LAV s-t tgd A LAV s-t tgd $Q(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ is *strict* if the atom $Q(\mathbf{x})$ contains no repeated variables.

2.1.3 Data exchange and data examples

Schema mapping A *schema mapping* is a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ consisting of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, and a set Σ of s-t tgds. Such a schema mapping may also be referred to as a *GLAV schema mapping*. If Σ is a finite set of LAV s-t tgds, then $\mathcal{M}$ may be referred to as a *LAV schema mapping*. Similarly, if Σ is a finite set of strict LAV s-t tgds, then $\mathcal{M}$ may be referred to as a *strict LAV schema mapping*.

Solution Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. If I is a source instance, then a *solution* for I with respect to (w.r.t.) $\mathcal{M}$ is a target instance J such that pair (I, J) satisfies every s-t tgd in Σ. This satisfiability is denoted by $(I, J) \models \Sigma$.

Data exchange Given a finite source instance I and a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, the *data exchange problem* is to find a target instance J such that $(I, J) \models \Sigma$. Originally, the data exchange problem was defined for schema mappings specified by both source-to-target and target dependencies [3]. The former ones can be expressed by tuple generating dependencies (tgds), while the latter ones can be conveyed by tgds as well as by equality generating dependencies (egds). In the context of this work, the data exchange problem considers only schema mappings specified by source-to-target tuple generating dependencies (s-t tgds).

Homomorphism Let J and J' be two instances over target schema $\mathbf{T}$. Furthermore, let $\underline{\text{Const}}$ be a fixed infinite set of constants, and let $\underline{\text{Var}}$ be a fixed infinite set of nulls such that $\underline{\text{Const}} \cap \underline{\text{Var}} = \emptyset$. A function h from $\underline{\text{Const}} \cup \underline{\text{Var}}$ to $\underline{\text{Const}} \cup \underline{\text{Var}}$ is a *homomorphism* from J to J' if for every $c \in \underline{\text{Const}}$, there is $h(c) = c$, and for every relation symbol R in $\mathbf{T}$ and every tuple $(a_1, ..., a_n) \in R^J$, there is $(h(a_1), ..., h(a_n)) \in R^{J'}$. R^J denotes a relation of instance J, whereas $R^{J'}$ denotes a relation of instance J'. A homomorphism from J to J' is denoted by $J \rightarrow J'$.

Universal solution Given a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a source instance I, a *universal solution* for I w.r.t. $\mathcal{M}$ is a solution J for I w.r.t. $\mathcal{M}$ such that for every solution J' for I w.r.t. $\mathcal{M}$, there is a homomorphism from J to J'.

Chase procedure If $\mathcal{M}$ is a schema mapping specified by s-t tgds, then the *chase procedure* can be used to produce, given a source instance I, a universal solution J for I [6].

Data example A *data example* is a pair (I, J) consisting of a source instance I and a target instance J. A *positive example* for $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a data example (I, J) such that $(I, J) \models \Sigma$. A *negative example* for $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a data example (I, J) such that $(I, J) \not\models \Sigma$.

Universal example Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping in which Σ is a finite set of s-t tgds. A data example (I, J) is a *universal example* for $\mathcal{M}$ if J is a universal solution for I w.r.t. $\mathcal{M}$.

2.1.4 Additional source-to-target tuple generating dependencies

n-modular s-t tgd A s-t tgd σ is *n-modular* if for every data example (I, J) such that $(I, J) \not\models \sigma$, there is a subinstance $I' \subseteq I$ such that $|adom(I')| \leq n$ and $(I', J) \not\models \sigma$. A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ in which Σ is a finite set of n-modular s-t tgds may be referred to as a *n-modular schema mapping*, where n is the maximum number of variables occurring on the left-hand side of the s-t tgds in Σ.

S-t tgd that is self-join-free on the source A s-t tgd $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ is *self-join-free on the source* if none of the relation symbols in $\varphi(\mathbf{x})$ is repeated. A schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ in which Σ is a finite set of s-t tgds that are self-join-free on the source may be referred to as a *self-join-free on the source schema mapping*.

2.2 Related work

Alexe et al. investigated in [2] the problem of explaining and understanding schema mappings through data examples by examining classes of GLAV schema mappings that can be uniquely characterized by finite sets of data examples.

A finite set $\mathcal{U}$ of universal examples *uniquely characterizes* schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ w.r.t. a class $\mathcal{C}$ of s-t tgds if for every finite set $\Sigma' \subseteq \mathcal{C}$ such that $\mathcal{U}$ is a set of universal examples for schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$, it is the case that Σ is logically equivalent to Σ', denoted by $\Sigma \equiv \Sigma'$. Informally, a finite set of data examples uniquely characterizes a schema mapping if there is, up to logical equivalence, only one schema mapping that is satisfied by the data examples in the finite set.

Several types of data examples were studied in terms of their goodness to illustrate schema mapping semantics. Among positive, negative, and universal examples, it was established that universal examples are more suitable for unique characterization of schema mappings. Positive examples were considered along with negative examples, since a GLAV schema mapping can be semantically identified with an infinite set of positive examples. It was shown that positive and negative examples can only uniquely characterize schema mappings whose source and target schemas contain relation symbols with at most one attribute. In fact, a LAV schema mapping whose source and target schemas contain a relation symbol with two attributes cannot be uniquely characterized by any finite set of positive and negative examples. However, such a LAV schema mapping can be uniquely characterized by a finite set of universal examples. Furthermore, in a universal example the target instance is a universal solution for the source instance w.r.t. the schema mapping, and the universal solution generalizes the entire space of solutions for the source instance. In addition, it was established that a uniquely characterizing set of universal examples for a GLAV schema mapping is an Armstrong basis[1], which is a relaxation of the notion of Armstrong database[2]. GLAV schema mappings rarely have an Armstrong database. As a matter of fact, a LAV schema mapping was shown to have an Armstrong basis, but not an Armstrong database. In other words, the LAV schema mapping can only be uniquely characterized by more than one universal example. Nevertheless, there is a finite number of them. For all these reasons, universal examples were asserted to be more appropriate for unique characterization of schema mappings.

Not all GLAV schema mappings can be uniquely characterized by finite sets of universal examples. For the purpose of this work, the following results from [2] are relevant.

Theorem 5.1 If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping specified by a finite set of LAV s-t tgds, then there is a finite set $\mathcal{U}$ of universal examples for $\mathcal{M}$ such that $\mathcal{U}$ uniquely characterizes $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds.

As an example, consider LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{P\}$, $\mathbf{T} = \{Q\}$, and Σ consists of the single s-t tgd $P(x, y) \rightarrow Q(x)$. $\mathcal{M}$ is uniquely characterized w.r.t. the class of all LAV s-t tgds by the finite set of universal examples $\mathcal{U} = \{(I_1, J_1), (I_2, J_2)\}$, where source instances I_1 and I_2 along with target instances J_1 and J_2 are as follows:

$$I_1 = \{P(a, a)\} \qquad J_1 = \{Q(a)\}$$
$$I_2 = \{P(a, b)\} \qquad J_2 = \{Q(a)\}$$

Proposition 5.7 If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping specified by a finite set of strict LAV s-t tgds, then there is a finite set $\mathcal{U}$ of universal examples for $\mathcal{M}$ such that $\mathcal{U}$ uniquely characterizes $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds.

[1] Let Σ and $\mathcal{C}$ be two sets of s-t tgds. An *Armstrong basis* for Σ w.r.t. $\mathcal{C}$ is a finite set $\mathcal{U} = \{(I_1, J_1), ..., (I_n, J_n)\}$ such that (I_i, J_i) satisfies all the dependencies in $\mathcal{C}$ that are logically implied by Σ, and no other dependencies in $\mathcal{C}$, for $1 \leq i \leq n$. An Armstrong basis for $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ w.r.t. $\mathcal{C}$ is an Armstrong basis for Σ w.r.t. $\mathcal{C}$.

[2] Let Σ and $\mathcal{C}$ be two sets of s-t tgds. An *Armstrong database* for Σ w.r.t. $\mathcal{C}$ is a data example (I, J) such that (I, J) satisfies all the dependencies in $\mathcal{C}$ that are logically implied by Σ, and no other dependencies in $\mathcal{C}$. An Armstrong database for $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ w.r.t. $\mathcal{C}$ is an Armstrong database for Σ w.r.t. $\mathcal{C}$.

Referring back to the previous example, $\mathcal{M}$ is uniquely characterized w.r.t. the class of all strict LAV s-t tgds by the finite set of universal examples $\mathcal{U} = \{(I, J)\}$, where $I = \{P(a, b)\}$ and $J = \{Q(a)\}$.

Theorem 5.9 Let n be a positive integer and let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping, where Σ is a finite set of s-t tgds. If $\mathcal{M}$ is n-modular, then there is a finite set $\mathcal{U}$ of universal examples such that $\mathcal{U}$ uniquely characterizes $\mathcal{M}$ w.r.t. the class of all m-modular s-t tgds, where $m \geq n$.

Corollary 5.11 Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping where Σ is a finite set of s-t tgds that are self-join-free on the source. Then there is a finite set $\mathcal{U}$ of universal example such that $\mathcal{U}$ uniquely characterizes $\mathcal{M}$ w.r.t. the class of all s-t tgds that are self-join-free on the source.

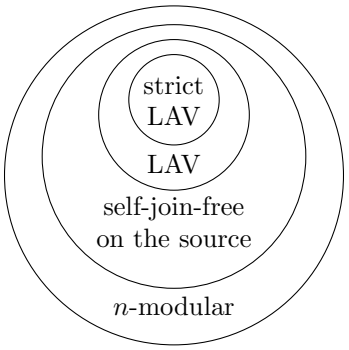


Figure 2.1: Classes of s-t tgds.

Figure 2.1 shows relatedness of the classes of s-t tgds considered here, which are not mutually exclusive.

In addition, the following schema mapping classes were shown to be uniquely characterizable by a finite set of universal examples w.r.t. a specific class of s-t tgds: (i) GAV schema mappings specified by the binary copy s-t tgd w.r.t. the class of GAV s-t tgds; (ii) GAV c-acyclic schema mappings w.r.t. the class of GAV s-t tgds; (iii) GAV schema mappings, where every s-t tgd has the property that all its variables are exported to its right-hand side, w.r.t. the class of GAV s-t tgds; and (iv) schema mappings specified by the s-t tgd $E(x, y) \rightarrow Q^H$, where E is the only relation symbol of the source schema and Q^H is a Boolean conjunctive query associated with an arbitrary instance over the target schema, w.r.t. the class of all s-t tgds. Moreover, it was asserted that for a c-acyclic schema mapping the uniquely characterizing set of universal examples can be effectively computed. The interested reader is referred to [2].

Chapter 3

Synthetic universal example generation

This chapter examines the problem of constructing synthetic universal examples for schema mappings belonging to one of the following classes: LAV, strict LAV, and n-modular, where the last-mentioned class also includes self-join-free on the source schema mappings. Since the expected input for this problem is only a schema mapping and no source instance, the universal examples are synthetic in the sense that they are constructed using artificial data as opposed to being created using data retrieved from a real source instance.

3.1 LAV schema mappings

In the proof of Theorem 5.1 from [2] were outlined some steps on constructing a finite set $\mathcal{U}$ of universal examples that uniquely characterize a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ w.r.t. the class of all LAV s-t tgds. Initially, a set D of distinct elements is defined, where $|D|$ is the maximum k among the arities $r_1, ..., r_s$ of all relations $R_1, ..., R_s$ in source schema $\mathbf{S}$. For each relation R_i, where $1 \leq i \leq s$, and for each possible tuple $\mathbf{d}_j$ of r_i elements from D, where $1 \leq j \leq k^{r_i}$, a universal example (I, J) is constructed, where source instance I consists of the single tuple $\mathbf{d}_j$ and target instance J consists of the result of chasing I using $\mathcal{M}$. Finally, each such universal example[1] is included in the set $\mathcal{U}$ that uniquely characterizes the LAV schema mapping $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds. By following this crude approach, the number of resulting universal examples is $|\mathcal{U}| = k^{r_1} + ... + k^{r_s}$.

Many of these universal examples are, however, redundant because their source instances are isomorphic. Since each source instance holds a single tuple, isomorphism between such source instances amounts to isomorphism between tuples.

Example 1. Consider the LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{R\}$, $\mathbf{T} = \{P, Q\}$, and Σ consists of the s-t tgd $R(x_1, x_2, x_3) \rightarrow P(x_1, x_2) \wedge Q(x_2, x_3)$.

Following the steps given above on constructing a finite set $\mathcal{U}$ of universal examples, there are 3^3 possible tuples of 3 elements from $D = \{a, b, c\}$, which result in a number of 3^3 potential universal examples. Consider tuples (a, a, a), (b, b, b), and (c, c, c), which lead to construction of universal examples (I, J), (I', J') and (I'', J''), respectively. Only one of these universal examples is enough to illustrate this particular mapping behavior, while the rest of them are redundant. This redundancy can be observed even before constructing target instances in that it is exposed by isomorphic source instances and inherently by isomorphic tuples.

$$I = \{R(a, a, a)\} \qquad J = \{P(a, a), Q(a, a)\}$$
$$I' = \{R(b, b, b)\} \qquad J' = \{P(b, b), Q(b, b)\}$$
$$I'' = \{R(c, c, c)\} \qquad J'' = \{P(c, c), Q(c, c)\}$$

[1] If source instance I of universal example (I, J) consists of a single tuple, then (I, J) is a *single-tuple-source example* or *sts example* [2].

In what follows, an algorithm is given, which generates no more than the necessary and sufficient universal examples to illustrate a LAV schema mapping. Out of finitely many universal examples that characterize the schema mapping, the algorithm yields a reduced number of them based on results concerning isomorphism between tuples.

Algorithm 1 basically follows the steps given above on constructing universal examples for a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. The algorithm relies on a function *NonIsomorphicTuples* that generates tuples in the size of arity r consisting of possibly repeated elements from set D (lines 16 to 38). However, instead of generating all possible r-tuples (or so-called permutations with repetitions), the aforementioned function efficiently generates only nonisomorphic r-tuples that are sufficient to construct only the necessary universal examples. Initially, a tuple $\mathbf{d}$ consisting of only one element from set D is constructed (line 17). For this tuple, $c_{\mathbf{d}}$ indicates the number of distinct tuple elements (line 18). In subsequent steps, a new tuple $\mathbf{d}'$ is constructed by appending the previous tuple $\mathbf{d}$ to one element d_j from set D (line 25). For this new tuple, the number $c_{\mathbf{d}'}$ of distinct tuple elements is based on the number $c_{\mathbf{d}}$ of distinct elements of the appended tuple $\mathbf{d}$. Essentially, if $\mathbf{d}$ already contained element d_j, then $c_{\mathbf{d}'}$ is the same as $c_{\mathbf{d}}$ (line 27). Otherwise, $c_{\mathbf{d}'}$ is $c_{\mathbf{d}}$ incremented by one (line 29).

It should be noted that function *MaxArity* returns the maximum among the arities of all relations in the source schema $\mathbf{S}$ (line 2), whereas function *Arity* returns the arity of relation R (line 6).

Algorithm 1 generates only the necessary universal examples based on several results that are introduced in what follows. These results have been uncovered by studying isomorphism between tuples.

3.1.1 The number of universal examples needed to illustrate LAV s-t tgds

In [2] was revealed that the number of universal examples that uniquely characterize a LAV schema mapping w.r.t. the class of all LAV s-t tgds is generally exponential in the size of the schema mapping. The number of variables occurring on the left-hand side of a s-t tgd that specifies a LAV schema mapping directly impacts the number of universal examples that characterize the schema mapping. In this section a precise indication of the number of universal examples needed to illustrate a LAV s-t tgd is given.

Consider a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ specified by the LAV s-t tgd $Q(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $x_1, x_2, ..., x_n$ are not necessarily distinct.

Taking into account that $D = \{d_1, d_2, ..., d_n\}$, relation Q can have n^n possible tuples of n elements from D, although many of them are isomorphic. These tuples can be arranged into groups such that any two tuples in the same group are isomorphic. Across all tuples in a group, the number of distinct elements is the same, although elements may differ from one tuple to another inside the same group. For instance, the group of tuples with one distinct element, denoted by $\mathcal{G}^1$, includes $(d_1, d_1, ..., d_1), (d_2, d_2, ..., d_2), ..., (d_n, d_n, ..., d_n)$, and the group of tuples with n distinct elements, denoted by $\mathcal{G}^n$, includes a tuple for each permutation of set D. Groups $\mathcal{G}^1$ and $\mathcal{G}^n$ are the most trivial ones and represent the extremes enclosing the rest of the groups. The number of tuples in a group $\mathcal{G}^m$ is given by

$$|\mathcal{G}^m| = \frac{n!}{(n-m)!}, \tag{3.1}$$

where $1 \leq m \leq n$.

It should be noted that the number of groups $\mathcal{G}^1, ..., \mathcal{G}^n$ is not equal to n. Although only one group $\mathcal{G}^1$ can exist, there are more than one group $\mathcal{G}^2$. Likewise, although only one group $\mathcal{G}^n$ can exist, there are more than one group $\mathcal{G}^{n-1}$. Furthermore, any two tuples $t_1 \in \mathcal{G}_1^m$ and $t_2 \in \mathcal{G}_2^m$ are not isomorphic, even if groups $\mathcal{G}_1^m$ and $\mathcal{G}_2^m$ share the same number m of distinct elements across their tuples. The number of groups sharing the same number m of distinct elements across their tuples is given by

$$\mathcal{S}(n, m) = \frac{1}{m!} \sum_{k=0}^{m} (-1)^{m-k} \frac{m!}{k!(m-k)!} k^n, \tag{3.2}$$

where $1 \leq m \leq n$. In combinatorics, $\mathcal{S}(n, m)$ is known as a Stirling number of the second kind [9].

Substituting $m = 2$ in equation (3.2) yields $\mathcal{S}(n, 2) = 2^{n-1} - 1$, which was used in the proof of Theorem 5.5 from [2] to show an exponential lower bound on the number of tuples across all instances in a uniquely characterizing set $\mathcal{U}$ of universal examples. Since there are $2^{n-1} - 1$ groups of tuples with two distinct constants and any two tuples in the same group are isomorphic, it suffices to arbitrarily choose one tuple from each group and therefore to have $2^{n-1} - 1$ nonisomorphic tuples with two distinct

Algorithm 1 Synthetic universal example generation for LAV schema mappings.

function LavUniversalExamples($\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$)
  $k \leftarrow$ MaxArity($\mathbf{S}$)
  $D \leftarrow \{d_1, ..., d_k\}$
  $\mathcal{U} \leftarrow \emptyset$
  for all $R \in \mathbf{S}$ **do**
    $r \leftarrow$ Arity(R)
    $T \leftarrow$ NonIsomorphicTuples(D, r)
    for all $\mathbf{d} \in T$ **do**
      $I \leftarrow \{R(\mathbf{d})\}$
      $J \leftarrow$ Chase($I, \mathcal{M}$)
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(I, J)\}$
    end for
  end for
  return $\mathcal{U}$
end function

function NonIsomorphicTuples($D = \{d_1, ..., d_k\}$, r)
  $\mathbf{d} \leftarrow (d_1)$
  $c_{\mathbf{d}} \leftarrow 1$
  $T \leftarrow \{\mathbf{d}\}$
  $i \leftarrow 1$
  while $i < r$ **do**
    $T' \leftarrow \emptyset$
    for all $\mathbf{d} \in T, \mathbf{d} = (e_1, ..., e_i)$ **do**
      for $j \leftarrow 1, c_{\mathbf{d}} + 1$ **do**
        $\mathbf{d}' \leftarrow (d_j, e_1, ..., e_i)$
        if $j \leq c_{\mathbf{d}}$ **then**
          $c_{\mathbf{d}'} \leftarrow c_{\mathbf{d}}$
        else
          $c_{\mathbf{d}'} \leftarrow c_{\mathbf{d}} + 1$
        end if
        $T' \leftarrow T' \cup \{\mathbf{d}'\}$
      end for
    end for
    $i \leftarrow i + 1$
    $T \leftarrow T'$
  end while
  return T
end function

constants. Doubling this number of nonisomorphic tuples yields $2^n - 2$, which is exactly the exponential lower bound and is consistent with the assumption that target instances have at least as many tuples as source instances.

Since only one tuple is chosen from each group to construct a source instance, the number of groups is an indication of the number of universal examples needed to illustrate the LAV s-t tgd. The number of groups and therefore the number of universal examples is given by

$$|\mathcal{U}| = \sum_{m=1}^{n} \mathcal{S}(n, m), \tag{3.3}$$

which is a sum of Stirling numbers of the second kind known in combinatorics as a Bell number [9]. Here $|\mathcal{U}|$ denotes the number of universal examples needed to illustrate a single LAV s-t tgd.

Table 3.1 shows $\mathcal{S}(n, m)$, for $1 \leq n \leq 7$ and for $1 \leq m \leq n$. Each table row corresponds to a LAV s-t tgd of the form $Q(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ with n variables on the left-hand side. For instance, the s-t tgd with three variables on the left-hand side has one group of tuples with one distinct element (as indicated by $\mathcal{S}(3, 1)$), three groups of tuples with two distinct elements (as indicated by $\mathcal{S}(3, 2)$), and one group of tuples with three distinct elements (as indicated by $\mathcal{S}(3, 3)$). Such a s-t tgd is further discussed in Example 2. It can be seen throughout the table that both $\mathcal{S}(n, 1)$ and $\mathcal{S}(n, n)$ (on the main diagonal) are equal to 1 regardless of n. Furthermore, table column $\mathcal{S}(n, 2)$ relates to the exponential lower bound uncovered in [2] on the number of tuples across all instances in a uniquely characterizing set $\mathcal{U}$ of universal examples.

n	$\lvert\mathcal{U}\rvert$	$\mathcal{S}(n,1)$	$\mathcal{S}(n,2)$	$\mathcal{S}(n,3)$	$\mathcal{S}(n,4)$	$\mathcal{S}(n,5)$	$\mathcal{S}(n,6)$	$\mathcal{S}(n,7)$
1	1	1						
2	2	1	1					
3	5	1	3	1				
4	15	1	7	6	1			
5	52	1	15	25	10	1		
6	203	1	31	90	65	15	1	
7	877	1	63	301	350	140	21	1

Table 3.1: The triangle of Stirling numbers of the second kind.

The general LAV s-t tgd $Q(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ considered so far includes two specific cases. First, the LAV s-t tgd can have exactly one repeated variable on the left-hand side such that $\mathbf{x} = (x, x, ..., x)$. Such a s-t tgd is further discussed in Example 3. Second, the LAV s-t tgd can have no repeated variables on the left-hand side and therefore can be strict. Group $\mathcal{G}^1$ discussed earlier in this section relates to the first case, whereas group $\mathcal{G}^n$ relates to the second case. Furthermore, in Table 3.1, column $\mathcal{S}(n, 1)$ is representative for the first case, while the main diagonal is representative for the second case. These two specific cases of LAV s-t tgds do not conform to the lower exponential bound uncovered in [2] on the number of tuples across all instances in a uniquely characterizing set $\mathcal{U}$ of universal examples.

Example 2. Consider the LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{R\}$, $\mathbf{T} = \{P\}$, and Σ consists of the single ternary copy s-t tgd $R(x_1, x_2, x_3) \rightarrow P(x_1, x_2, x_3)$. This example shows how the space of finitely many universal examples that characterize $\mathcal{M}$ can be reduced by considering isomorphism between tuples.

Following the steps given in Section 3.1 on constructing a finite set $\mathcal{U}$ of universal examples, there are 3^3 possible tuples of 3 elements from $D = \{a, b, c\}$, and each one of them represents a potential universal example. These tuples are grouped as follows:

$\mathcal{G}^1$	$\mathcal{G}_1^2$	$\mathcal{G}_2^2$	$\mathcal{G}_3^2$	$\mathcal{G}^3$
(a, a, a)	(b, a, a)	(a, b, a)	(b, b, a)	(c, b, a)
(b, b, b)	(c, a, a)	(a, c, a)	(c, c, a)	(b, c, a)
(c, c, c)	(a, b, b)	(b, a, b)	(a, a, b)	(c, a, b)
	(c, b, b)	(b, c, b)	(c, c, b)	(a, c, b)
	(a, c, c)	(c, a, c)	(a, a, c)	(b, a, c)
	(b, c, c)	(c, b, c)	(b, b, c)	(a, b, c)

Group $\mathcal{G}^1$ consists of tuples with exactly one element from D, whereas group $\mathcal{G}^3$ consists of tuples with all elements from D. The remaining groups consist of tuples with exactly two distinct elements from D, one of which is repeated. In group $\mathcal{G}_1^2$ an element is repeated on the last two positions of each tuple. In group $\mathcal{G}_2^2$ an element is repeated on the first and last positions of each tuple. Finally, in group $\mathcal{G}_3^2$ an element is repeated on the first two positions of each tuple. Any two tuples in the same group are in fact isomorphic. The space of tuples to consider for universal example construction can be reduced to nonisomorphic tuples and therefore only one arbitrary tuple from each group can be considered for universal example construction.

Taking into account only the first tuple of each group, the following source instances I_i can be obtained, where $1 \leq i \leq 5$. Furthermore, chasing each I_i results in a corresponding target instance J_i.

$$\begin{aligned} I_1 &= \{R(a,a,a)\} & J_1 &= \{P(a,a,a)\} \\ I_2 &= \{R(b,a,a)\} & J_2 &= \{P(b,a,a)\} \\ I_3 &= \{R(a,b,a)\} & J_3 &= \{P(a,b,a)\} \\ I_4 &= \{R(b,b,a)\} & J_4 &= \{P(b,b,a)\} \\ I_5 &= \{R(c,b,a)\} & J_5 &= \{P(c,b,a)\} \end{aligned}$$

Finally, the finite set of universal examples is $\mathcal{U} = \{(I_i, J_i) | 1 \leq i \leq 5\}$.

Example 3. Consider a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{R\}$, $\mathbf{T} = \{P\}$, and Σ consists of the single ternary copy s-t tgd $R(x,x,x) \rightarrow P(x,x,x)$ with exactly one repeated variable on the left-hand side.

Following the steps given in Section 3.1 on constructing a finite set $\mathcal{U}$ of universal examples, there are 3^3 possible tuples of 3 elements from $D = \{a,b,c\}$. However, only 3 tuples consist of exactly one repeated element, namely (a,a,a), (b,b,b), and (c,c,c). Since these tuples are isomorphic, any one of them is sufficient to construct a single universal example. Taking into account the first tuple, the set that uniquely characterizes $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds is $\mathcal{U} = \{(I,J)\}$, where $I = \{R(a,a,a)\}$ and $J = \{P(a,a,a)\}$.

It should be noted that the algorithm given in Section 3.1 yields four additional universal examples whose target instance is empty. These universal examples reveal no more than disallowed mapping behavior.

$$\begin{aligned} I_2 &= \{R(b,a,a)\} & J_2 &= \emptyset \\ I_3 &= \{R(a,b,a)\} & J_3 &= \emptyset \\ I_4 &= \{R(b,b,a)\} & J_4 &= \emptyset \\ I_5 &= \{R(c,b,a)\} & J_5 &= \emptyset \end{aligned}$$

3.2 Strict LAV schema mappings

Algorithm 2 is a stripped-down version of Algorithm 1 tailored to construct universal examples for a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ specified by strict LAV s-t tgds. The algorithm basically constructs a single universal example (I,J) for every relation symbol $R \in \mathbf{S}$ such that source instance I holds a single r-tuple $\mathbf{d}$ consisting of distinct elements $d_1, ..., d_r$. This approach is consistent with the assertion made in [2] that the number of universal examples that uniquely characterize $\mathcal{M}$ is precisely the number of relation symbols in $\mathbf{S}$.

3.3 n-modular schema mappings

In the proof of Theorem 5.9 from [2] were outlined some steps on constructing a set $\mathcal{U}$ of universal examples that uniquely characterizes a n-modular schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \mathbf{\Sigma})$ w.r.t. the class of n-modular s-t tgds, where n is the maximum number of variables occurring on the left-hand side of the s-t tgds in Σ, as stated in the proof of Proposition 2.7 from [10]. These steps also apply to self-join-free on the source schema mappings.

Initially, a set D of distinct elements is defined, where $|D| = n$. Next, a set of all possible source instances having tuples of elements from D is constructed, where the cardinality of this set is k. For each

Algorithm 2 Synthetic universal example generation for strict LAV schema mappings.

```
1: function StrictLavUniversalExamples(M = (S, T, Σ))
2:     U ← ∅
3:     for all R ∈ S do
4:         r ← Arity(R)
5:         d ← (d_1, ..., d_r)
6:         I ← {R(d)}
7:         J ← Chase(I, M)
8:         U ← U ∪ {(I, J)}
9:     end for
10:    return U
11: end function
```

source instance I, a universal example (I, J) is constructed by chasing I using $\mathcal{M}$. Finally, each such universal example is included in the set $\mathcal{U}$ that uniquely characterizes the n-modular schema mapping $\mathcal{M}$ w.r.t. the class of n-modular s-t tgds, where $|\mathcal{U}| = k$.

Algorithm 3 follows the steps given above on constructing universal examples for a n-modular schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. For each s-t tgd σ of the form $\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$, the algorithm constructs universal examples in two steps. First, for each atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$, a power set $\mathcal{P}(T)_{R(\mathbf{x})}$ without the empty set is created (lines 7 to 13). Second, based on each power set $\mathcal{P}(T)_{R(\mathbf{x})}$, a number of k source instances are constructed and chased (lines 15 to 26).

Function *MaxLeftVariableCount* returns the maximum among the number of variables occurring on the left-hand side of the s-t tgds in Σ (line 2).

Function *Permute* generates so-called permutations with repetitions, which are essentially all possible tuples in the size of arity r consisting of possibly repeated elements from set D (line 9). This function returns set T of tuples whose cardinality is given by $|T| = |D|^r$. Given set T, a power set $\mathcal{P}(T)_{R(\mathbf{x})}$ corresponding to atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$ is created (line 10). This power set lacks the empty set, and the power set cardinality is given by $|\mathcal{P}(T)_{R(\mathbf{x})}| = 2^{|T|} - 1$.

As an example, consider set $D = \{a, b\}$ and atom $E(x, y)$, which can have a set of tuples

$$T = \{(a, a), (b, b), (a, b), (b, a)\}$$

and a corresponding power set

$$\begin{aligned}
\mathcal{P}(T)_{E(x,y)} = \big\{ & \{(a, a)\}, \\
& \{(b, b)\}, \\
& \{(a, b)\}, \\
& \{(b, a)\}, \\
& \{(a, a), (b, b)\}, \\
& \{(a, a), (a, b)\}, \\
& \{(a, a), (b, a)\}, \\
& \{(b, b), (a, b)\}, \\
& \{(b, b), (b, a)\}, \\
& \{(a, b), (b, a)\}, \\
& \{(a, a), (b, b), (a, b)\}, \\
& \{(a, a), (b, b), (b, a)\}, \\
& \{(a, a), (a, b), (b, a)\}, \\
& \{(b, b), (a, b), (b, a)\}, \\
& \{(a, a), (b, b), (a, b), (b, a)\}\big\}.
\end{aligned} \tag{3.4}$$

For reasons that will become clear later, for each atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$, a zero-based index $i_{R(\mathbf{x})}$ used for accessing a subset of power set $\mathcal{P}(T)_{R(\mathbf{x})}$ is initialized (line 11). Moreover, a product k between the cardinalities corresponding to each power set $\mathcal{P}(T)_{R(\mathbf{x})}$ is computed (line 12).

Product k is used to control a *while* loop that constructs a universal example (I, J) at each iteration (lines 15 to 26). In other words, k is the number of universal examples corresponding to s-t tgd σ. Source instance I of universal example (I, J) is a union of subsets from each power set $\mathcal{P}(T)_{R(\mathbf{x})}$. For this union, one subset of each power set $\mathcal{P}(T)_{R(\mathbf{x})}$ is considered.

As an example, consider the 2-modular s-t tgd $E(x, y) \wedge G(y, x) \rightarrow F(x, y)$, which is also self-join-free on the source. Since relation symbols E and G have the same arity, atoms $E(x, y)$ and $G(y, x)$ can share the power set given by equation (3.4) and therefore $\mathcal{P}(T)_{E(x,y)} = \mathcal{P}(T)_{G(y,x)}$. In this example, a source instance I is a union of a subset from $\mathcal{P}(T)_{E(x,y)}$ and a subset from $\mathcal{P}(T)_{G(y,x)}$. Moreover, multiple source instances can be constructed in this way. Figure 3.1 shows construction of source instances as a complete bipartite graph, where the two disjoint sets of vertices correspond to power sets $\mathcal{P}(T)_{E(x,y)}$ and $\mathcal{P}(T)_{G(y,x)}$, and each edge between a subset of one power set and a subset of the other power set represents a source instance consisting of the union of the two subsets.

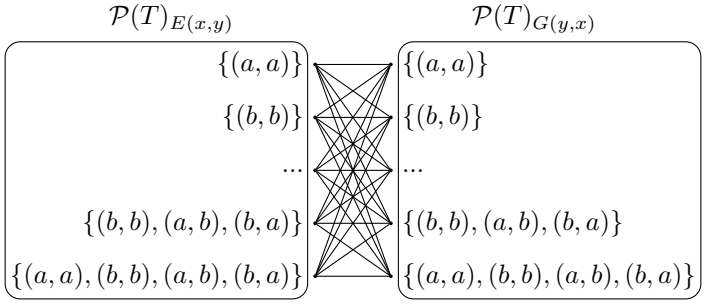


Figure 3.1: Construction of source instances for s-t tgd $E(x, y) \wedge G(y, x) \rightarrow F(x, y)$.

The number of possible source instances and therefore the number of possible universal examples is the product k between the cardinality of $\mathcal{P}(T)_{E(x,y)}$ and the cardinality of $\mathcal{P}(T)_{G(y,x)}$.

$$k = |\mathcal{P}(T)_{E(x,y)}| \cdot |\mathcal{P}(T)_{G(y,x)}|$$

Although product k controls the number of source instances to be constructed, an additional way is required to control which subset of each power set is considered for construction of a source instance I. This is where index $i_{R(\mathbf{x})}$ comes into play. At each *while* iteration, index $i_{R(\mathbf{x})}$ of each atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$ points to a subset of power set $\mathcal{P}(T)_{R(\mathbf{x})}$ (line 18).

Each index $i_{R(\mathbf{x})}$ can be seen as a digit of a number representation in a mixed-radix system[2], where $0 \leq i_{R(\mathbf{x})} < |\mathcal{P}(T)_{R(\mathbf{x})}|$. In other words, the mixed-radix number has as many digits as the number of atoms in $\varphi(\mathbf{x})$. Furthermore, each digit of the mixed-radix number has a corresponding base equal to $|\mathcal{P}(T)_{R(\mathbf{x})}|$.

The mixed-radix number shadows variable i used by the *while* loop. Both the mixed-radix number and variable i are initialized before the *while* loop and subsequently are repeatedly incremented inside the *while* loop. In fact, each digit of the mixed-radix number is initialized in the *for* loop preceding the *while* loop (line 11) and subsequently is incremented by procedure *Increment* (lines 30 to 39). Whenever a digit of the mixed-radix number reaches its upper bound, a carry operation is performed by reinitializing the digit and incrementing the next more significant digit[3].

[2]Using positional notation, a four-digit positive integer can be represented as $(a_3a_2a_1a_0)_b = a_3b^3 + a_2b^2 + a_1b^1 + a_0$, where a_i is a digit, $0 \leq i \leq 3$, and b is a base or radix. Using a mixed-radix system, a four-digit positive integer can be represented as $\begin{bmatrix} a_3, a_2, a_1, a_0 \\ b_3, b_2, b_1, b_0 \end{bmatrix} = a_3b_2b_1b_0 + a_2b_1b_0 + a_1b_0 + a_0$, where a_i is a digit, b_i is a base, and $0 \leq i \leq 3$. While in positional notation each digit has the same base, in a mixed-radix system each digit has its own base [5].

[3]Considering a four-digit positive integer $a_3a_2a_1a_0$, the most significant digit is a_3, whereas the least significant digit is a_0 [5].

Algorithm 3 Synthetic universal example generation for n-modular schema mappings.

function nModularUniversalExamples($\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$)
  $n \leftarrow$ MaxLeftVariableCount(Σ)
  $D \leftarrow \{d_1, ..., d_n\}$
  $\mathcal{U} \leftarrow \emptyset$
  for all $\sigma \in \Sigma$ **do**
    $k \leftarrow 1$
    for all $R(\mathbf{x}) \in \varphi(\mathbf{x})$ **do**
      $r \leftarrow$ Arity(R)
      $T \leftarrow$ Permute(D, r)
      $\mathcal{P}(T)_{R(\mathbf{x})} \leftarrow$ PowerSet(T)
      $i_{R(\mathbf{x})} \leftarrow 0$
      $k \leftarrow k * |\mathcal{P}(T)_{R(\mathbf{x})}|$
    end for
    $i \leftarrow 0$
    while $i < k$ **do**
      $I \leftarrow \emptyset$
      for all $R(\mathbf{x}) \in \varphi(\mathbf{x})$ **do**
        for all $\mathbf{d} \in$ Subset($\mathcal{P}(T)_{R(\mathbf{x})}, i_{R(\mathbf{x})}$) **do**
          $I \leftarrow I \cup \{R(\mathbf{d})\}$
        end for
      end for
      $J \leftarrow$ Chase($I, \mathcal{M}$)
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(I, J)\}$
      Increment($\varphi(\mathbf{x})$)
      $i \leftarrow i + 1$
    end while
  end for
  return $\mathcal{U}$
end function

procedure Increment($\varphi(\mathbf{x})$)
  for all $R(\mathbf{x}) \in \varphi(\mathbf{x})$ **do**
    if $i_{R(\mathbf{x})} < |\mathcal{P}(T)_{R(\mathbf{x})}| - 1$ **then**
      $i_{R(\mathbf{x})} \leftarrow i_{R(\mathbf{x})} + 1$
      break
    else
      $i_{R(\mathbf{x})} \leftarrow 0$
    end if
  end for
end procedure

3.3.1 Reductions on the number of universal examples

Algorithm 3 exhaustively constructs and chases all possible source instances in order to yield universal examples. In fact, the number of universal examples that illustrate a single n-modular s-t tgd is given by

$$|\mathcal{U}| = \prod_{R(\mathbf{x}) \in \varphi(\mathbf{x})} |\mathcal{P}(T)_{R(\mathbf{x})}|.$$

In what follows, some reductions of the space of source instances are explored. Consider the previous example consisting of the 2-modular s-t tgd $E(x, y) \wedge G(y, x) \rightarrow F(x, y)$, which is also self-join-free on the source. Recall that in Figure 3.1 construction of source instances is shown as a complete bipartite graph, where each edge between subsets of distinct power sets indicates a possible source instance. These subsets vary in the number of tuples they contain, but for the purpose of this discussion, the focus will be on subsets consisting of a single tuple. Figure 3.2 shows the portion of the complete bipartite graph which covers subsets with one tuple.

A first reduction arises from the fact that among the source instances constructed from these subsets there are some instances which yield empty target instances when being chased because of unsatisfied join conditions (e.g. $I = \{E(a, a), G(a, b)\}$). Such source instances do not contribute to construction of informative universal examples and therefore can be discarded. Such universal examples would reveal at most behavior disallowed by the schema mapping.

A second reduction is based on the observation that some source instances are isomorphic to other source instances and consequently are redundant (e.g. $I' = \{E(a, b), G(b, a)\}$ is isomorphic to $I'' = \{E(b, a), G(a, b)\}$). Such source instances lead to construction of unnecessary universal examples and consequently can be discarded.

These two reductions are applicable to the entire complete bipartite graph and not only to a portion of it. Some results concerning these reductions are discussed in Section 5.2.2.

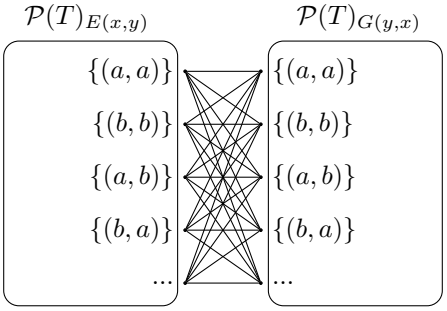


Figure 3.2: Portion of the complete bipartite graph from Figure 3.1.

Chapter 4

Real universal example generation

In this chapter the problem of constructing real universal examples for LAV and strict LAV schema mappings is examined. In addition to a schema mapping, the expected input for this problem also includes a source instance. Universal examples are created using data retrieved from this source instance as opposed to the approach of synthetically constructing universal examples.

The intuition behind the construction of real universal examples is to create universal examples that mimic their synthetic counterparts using real data. The construction of synthetic universal examples is conducted under the assumption that relation attributes share the same domain of values. However, the construction of real universal examples should be performed under the assumption that each relation attribute can have its own distinct domain. A relation is in fact a subset of a Cartesian product of a set of domains corresponding to each relation attribute, where a domain is a set of all possible attribute values [8].

4.1 LAV schema mappings

Two approaches to construct real universal examples for LAV schema mappings have been considered. On the one hand, an eager approach attempts to generate all universal examples in one go. On the other hand, a lazy approach generates one universal example at a time as the need for it arises.

4.1.1 Eager approach

Given a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a source instance I, the steps to construct real universal examples in an eager fashion are as follows:

First, groups of isomorphic synthetic tuples are generated for each relation R in $\mathbf{S}$. As seen before in Section 3.1, a set $D = \{d_1, ..., d_k\}$ of distinct elements is defined, where $|D|$ is the maximum arity k of all relations in $\mathbf{S}$. For each R in $\mathbf{S}$, where R has arity r, all possible r-tuples with elements from D are generated. These r-tuples for R are grouped such that any two tuples in the same group are isomorphic. Based on whether attribute domains of R are disjoint, some groups of tuples may be discarded.

Second, conjunctive queries to be executed on source instance I are generated based on the remaining groups of tuples, for every R in $\mathbf{S}$. Selecting one tuple from each remaining group results is a sequence of nonisomorphic tuples. Moreover, depending on the number of remaining groups and the number of tuples in each group, multiple sequences of nonisomorphic tuples can be possible. Nevertheless, this space of sequences may be reduced by considering yet again the constraints imposed by the attribute domains of R and the isomorphism between sequences. At the previous step some groups of tuples may have been discarded based on attribute domains of R. In that step the unit considered for establishing whether domain constraints are satisfied was a tuple, whereas the unit considered in this step is a sequence of tuples. Likewise, at the previous step it was reasoned about isomorphism in terms of tuples, whereas in this step isomorphism is considered in terms of sequences of tuples. After applying the aforementioned reductions, each remaining sequence of every R is translated into a conjunctive query that is executed on source instance I with the purpose of retrieving a sequence of tuples with real values, which mimics the initial synthetic sequence. Since each R can have multiple remaining synthetic sequences, this operation

stops as soon as one conjunctive query for every R is successful in retrieving a sequence of tuples with real values.

Third, universal examples $(I_1, J_1), ..., (I_n, J_n)$ are constructed using the retrieved sequence of tuples with real values, for every R in $\mathbf{S}$. Each source instance I_i holds a single tuple from the retrieved sequence, and each target instance J_i is the result of chasing I_i using $\mathcal{M}$, where $1 \leq i \leq n$.

Example 4. Consider the LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where

$$\mathbf{S} = \{location(area : varchar(32), latitude : float, longitude : float)\},$$
$$\mathbf{T} = \{coordinates(latitude : float, longitude : float)\},$$

and Σ consists of the single s-t tgd $location(a, lat, long) \rightarrow coordinates(lat, long)$. Furthermore, consider source instance

$$\begin{aligned} I = \{&location(\text{Europe}, 38.13, 15.82),\\ &location(\text{Africa}, 15.82, 15.82),\\ &location(\text{Africa}, 15.82, 38.13)\}. \end{aligned}$$

Below are all possible 3-tuples with elements from $D = \{a, b, c\}$ for relation *location*. These synthetic tuples are grouped such that any two tuples in the same group are isomorphic.

(1)	(2)	(3)	(4)	(5)
(a, a, a)	(b, a, a)	(a, b, a)	(b, b, a)	(c, b, a)
(b, b, b)	(c, a, a)	(a, c, a)	(c, c, a)	(b, c, a)
(c, c, c)	(a, b, b)	(b, a, b)	(a, a, b)	(c, a, b)
	(c, b, b)	(b, c, b)	(c, c, b)	(a, c, b)
	(a, c, c)	(c, a, c)	(a, a, c)	(b, a, c)
	(b, c, c)	(c, b, c)	(b, b, c)	(a, b, c)

Taking into account that attribute *area* of relation *location* is associated with domain $varchar(32)$ and both attributes *latitude* and *longitude* are associated with domain $float$, it can be inferred that tuples of groups (1), (3), and (4) do not comply with these domains.

It should be noted that in a different scenario where all attributes of relation *location* would be associated with disjoint domains, only group (5) would be retained. Such a scenario would be a best case, since the search space is reduced to only one group. Furthermore, in a scenario where all attributes of relation *location* would share the same domain, all groups would be retained. Such a scenario would be a worst case, since the search space would not undergo any reduction. Section 5.2.1 examines the worst case for a simple LAV schema mapping.

The remaining groups (2) and (5) represent a space of multiple possibilities in terms of selecting one tuple from each remaining group. There are 36 such possible sequences of tuples, but some of them do not comply with the aforementioned domains (e.g. sequence $(b, a, a)(c, b, a)$, where b cannot be both $varchar(32)$ and $float$), while others are isomorphic (e.g. $(b, a, a)(b, c, a)$ is isomorphic to $(a, b, b)(a, c, b)$). After discarding sequences that violate domain constraints, there are 12 remaining sequences. After discarding isomorphic sequences, only 2 sequences are left, namely $(b, a, a)(b, c, a)$ and $(b, a, a)(b, a, c)$. The former sequence is translated into a conjunctive query given in Listing 4.1, which yields no result when executed on source instance I. The latter sequence is translated into a conjunctive query given in Listing 4.2, which in turn retrieves sequence $(\text{Africa}, 15.82, 15.82)(\text{Africa}, 15.82, 38.13)$ from I as one composite tuple.

```
SELECT * FROM
  location AS location1,
  location AS location2
WHERE
  location1.area = location2.area AND
  location1.longitude = location2.longitude AND
  location1.latitude = location1.longitude AND
  location1.ctid <> location2.ctid
LIMIT 1;
```

Listing 4.1: Conjunctive query for $(b, a, a)(b, c, a)$.

```
SELECT * FROM
  location AS location1,
  location AS location2
WHERE
  location1.area = location2.area AND
  location1.latitude = location2.latitude AND
  location1.latitude = location1.longitute AND
  location1.ctid <> location2.ctid
LIMIT 1;
```

Listing 4.2: Conjunctive query for $(b, a, a)(b, a, c)$.

Both queries enforce conditions on attributes of relation *location*, but one particular attribute is not listed in source schema $\mathbf{S}$, namely *ctid*. This is a system attribute specific to PostgreSQL, and it is used here to uniquely identify a tuple within relation *location* for the purpose of disallowing repeated tuples in a retrieved sequence.

Finally, each tuple in the retrieved sequence from I contributes to construction of the following source and target instances, which form real universal examples.

$$I_1 = \{location(\text{Africa}, 15.82, 15.82)\} \qquad J_1 = \{coordinates(15.82, 15.82)\}$$
$$I_2 = \{location(\text{Africa}, 15.82, 38.13)\} \qquad J_2 = \{coordinates(15.82, 38.13)\}$$

In general, in case all conjunctive queries are unsuccessful in retrieving a sequence from I, a replacement sequence with real values from I can be constructed as an alternative. The replacement sequence would consist of tuples that are not actually present in I, but nevertheless it would contribute to construction of indicative universal examples.

4.1.2 Lazy approach

Given a LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a source instance I, the steps to construct real universal examples in a lazy fashion are as follows:

First, groups of isomorphic synthetic tuples are generated for each relation R in $\mathbf{S}$ in the same way as in the eager approach. Out of all these groups for R, only a number of them are retained based on attribute domains of R. The remaining groups should reflect whether attribute domains of R are disjoint or not.

Second, based on a single sequence of synthetic tuples chosen from the remaining groups for R, distinct queries to be executed on source instance I are generated. Each query is meant to retrieve a single tuple from I, which mimics a synthetic tuple of the sequence. In case a query cannot retrieve a tuple simply because I does not contain such a tuple, values from I are used to construct a replacement tuple that resembles its synthetic counterpart. It should be noted that while in the eager approach a sequences of synthetic tuples is translated into a single conjunctive query meant to retrieve data in one go, in this approach a sequence is translated into a number of distinct queries meant to retrieve data in more than one go.

Third, universal examples $(I_1, J_1), ..., (I_n, J_n)$ are constructed using the tuples of real values from the previous step, where each source instance I_i holds a single tuple, and each target instance J_i is the result of chasing I_i using $\mathcal{M}$, for $1 \leq i \leq n$.

Example 5. Consider the LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ from Example 4, where

$$\mathbf{S} = \{location(area : varchar(32), latitude : float, longitude : float)\},$$
$$\mathbf{T} = \{coordinates(latitude : float, longitude : float)\},$$

and Σ consists of the single s-t tgd $location(a, lat, long) \rightarrow coordinates(lat, long)$. Furthermore, consider source instance

$$I = \{location(\text{Europe}, 38.13, 15.82),$$
$$location(\text{Africa}, 15.82, 38.13)\}.$$

which is different from Example 4.

Out of all groups of possible synthetic tuples for relation *location*, only the following ones are retained based on domain constraints of the aforementioned relation, as seen before in Example 4.

(2)	(5)
(b, a, a)	(c, b, a)
(c, a, a)	(b, c, a)
(a, b, b)	(c, a, b)
(c, b, b)	(a, c, b)
(a, c, c)	(b, a, c)
(b, c, c)	(a, b, c)

From each of the remaining groups, only one tuple is deemed sufficient to be selected, since any two tuples in the same group are isomorphic. However, an arbitrary tuple choice can still lead to a violation of the domain constraints. For instance, tuples (b, a, a) and (c, b, a) from groups (2) and (5), respectively, do not comply with attribute domains of relation *location*. As a result, the remaining groups are explored until a compliant sequence of synthetic tuples is identified such as (b, a, a) and (b, c, a).

Next, the selected synthetic tuples are translated into distinct queries to be executed on source instance I. Tuple (b, a, a) becomes query

```
SELECT * FROM location WHERE latitude = longitude LIMIT 1;
```

while tuple (b, c, a) is translated into query

```
SELECT * FROM location LIMIT 1;
```

The former query returns no tuple of real values, whereas the latter query retrieves tuple (Europe, 38.13, 15.82) from I.

In general, in the lazy approach, tuples that are successfully retrieved from I will form a pool of real values that will be used to construct a replacement tuple, in case a query returns no tuple. In this example, tuple $(\text{Europe}, 15.82, 15.82)$ is created as a substitute for the missing tuple. Although such a replacement tuple does not convey a genuine fact from I, it does reveal behavior allowed by source schema $\mathbf{S}$.

In spite of the fact that the synthetic tuples exhibit a "relatedness" in the sense that they share some elements, the queries translated from these tuples are distinct and therefore can potentially retrieve unrelated tuples of real values from I. Nevertheless, this "relatedness" is employed in construction of replacement tuples when the need for such tuples arises. The choice of issuing distinct queries is motivated by the aim of constructing real universal examples on demand, that is, creating one real universal example at a time as the need for it arises. Moreover, it is preferable to expose the arrangement of possibly repeated elements inside a tuple of real values rather than the tuple relatedness to other tuples.

Finally, in this example, the source and target instances of the real universal examples are as follows:

$$I_1 = \{location(\text{Europe}, 38.13, 15.82)\} \qquad J_1 = \{coordinates(38.13, 15.82)\}$$
$$I_2 = \{location(\text{Europe}, 15.82, 15.82)\} \qquad J_2 = \{coordinates(15.82, 15.82)\}$$

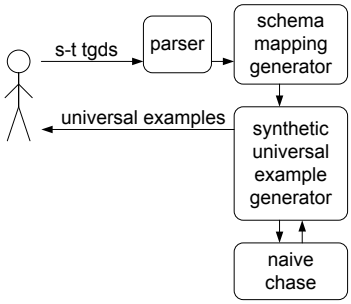
Chapter 5

Prototype implementation and case studies

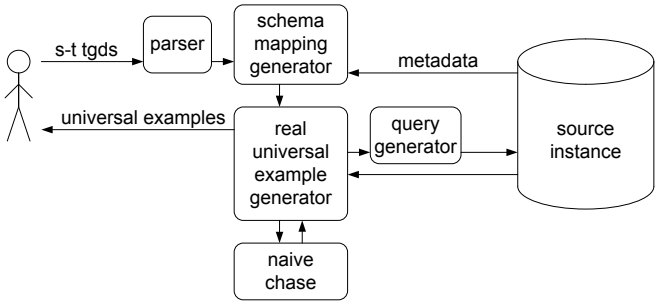
This chapter discusses a prototype implementation of the approaches to generate synthetic and real universal examples. This prototype implementation has been used to investigate a number of case studies that are also reported here.

5.1 Prototype implementation

A prototype implementation that generates synthetic and real universal examples has been developed in Java. The synthetic generator supports LAV, strict LAV, and n-modular schema mappings (including self-join-free on the source). The real generator supports LAV schema mappings (including strict LAV).



(a) Synthetic universal example generation.

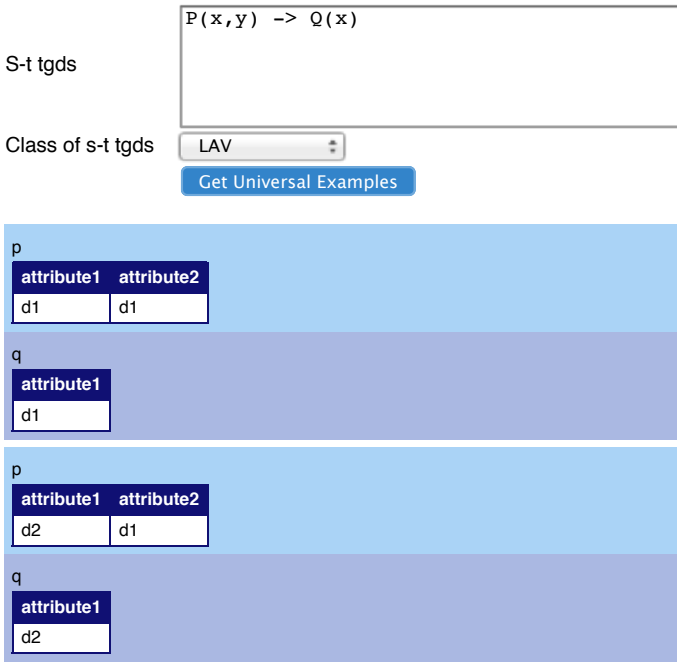


(b) Real universal example generation.

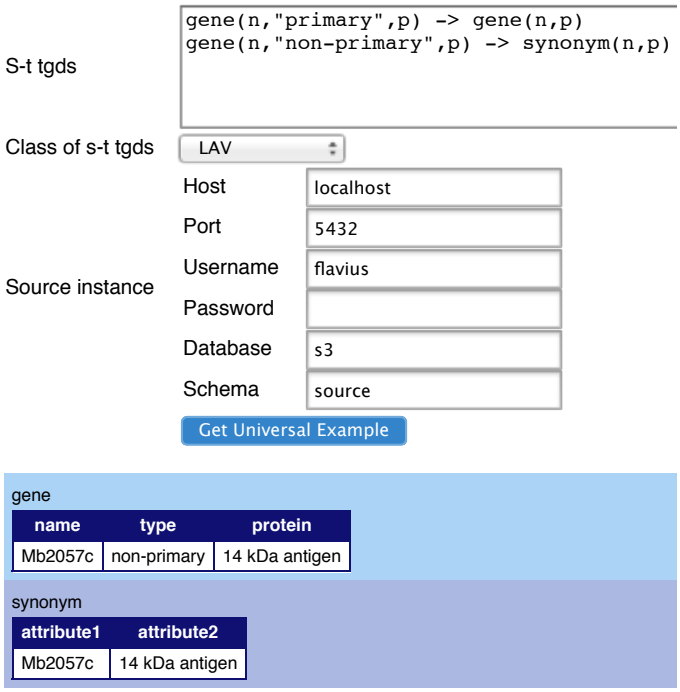
Figure 5.1: Synthetic and real universal example generation workflows.

Figure 5.1a shows the typical workflow of generating synthetic universal examples. Initially, a user is expected to provide s-t tgds that specify a schema mapping. The given s-t tgds are parsed, and a

schema mapping is generated by inferring the source and target schemas from these s-t tgds. Next, universal examples are generated by relying on one of the algorithms introduced in Chapter 3. The user is expected to indicate the algorithm of choice by stating in the beginning the class of s-t tgds w.r.t. which the universal example construction should be performed. In other words, the schema mapping



(a) Synthetic.



(b) Real.

Figure 5.2: GUIs of the synthetic and real universal example generators.

will be illustrated by synthetic universal examples w.r.t. the stated class of s-t tgds. The prototype implementation is robust to unsound cases such as illustration of a non-LAV schema mapping w.r.t. the class of LAV s-t tgds. Furthermore, the universal example construction is dependent on a variation of the chase procedure known as the naive chase [6]. The source and target instances of each universal example are only constructed in memory rather than being persisted in a database management system (DBMS). Finally, the generated universal examples are outputted to the user.

Figure 5.1b shows the typical workflow of generating real universal examples. As before, the user is expected to provide s-t tgds that describe a schema mapping. In addition, the user is required to provide information for accessing a source instance stored in a DBMS. The prototype implementation supports only PostgreSQL[1], but this support can be extended to other DBMSs. After parsing the given s-t tgds, a schema mapping is generated based not only on these s-t tgds but also on metadata about the source instance. Next, universal examples are generated through repetitive user interactions by relying on the lazy approach introduced in Section 4.1.2. The eager approach is not exposed to the user due to its unfeasibility even for simple schema mappings, which is further discussed in Section 5.2.1. The universal example construction involves generation and execution of queries for retrieval of data from the given source instance. A query is issued for every universal example to be constructed. As seen before, the source and target instances of each universal example are only constructed in memory. Finally, every generated universal example is outputted to the user.

Figure 5.2a shows the graphical user interfaces (GUI) of the synthetic universal example generator, whereas Figure 5.2b shows the GUI of the real universal example generator.

5.1.1 S-t tgds parser

A parser for s-t tgds has been developed to enable proper reading of s-t tgds from input. The parser was generated from a basic grammar for the schema-mapping language of s-t tgds using ANTLR [7]. The grammar drops the universal and existential quantifiers, but such quantifiers are implicitly assumed. Moreover, the grammar allows the use of string and numeric constants inside atoms. It should be noted that variants of the synthetic and real universal example generators which support s-t tgds with string and numeric constants have also been implemented. The aforementioned grammar is given in Listing 5.1.

```
grammar STTGD;

// parser rules
init        : source '->' target ;
source      : atom ( '&' atom )* ;
target      : atom ( '&' atom )* ;
atom        : relation '(' sequence ')' ;
relation    : ID ;
sequence    : (variable | constant) ( ',' (variable | constant) )* ;
variable    : ID ;
constant    : STRING | NUMBER ;

// lexer rules
ID          : [a-zA-Z] [a-zA-Z0-9_]* ;
STRING      : '"' .*? '"' ;
NUMBER      : [0-9]+ ( '.' [0-9]+ )? ;
WS          : [ \r\t\n]+ -> skip ;
```

Listing 5.1: A grammar for the schema-mapping language of s-t tgds.

The following is a typical s-t tgd complying with the grammar.

```
gene(n1,"primary",p) & gene(n2,"non-primary",p) -> synonym(n2,n1)
```

[1] http://www.postgresql.org/

5.1.2 Naive chase

In [2] construction of universal examples relied on the naive chase and therefore an implementation of this chase has been developed. The pseudocode is given in Algorithm 4, which performs a naive chase on a source instance I using a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and returns a target instance J.

For each s-t tgd σ of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, the algorithm exhaustively explores all possible matches between atoms of $\varphi(\mathbf{x})$ and facts of I using backtracking. Procedure *Backtrack* (lines 17 to 28) traverses an implicit tree of depth equal to the number of atoms in $\varphi(\mathbf{x})$. Any tree node except the root is a tentative match between an atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$ and a fact $R(\mathbf{a})$ of I (line 23). Whenever no match can be established between an atom and a fact, the entire subtree rooted in that node will be disregarded. Only paths of length equal to the tree depth from root to leaves will represent successful matches between atoms of $\varphi(\mathbf{x})$ and facts of I. Procedure *Backtrack* takes as input two multisets A_{match} and A of atoms from $\varphi(\mathbf{x})$ and two multisets I_{match} and I of facts. In the initial call to this procedure A will consist of all atoms of $\varphi(\mathbf{x})$ and I will correspond to the initial source instance, whereas A_{match} and I_{match} will be empty as no matches will have been established at that point (line 8). Subsequent calls to procedure *Backtrack* signal an advance to the next level of the tree from root to leaves and therefore an atom $R(\mathbf{x})$ that matches with a fact $R(\mathbf{a})$ will be subtracted from A (line 24). Moreover, the atom $R(\mathbf{x})$ and the fact $R(\mathbf{a})$ will be added to A_{match} and I_{match}, respectively. Whenever all atoms of $\varphi(\mathbf{x})$ have been successfully matched with some facts of I, I_{match} will consist of one single fact for each atom of $\varphi(\mathbf{x})$. However, there may be the case that further matches between atoms of $\varphi(\mathbf{x})$ and other facts of I can be uncovered. For this reason, each successful match between all atoms of $\varphi(\mathbf{x})$ and some facts of I will be preserved in a set $\mathcal{S}$ with a global scope (line 19). In other words, $\mathcal{S}$ will consist of all paths of length equal to the tree depth from root to leaves. It should be noted that procedure *Backtrack* will always attempt to match the leftmost atom from the remaining atoms of $\varphi(\mathbf{x})$ (line 21), thus the matching of atoms from $\varphi(\mathbf{x})$ will progress from left to right.

The final part of the naive chase consists of generating and adding facts to J for each successful match between all atoms of $\varphi(\mathbf{x})$ and facts of I (lines 9 to 13). Based on I_{match}, which consists of a fact $R(\mathbf{a})$ for each atom $R(\mathbf{x})$ of $\varphi(\mathbf{x})$, function *TargetFacts* generates a fact $P(\mathbf{a}, \mathbf{b})$ for each atom $P(\mathbf{x}, \mathbf{y})$ of $\psi(\mathbf{x}, \mathbf{y})$, where $\mathbf{a}$ is a tuple of constants interpreting tuple $\mathbf{x}$ of universally quantified variables and $\mathbf{b}$ is a tuple of new nulls interpreting tuple $\mathbf{y}$ of existentially quantified variables.

This naive chase never fails and always terminates, since schema mapping $\mathcal{M}$ is specified only by source-to-target dependencies and no target dependencies. In general, the chase fails when a target dependency such as an equality generating dependency is not satisfied and may not terminate due to cyclic target dependencies. However, when the set of target dependencies is empty, a universal solution always exists and can be constructed in polynomial time [3, 4].

The naive chase is generally nondeterministic because of an arbitrary trigger[2] choice at each chase step[3] [6]. Nevertheless, this naive chase is deterministic, since triggers are chosen in the same order at each execution, for the same source instance.

Example 6. Consider source instance $I = \{E(1,3), E(2,4), E(3,4), E(4,3)\}$ and schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{E\}$, $\mathbf{T} = \{F\}$, and Σ consists of the single s-t tgd $E(x,z) \wedge E(z,y) \rightarrow F(x,y)$, denoted by σ.

The following is the chase sequence of instance I with Σ, where $J_i \xrightarrow{*,(\sigma,h_i)} J_{i+1}$ is a chase step, (σ, h_i) is a trigger, h_i is a homomorphism that maps variables occurring on the left-hand side of σ to I, $0 \leq i \leq 3$, and J_4 is a universal solution.

$$J_0 \xrightarrow{*,(\sigma,h_0)} J_1 \xrightarrow{*,(\sigma,h_1)} J_2 \xrightarrow{*,(\sigma,h_2)} J_3 \xrightarrow{*,(\sigma,h_3)} J_4$$

[2] A trigger is a pair consisting of a s-t tgd and a homomorphism that maps variables occurring on the left-hand side of the s-t tgd to a source instance.

[3] A chase step is the application of a s-t tgd to an instance using a homomorphism. In function *NaiveChase*, each iteration of the middle *for* loop (lines 9 to 13) corresponds to a chase step.

Algorithm 4 Naive chase.

```
1: function NaiveChase(I, M)
2:     J ← ∅
3:     for all σ ∈ Σ do
4:         S ← ∅
5:         A_match ← ∅
6:         A ← Atoms(φ(x))
7:         I_match ← ∅
8:         Backtrack(A_match, A, I_match, I)
9:         for all I_match ∈ S do
10:            for all P(a, b) ∈ TargetFacts(ψ(x, y), I_match) do
11:                J ← J ∪ {P(a, b)}
12:            end for
13:        end for
14:    end for
15:    return J
16: end function

17: procedure Backtrack(A_match, A, I_match, I)
18:    if A = ∅ then
19:        S ← S ∪ {I_match}
20:    else
21:        R(x) ← LeftmostAtom(A)
22:        for all R(a) ∈ I, do
23:            if Match(R(x), R(a)) then
24:                Backtrack(A_match ∪ {R(x)}, A − {R(x)}, I_match ∪ {R(a)}, I)
25:            end if
26:        end for
27:    end if
28: end procedure
```

$$
\begin{array}{ll}
 & J_0 = \emptyset \\
h_0 = \{x/1, y/4, z/3\} & J_1 = \{F(1,4)\} \\
h_1 = \{x/2, y/3, z/4\} & J_2 = \{F(1,4), F(2,3)\} \\
h_2 = \{x/3, y/3, z/4\} & J_3 = \{F(1,4), F(2,3), F(3,3)\} \\
h_3 = \{x/4, y/4, z/3\} & J_4 = \{F(1,4), F(2,3), F(3,3), F(4,4)\}
\end{array}
$$

5.2 Case studies

5.2.1 Eager generation of real universal examples for a LAV schema mapping

The worst case of eager generation of real universal examples for a simple LAV schema mapping is examined.

Consider $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{R\}$, $\mathbf{T} = \{P\}$, and Σ consists of the single ternary copy s-t tgd $R(x_1, x_2, x_3) \rightarrow P(x_1, x_2, x_3)$. All attributes of relation R share the same domain, and the same holds for relation P. The aim is to examine the worst case in which the search space undergoes no reduction w.r.t. domain constraints. In addition, consider a source instance I.

The groups of isomorphic synthetic tuples for relation R coincide with the groups already seen in Example 4. Since all attributes of relation R share the same domain, all groups are retained, and no reduction in terms of domain constraints is possible. Taking into account the number of groups and the number of tuples in every group, there are 3888 possible sequences of synthetic tuples. The only viable reduction to this space of sequences is the removal of isomorphic ones. As a result, the number of sequences is substantially reduced to 648. However, this is still a significant number in terms of potential conjunctive queries to be executed on I.

In the following, the execution time of the eager approach is reported in three different settings. The execution time was averaged over 10 runs.

In the first setting, source instance I was loaded with 574,603 tuples in a particular order. For each attribute of relation R, its set of values from $adom(I)$ was disjoint from the set of values from $adom(I)$ of any other attribute of R. In other words, executing all conjunctive queries on I would have yielded no result. The eager approach took 2 minutes and 38.298 seconds to execute all 648 conjunctive queries.

In the second setting, source instance I was loaded with 574,603 tuples in the same order as before. However, five tuples at random positions were replaced with tuples (a, a, a) (c, b, b) (c, b, c) (b, b, a) (c, b, a), which in fact were an answer to one of the conjunctive queries. The eager approach took 5 minutes and 46.287 seconds to find the answer after executing 549 out of 648 conjunctive queries.

In the third setting, source instance I was loaded with 574,603 tuples in the same order as before. However, five tuples at the same positions as before were replaced with tuples (b, b, b) (a, c, c) (a, c, a) (a, a, b) (c, b, a), which again were an answer to one of the conjunctive queries. The eager approach took 3 minutes and 57.223 seconds to find the answer after executing 401 out of 648 conjunctive queries.

Interestingly, executing all conjunctive queries on a source instance with no answer took less than executing only a number of queries on a source instance with an answer.

The eager approach, however, is impractical for a LAV schema mapping $\mathcal{M}' = (\mathbf{S}', \mathbf{T}', \Sigma')$, where $\mathbf{S}' = \{E\}$, $\mathbf{T}' = \{F\}$, and Σ' consists of the quaternary copy s-t tgds $E(x_1, x_2, x_3, x_4) \rightarrow F(x_1, x_2, x_3, x_4)$. Using equation (3.1) and equation (3.2) from Section 3.1.1, the number of possible sequences of synthetic tuples is given by

$$|\mathcal{G}^1|^{\mathcal{S}(4,1)} \cdot |\mathcal{G}^2|^{\mathcal{S}(4,2)} \cdot |\mathcal{G}^3|^{\mathcal{S}(4,3)} \cdot |\mathcal{G}^4|^{\mathcal{S}(4,4)} = 4^1 \cdot 12^7 \cdot 24^6 \cdot 24^1 = 657,366,253,849,018,368,$$

which cannot be supported by the standard data structures of the prototype implementation.

5.2.2 Reducing the number of universal examples for a 2-modular schema mapping

Reductions on the space of source instances and therefore on the space of synthetic universal examples that illustrate a simple 2-modular schema mapping are examined. These reductions have been introduced in Section 3.3.1.

Consider the 2-modular schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{E, G\}$, $\mathbf{T} = \{F\}$, and Σ consists of the single s-t tgd $E(x, y) \wedge G(y, x) \rightarrow F(x, y)$. This s-t tgd was used as a running example in Section 3.3.

Initially, the number of possible source instances and therefore the number of possible universal examples is given by $|\mathcal{P}(T)_{E(x,y)}| \cdot |\mathcal{P}(T)_{G(y,x)}| = 15 \cdot 15 = 225$, where $\mathcal{P}(T)_{E(x,y)}$ and $\mathcal{P}(T)_{G(y,x)}$ denote power sets without the empty set corresponding to source atoms. After applying the unsatisfied join condition reduction, the number of source instances decreases to 193. Furthermore, after applying the isomorphic instance reduction, the number of source instances drops to 134.

The unsatisfied join condition reduction arose from the observation that some source instances yield empty target instances. An example of such a source instance is $I = \{E(a, a), E(a, b), G(b, b)\}$, where the set of values $\{a, a\}$ corresponding to variable x of E is disjoint from the set of values $\{b\}$ corresponding to variable x of G. In the prototype implementation the unsatisfied join condition reduction basically discards a source instance if sets of values corresponding to a join variable are disjoint. However, this check is not resilient to every case. An example is source instance $I' = \{E(a, b), E(b, a), G(a, a)\}$. The set of values $\{a, b\}$ corresponding to variable x of E is not disjoint from the set of values $\{a\}$ corresponding to variable x of G. Furthermore, the set of values $\{b, a\}$ corresponding to variable y of E is not disjoint from the set of values $\{a\}$. As a result, the source instance is not discarded, although it yields an empty target instance. The limitation lies in the fact that $\{a, b\}$ of x and $\{b, a\}$ of y are not correlated in a row-wise fashion, where both x and y are variables of E. For this reason, 13 out of the remaining 134 source instances still yield an empty target instance. Nevertheless, uncovering that a simple schema mapping such as $\mathcal{M}$ requires a number of illustrative universal examples in the order of a hundred w.r.t. the class of all 2-modular s-t tgds is not a promising result. In the next section, a relatively simple schema mapping is shown to require a number of universal examples in the order of hundreds w.r.t. the class of all 3-modular s-t tgds.

5.2.3 STBenchmark basic mapping scenarios

STBenchmark [1] was originally introduced to support assessment of tools intended for schema mapping design. It includes a number of basic mapping scenarios that describe typical data transformations from a source schema to a target schema such as copying of relations, horizontal and vertical partitioning of relations, and augmenting of relations with additional attributes. These mapping scenarios were initially conceived for XML data and therefore some of them considered nested structures. For the purpose of this work, relevant mapping scenarios were adapted to the relational model. Each mapping scenario consists of a schema mapping and a realistic source instance. This source instance is used only to generate real universal examples. In what follows, the horizontal partition scenario is discussed, but additional scenarios are covered in Appendix A.

Horizontal Partition

Consider LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{gene\}$, $\mathbf{T} = \{gene, synonym\}$, and Σ consists of the following s-t tgds:

$$gene(n, \text{“primary”}, p) \rightarrow gene(n, p)$$
$$gene(n, \text{“non-primary”}, p) \rightarrow synonym(n, p)$$

In this scenario, source relation *gene* is horizontally partitioned into target relations *gene* and *synonym* based on attribute *type* of source relation *gene*. If the value of this attribute is “primary”, a tuple is created in target relation *gene*. However, if the value of this attribute is “non-primary”, a tuple is created in target relation *synonym*.

The synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances given in Table 5.1 and the target instances shown in Tables 5.2 and 5.3. For readability purposes, all source instances are listed in the same table. However, the reader should bear in mind that each source instance I_i consists of one relation *gene* with one single tuple, where $1 \leq i \leq 10$. This fact is in accordance with the steps on constructing universal examples given in Section 3.1. Synthetic universal examples are generated under the assumption that all attributes of source relation *gene* share the same domain of values. For this reason, some tuples include repeated values.

	name	type	protein
I_1	c	primary	b
I_2	b	primary	b
I_3	primary	primary	b
I_4	b	primary	primary
I_5	primary	primary	primary
I_6	c	non-primary	b
I_7	b	non-primary	b
I_8	non-primary	non-primary	b
I_9	b	non-primary	non-primary
I_{10}	non-primary	non-primary	non-primary

Table 5.1: gene (source)

	name	protein
J_1	c	b
J_2	b	b
J_3	primary	b
J_4	b	primary
J_5	primary	primary

Table 5.2: gene (target)

	name	protein
J_6	c	b
J_7	b	b
J_8	non-primary	b
J_9	b	non-primary
J_{10}	non-primary	non-primary

Table 5.3: synonym (target)

The synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds include the source instances given in Table 5.4 and the target instances shown in Tables 5.5 and 5.6. Source instance I_1 of Table 5.4 is isomorphic to source instance I_1 of Table 5.1. Likewise, the corresponding target instance J_1 of Table 5.5 is isomorphic to target instance J_1 of Table 5.2. In effect, universal examples that illustrate a schema mapping w.r.t. the class of all strict LAV s-t tgds have isomorphic counterparts among universal examples that illustrate a schema mapping w.r.t. the class of all LAV s-t tgds.

	name	type	protein
I_1	a	primary	b
I_2	a	non-primary	b

Table 5.4: gene (source)

	name	protein
J_1	a	b

Table 5.5: gene (target)

	name	protein
J_2	a	b

Table 5.6: synonym (target)

The initial number of synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all 3-modular s-t tgds is given by

$$|\mathcal{P}(T)| + |\mathcal{P}(T')| = 2^{|T|} - 1 + 2^{|T'|} - 1 = 2^9 - 1 + 2^9 - 1 = 1022,$$

where $\mathcal{P}(T)$ is the power set without the empty set corresponding to atom $gene(n, \text{“primary”}, p)$, and $\mathcal{P}(T')$ is the power set without the empty set corresponding to atom $gene(n, \text{“non-primary”}, p)$. Set T is obtained by first considering all possible 3-tuples with elements from $D = \{primary, b, c\}$ and afterwards discarding those 3-tuples whose second element is not constant “primary”. Set D is defined as indicated in Section 3.3, but here one of the set elements is replaced by constant “primary”. Set T' is obtained similarly.

Considering the isomorphic instance reduction introduced in Section 3.3.1, the number of universal examples decreases to 682. Even for a relatively simple schema mapping such as $\mathcal{M}$, the number of synthetic universal examples generated w.r.t. the class of all 3-modular s-t tgds can be high. Only one of these universal examples is reported here. Table 5.7 shows the source instance I of this universal example, whereas Table 5.8 gives the target instance J of this universal example.

	name	**type**	**protein**
I	primary	primary	primary
	b	primary	primary
	c	primary	primary
	primary	primary	b
	b	primary	b
	c	primary	b
	primary	primary	c
	b	primary	c
	c	primary	c

Table 5.7: gene (source)

	name	**protein**
J	primary	primary
	b	primary
	c	primary
	primary	b
	b	b
	c	b
	primary	c
	b	c
	c	c

Table 5.8: gene (target)

The real universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances given in Table 5.9 and the target instances shown in Table 5.10 and 5.11. These universal examples consist of realistic tuples as opposed to the universal examples given above, and these tuples have been retrieved from a realistic source instance accompanying the mapping scenario. Furthermore, these real universal examples have been generated under the assumption that attributes of relation *gene* have distinct domains of values. It can be argued that a value such as "GF14A" of attribute *name* and a value such as "primary" of attribute *type* can originate from a domain such as $varchar(64)$ associated with both attributes, but nevertheless these attributes have different semantics and therefore their values are assumed to be distinct.

	name	**type**	**protein**
I_1	GF14A	primary	14-3-3-like protein A
I_2	Mb2057c	non-primary	14 kDa antigen

Table 5.9: gene (source)

	name	**protein**
J_1	GF14A	14-3-3-like protein A

Table 5.10: gene (target)

	name	**protein**
J_2	Mb2057c	14 kDa antigen

Table 5.11: synonym (target)

Chapter 6

Conclusions

In this work, construction of synthetic and real universal examples for schema mappings with a finite semantic description was examined. A schema mapping specified by a class of s-t tgds such as LAV, strict LAV, n-modular, and self-join-free on the source can be illustrated by finitely many universal examples, and a main concern was the reduction of this space of universal examples.

For LAV schema mappings a reduction based on isomorphism between tuples was shown. The source instance of each universal example that describes a LAV schema mapping holds a single relation with one tuple and therefore it was sufficient to reason about isomorphism only in terms of tuples. By contrast, n-modular and self-join-free on the source schema mappings can be described by universal examples whose source instance can hold multiple tuples across different relations, and in this case a reduction based on isomorphism between source instances was shown.

In addition, for n-modular and self-join-free on the source schema mappings a second reduction of the number of illustrative universal examples was proposed, which is based on unsatisfied join conditions between relations from the same source instance. Such a source instance yields an empty target instance, and a universal example consisting of such a pair reveals no more than behavior disallowed by the schema mapping and is not deemed to be informative enough.

The space of illustrative universal examples for strict LAV schema mappings is already minimal and therefore requires no reduction, since the number of universal examples is given by the number of source relations.

In addition, the reduction of universal examples for LAV schema mappings revealed a result in terms of the number of universal examples needed to illustrate LAV s-t tgds. This result confirms that the number of variables occurring on the left-hand side of a LAV s-t tgd exponentially impacts the number of illustrative universal examples and is closely connected with combinatorial notions such as Bell number and Stirling number of the second kind.

Two approaches to construct real universal examples for LAV schema mappings were shown. In both approaches the starting point is a space of synthetic tuples corresponding to source relations. Furthermore, in both approaches domain constraints from source relations are considered. The first approach attempts to eagerly generate all illustrative universal examples in one go, and it employs several reductions of the synthetic tuple space. Initially, isomorphism between tuples is used to group them, and domain constraints are used to discard some of them. Next, the focus shifts on sequences of tuples, that is, all possible ways of selecting one tuple from every remaining group. Isomorphism between sequences is used to discard some of them, and domain constraints are used once more to discard some of the sequences. The remaining sequences form building blocks for conjunctive queries intended to retrieve real tuples mimicking the synthetic ones from a source database. Finally, each tuple leads to construction of a universal example, since universal examples for LAV schema mapping hold no more than a single tuple in their source instance.

The second approach generates in a lazy fashion one universal example at a time as the need for it arises and is more light in terms of reductions on the synthetic tuple space. As in the eager approach, isomorphism between tuples is used to group them, and domain constraints are used to discard them. However, no rigorous reduction is performed for sequences of tuples because the aim in this case is to find only one such sequence, which still needs to conform with domain constraints. Finally, each tuple of the identified sequence is a building block for a simple query that retrieves a real tuple mimicking

the synthetic one. As in the eager approach, each retrieved tuple leads to construction of a universal example. Although this approach might lack the sophistication of the eager approach, it is resilient to cases that cannot be resolved by the eager approach.

By using a prototype implementation of the approaches to construct synthetic and real universal examples, several case studies were examined. The eager approach to construct real universal examples was revealed to be impractical even for simple LAV schema mappings due to an exponential blow-up of the space represented by possible sequences of tuples. Furthermore, it was uncovered that illustration of relatively simple schema mappings w.r.t. the class of all n-modular s-t tgds already requires a number of synthetic universal examples in the order of hundreds, and the prototype implementation is not suited to explore huge spaces of universal examples associated with more complex schema mappings. However, this finding gives enough ground to assert that for a n-modular s-t tgd not only the number of variables occurring on the left-hand side exponentially impacts the number of illustrative universal examples, as seen in the case of LAV s-t tgds, but also the number of atoms in the conjunction on the left-hand side.

Future work

The number of illustrative universal examples for relatively simple n-modular schema mappings can be high in spite of considering two reductions of the space of universal examples. A more precise indication of this number similar to the combinatorial result for LAV s-t tgds would be relevant in establishing whether further reductions are possible.

An aspect overlooked by this work is whether universal examples generated by the prototype implementation can support users in gaining understanding of schema mappings, and such a study would be in order.

The algorithm for generating synthetic universal examples for LAV schema mappings can yield universal examples with an empty target instance. This outcome can be observed by considering a LAV s-t tgd whose left-hand side consists of repeated variables. Should universal examples with an empty target instance be deemed uninformative just as in the n-modular case, the LAV algorithm could be adjusted to rely on the number of variables occurring on the left-hand side of the s-t tgd instead of the arity of the source relation symbol.

Extending the support of the prototype implementation to other DBMSs would involve not only a simple connectivity adjustment, but also a reconsideration of how some queries are generated.

Bibliography

[1] Bogdan Alexe, Wang Chiew Tan, and Yannis Velegrakis. Stbenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1):230–244, 2008.

[2] Bogdan Alexe, Balder ten Cate, Phokion G. Kolaitis, and Wang Chiew Tan. Characterizing schema mappings via data examples. *ACM Trans. Database Syst.*, 36(4):23, 2011.

[3] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[4] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.

[5] Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981.

[6] Adrian Onet. The chase procedure and its applications in data exchange. In Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt, editors, *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*, pages 1–37. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

[7] Terence Parr. *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd edition, 2013.

[8] A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill Education, 6th edition, 2010.

[9] Richard P. Stanley. *Enumerative Combinatorics: Volume 1*. Cambridge University Press, New York, NY, USA, 2nd edition, 2011.

[10] Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 63–72. ACM, 2009.

Appendix A

Additional STBenchmark basic mapping scenarios

Copying

Consider LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{protein\}$, $\mathbf{T} = \{protein\}$, and Σ consists of the s-t tgd

$$protein(n, a, c) \rightarrow protein(n, a, c).$$

In this scenario, a ternary relation is simply copied from source to target. In Tables A.1 and A.2 are given the source and target instances, respectively, of the synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds.

	name	**accession**	**created**
I_1	a	a	a
I_2	b	a	a
I_3	a	b	a
I_4	b	b	a
I_5	c	b	a

Table A.1: protein (source)

	name	**accession**	**created**
J_1	a	a	a
J_2	b	a	a
J_3	a	b	a
J_4	b	b	a
J_5	c	b	a

Table A.2: protein (target)

In addition, in Tables A.3 and A.4 are given the source and target instances, respectively, of the synthetic universal example that illustrates $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds.

	name	**accession**	**created**
I	a	b	c

Table A.3: protein (source)

	name	**accession**	**created**
J	a	b	c

Table A.4: protein (target)

The initial number of synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all 3-modular s-t tgds is given by

$$|\mathcal{P}(T)| = 2^{|T|} - 1 = 2^{27} - 1,$$

where $\mathcal{P}(T)$ is the power set without the empty set corresponding to source atom $protein(n, a, c)$. Due to the largeness of the space of synthetic universal examples, no reductions could be performed using the prototype implementation.

In Tables A.5 and A.6 are given the source and target instances, respectively, of the real universal example that illustrates $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds. This real universal example has been generated under the assumption that attributes of source relation *protein* have distinct domains of values.

	name	accession	created
I	12-alpha-hydroxysteroid dehydrogenase	P21215	1991-08-01

Table A.5: protein (source)

	name	accession	created
J	12-alpha-hydroxysteroid dehydrogenase	P21215	1991-08-01

Table A.6: protein (target)

Surrogate Key Assignment

Consider LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{gene\}$, $\mathbf{T} = \{gene, synonym\}$, and Σ consists of the following s-t tgds:

$$gene(n, \text{“primary”}, p) \rightarrow \exists wid(gene(n, p, wid))$$
$$gene(n, \text{“non-primary”}, p) \rightarrow \exists wid(synonym(n, p, wid))$$

This scenario is similar to the horizontal partition scenario discussed in Section 5.2.3. However, in this scenario, target relations *gene* and *synonym* are augmented with attribute *wid*. The synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances shown in Table A.7 and the target instances given in Tables A.8 and A.9.

	name	type	protein
I_1	c	primary	b
I_2	b	primary	b
I_3	primary	primary	b
I_4	b	primary	primary
I_5	primary	primary	primary
I_6	c	non-primary	b
I_7	b	non-primary	b
I_8	non-primary	non-primary	b
I_9	b	non-primary	non-primary
I_{10}	non-primary	non-primary	non-primary

Table A.7: gene (source)

	name	protein	wid
J_1	c	b	null0
J_2	b	b	null0
J_3	primary	b	null0
J_4	b	primary	null0
J_5	primary	primary	null0

Table A.8: gene (target)

	name	protein	wid
J_6	c	b	null0
J_7	b	b	null0
J_8	non-primary	b	null0
J_9	b	non-primary	null0
J_{10}	non-primary	non-primary	null0

Table A.9: synonym (target)

In addition, the synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds include the source instances given in Table A.10 and the target instances shown in Tables A.11 and A.12.

	name	type	protein
I_1	a	primary	b
I_2	a	non-primary	b

Table A.10: gene (source)

	name	protein	wid
J_1	a	b	null0

Table A.11: gene (target)

	name	protein	wid
J_2	a	b	null0

Table A.12: synonym (target)

The initial number of synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all 3-modular

s-t tgds is given by

$$|\mathcal{P}(T)| + |\mathcal{P}(T')| = 2^{|T|} - 1 + 2^{|T'|} - 1 = 2^9 - 1 + 2^9 - 1 = 1022,$$

where $\mathcal{P}(T)$ is the power set without the empty set corresponding to atom $gene(n, \text{“primary”}, p)$ and $\mathcal{P}(T')$ is the power set without the empty set corresponding to atom $gene(n, \text{“non-primary”}, p)$. As seen before in the horizontal partion scenario discussed in Section 5.2.3, after applying the isomorphic instance reduction, the number of remaining universal examples is 682.

The real universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances show in Table A.13 and the target instances given in Tables A.14 and A.15. These real universal examples have been generated under the assumption that attributes of relation *gene* have distinct domains of values.

	name	**type**	**protein**
I_1	GF14A	primary	14-3-3-like protein A
I_2	Mb2057c	non-primary	14 kDa antigen

Table A.13: gene (source)

	name	**protein**	**wid**
J_1	GF14A	14-3-3-like protein A	null0

Table A.14: gene (target)

	name	**protein**	**wid**
J_2	Mb2057c	14 kDa antigen	null0

Table A.15: synonym (target)

Vertical Partition (Normalization)

Consider LAV schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S} = \{reaction\}$, $\mathbf{T} = \{reaction, chemicalinfo\}$, and Σ consists of the s-t tgd

$$reaction(en, n, c, o, d, eq) \rightarrow \exists f(reaction(en, n, c, o, \underline{f}) \wedge chemicalinfo(d, eq, \underline{f})).$$

In this scenario, source relation *reaction* is vertically partitioned into target relations *reaction* and *chemicalinfo*. The synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances given in Table A.16 and the target instances show in Tables A.17 and A.18.

	entry	**name**	**comment**	**orthology**	**definition**	**equation**
I_1	a	a	a	a	a	a
I_2	b	a	a	a	a	a
I_3	a	b	a	a	a	a
I_4	b	b	a	a	a	a
I_5	c	b	a	a	a	a
...	...	...	...	...	...	...
I_{199}	b	e	d	c	b	a
I_{200}	c	e	d	c	b	a
I_{201}	d	e	d	c	b	a
I_{202}	e	e	d	c	b	a
I_{203}	f	e	d	c	b	a

Table A.16: reaction (source)

	entry	**name**	**comment**	**orthology**	**cofactor**
J_1	a	a	a	a	null0
J_2	b	a	a	a	null0
J_3	a	b	a	a	null0
J_4	b	b	a	a	null0
J_5	c	b	a	a	null0
...	...	...	...	...	...
J_{199}	b	e	d	c	null0
J_{200}	c	e	d	c	null0
J_{201}	d	e	c	c	null0
J_{202}	e	e	d	c	null0
J_{203}	f	e	d	c	null0

Table A.17: reaction (target)

	definition	**equation**	**cofactor**
J_1	a	a	null0
J_2	a	a	null0
J_3	a	a	null0
J_4	a	a	null0
J_5	a	a	null0
...	...	...	...
J_{199}	b	a	null0
J_{200}	b	a	null0
J_{201}	b	a	null0
J_{202}	b	a	null0
J_{203}	b	a	null0

Table A.18: chemicalinfo (target)

Furthermore, the synthetic universal example that illustrates $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds includes the source instance shown in Table A.19 and the target instance given in Tables A.20 and A.21.

	entry	**name**	**comment**	**orthology**	**definition**	**equation**
I	a	b	c	d	e	f

Table A.19: reaction (source)

	entry	**name**	**comment**	**orthology**	**cofactor**
J	a	b	c	d	null0

Table A.20: reaction (target)

	definition	**equation**	**cofactor**
J	e	f	null0

Table A.21: chemicalinfo (target)

The initial number of synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all 6-modular s-t tgds is given by

$$|\mathcal{P}(T)| = 2^{|T|} - 1 = 2^{46656} - 1,$$

where $\mathcal{P}(T)$ is the power set without the empty set corresponding to atom $reaction(en, n, c, o, d, eq)$. Due to the largeness of the space of synthetic universal examples, no reductions could be performed using the prototype implementation.

The real universal example that illustrates $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds includes the source instance shown in Table A.22 and the target instance given in Tables A.23 and A.24. This real universal example has been generated under the assumption that attributes of relation *reaction* have distinct domains of values.

	entry	**name**	**comment**	**orthology**	**definition**	**equation**
I	R06733	carboxylesterase	multi-step reaction	KO: K01913	Hygrine gives Tropinone	C06179 gives C00783

Table A.22: reaction (source)

	entry	**name**	**comment**	**orthology**	**cofactor**
J	R06733	carboxylesterase	multi-step reaction	KO: K01913	null0

Table A.23: reaction (target)

	definition	**equation**	**cofactor**
J	Hygrine gives Tropinone	C06179 gives C00783	null0

Table A.24: chemicalinfo (target)

Nesting

Consider LAV schema mapping $\mathcal{M} = \{\mathbf{S}, \mathbf{T}, \Sigma\}$, where $\mathbf{S} = \{reference\}$, $\mathbf{T} = \{period, author, publication\}$, and Σ consists of the s-t tgd

$$reference(t, y, p, n) \rightarrow \exists pid, aid(period(y, \underline{pid}) \wedge author(n, \underline{aid}, \underline{pid}) \wedge publication(t, p, \underline{aid})).$$

In this scenario, the "flat" source relation *reference* is transformed into target relations *period*, *author*, and *publication* such that for each period there is a number of authors and for each author there is a number of publications. However, this hierarchical structure is not completely captured by $\mathcal{M}$ due to the lack of target constraints. For instance, relation *period* may contain more than one tuple for the same period, but each of these tuples would have a different value for attribute *pid*. The synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds include the source instances given in Table A.25 and the target instances shown in Tables A.26, A.27, and A.28.

	title	**year**	**publishedin**	**name**
I_1	a	a	a	a
I_2	b	a	a	a
I_3	a	b	a	a
I_4	b	b	a	a
I_5	c	b	a	a
I_6	a	a	b	a
I_7	b	a	b	a
I_8	c	a	b	a
I_9	a	b	b	a
I_{10}	b	b	b	a
I_{11}	c	b	b	a
I_{12}	a	c	b	a
I_{13}	b	c	b	a
I_{14}	c	c	b	a
I_{15}	d	c	b	a

Table A.25: reference (source)

	year	pid
J_1	a	null0
J_2	a	null0
J_3	b	null0
J_4	b	null0
J_5	b	null0
J_6	a	null0
J_7	a	null0
J_8	a	null0
J_9	b	null0
J_{10}	b	null0
J_{11}	b	null0
J_{12}	c	null0
J_{13}	c	null0
J_{14}	c	null0
J_{15}	c	null0

Table A.26: period (target)

	name	aid	pid
J_1	a	null1	null0
J_2	a	null1	null0
J_3	a	null1	null0
J_4	a	null1	null0
J_5	a	null1	null0
J_6	a	null1	null0
J_7	a	null1	null0
J_8	a	null1	null0
J_9	a	null1	null0
J_{10}	a	null1	null0
J_{11}	a	null1	null0
J_{12}	a	null1	null0
J_{13}	a	null1	null0
J_{14}	a	null1	null0
J_{15}	a	null1	null0

Table A.27: author (target)

	title	publishedin	aid
J_1	a	a	null1
J_2	b	a	null1
J_3	a	a	null1
J_4	b	a	null1
J_5	c	a	null1
J_6	a	b	null1
J_7	b	b	null1
J_8	c	b	null1
J_9	a	b	null1
J_{10}	b	b	null1
J_{11}	c	b	null1
J_{12}	a	b	null1
J_{13}	b	b	null1
J_{14}	c	b	null1
J_{15}	d	b	null1

Table A.28: publication (target)

Moreover, the synthetic universal example that illustrates $\mathcal{M}$ w.r.t. the class of all strict LAV s-t tgds includes the source instance shown in Table A.29 and the target instance given in Tables A.30, A.31, and A.32.

	title	year	publishedin	name
I	a	b	c	d

Table A.29: reference (source)

	year	pid
J	b	null0

Table A.30: period (target)

	name	aid	pid
J	d	null1	null0

Table A.31: author (target)

	title	publishedin	aid
J	a	c	null1

Table A.32: publication (target)

The initial number of synthetic universal examples that illustrate $\mathcal{M}$ w.r.t. the class of all 4-modular s-t tgds is given by

$$|\mathcal{P}(T)| = 2^{|T|} - 1 = 2^{256} - 1,$$

where $\mathcal{P}(T)$ is the power set without the empty set corresponding to atom $reference(t, y, p, n)$. Due to the largeness of the space of synthetic universal examples, no reductions could be performed using the prototype implementation.

The real universal example that illustrates $\mathcal{M}$ w.r.t. the class of all LAV s-t tgds includes the source instance given in Table A.33 and the target instance shown in Tables A.34, A.35, and A.36. This real universal example has been generated under the assumption that attributes of relation $reference$ have distinct domains of values.

	title	**year**	**publishedin**	**name**
I	Trie Methods for Text and Spatial Data on Secondary Storage	1994	Published by McGill University	Heping Shang

Table A.33: reference (source)

	year	**pid**
J	1994	null0

Table A.34: period (target)

	name	**aid**	**pid**
J	Heping Shang	null1	null0

Table A.35: author (target)

	title	**publishedin**	**aid**
J	Trie Methods for Text and Spatial Data on Secondary Storage	Published by McGill University	null1

Table A.36: publication (target)