Representation systems for data exchange

Gösta Grahne
Concordia University
Montreal, Canada, H3G 1M8
grahne@cs.concordia.ca

Adrian Onet
Concordia University
Montreal, Canada, H3G 1M8
a_onet@cs.concordia.ca

ABSTRACT

The notion of *representation systems* describes structures that are algebraically closed under queries. It has recently been realized that representation systems are highly relevant also in the context of data exchange. We extend the notion of representation system to encompass *data exchange mappings* and their composition. Seen through this lens, two major classes of representation systems emerge, namely *homomorphic data exchange systems* and *strong data exchange systems*. The homomorphic "OWA" systems encompass the "classical" part of data exchange. Reasoning is modulo homomorphic equivalence (CQ-equivalence), and only unions of conjunctive queries and monotone data exchange mappings are supported.

We then develop some new technical tools that allow us to prove that there is a class of '"CWA" strong representation systems in which reasoning is modulo isomorphic equivalence. These systems are based on conditional tables, and they support first order queries and non-monotonic data exchange mappings specified by a large class of second order dependencies. We achieve this by showing that, under a CWA-interpretation, conditional tables are chaseable with the aforementioned class of second order dependencies, and that the class is closed under composition in the CWA-setting.

We also introduce a stricter notion of composability, and show that the class of (first order) source-to-target tuple generating dependencies is closed under the stricter notion of composability.

Categories and Subject Descriptors

H.2.5 [**Heterogeneous Databases**]: Data translation

General Terms

Algorithms, Theory

ICDT 2012, March 26–30, 2012, Berlin, Germany.
Copyright 2012 ACM 978-1-4503-0791-8/12/03 ...$10.00

Keywords

Data exchange, schema mapping, metadata management, incomplete databases

1. INTRODUCTION

Data exchange deals with mappings between schemas and the realization of the application of the mappings on (data base) instances through an algebraic proof procedure known as the *Chase*. Significant advances have already been made in determining closure properties of a number of subclasses of mappings under "schema-management" operations such as composition and inverse. For a snapshot of this rapidly advancing area, see e.g. [5].

There has been less focus on closure properties of classes of instances under various classes of mappings. Recently Arenas et al. [4] made an important contribution by showing that the concept of *representation systems*, that has long been used in the semantics of querying incomplete information data bases, has useful applications in data exchange.

In this paper we provide a systematic application of representation systems in data exchange. A representation system for data exchange consists of a class of *tables* used to represent incomplete instances, *an interpretation function* for the tables, a class of *queries*, and a class of *mappings*, such that the table class is closed both under data exchange (i.e. application of the mappings, target materialization) and answering queries over the target. This takes into account the important interdependence between querying and exchanging data.

Two major classes of representation systems for data exchange emerge. The first, called *homomorphic data exchange systems* accounts for most of the classical results in data exchange. The "canonical" system consists of classical tableaux as table class, open world interpretation, unions of conjunctive queries as query class, and "plain" second order source-to-target tgd's as mapping class. This system is closely related to the concept of "universal models" heavily used in data exchange.

The second class of representation systems for data exchange consists of so called *strong systems* that are based on the closed world assumption and sets of models, rather than universal models. We obtain important technical results that allow us to show that the class of conditional tables, closed world interpretation, the class of first order queries, and mappings specified by a large class of second order dependencies form such a strong representation system. Our analysis also shows that in order to get "the full effect" of second order dependencies, we need strong representation systems, where the tableaux are upgraded to conditional tables.

We then integrate the concept of closure of mappings under composition into the representation systems. When adding composition to a data exchange system we are interested in either *homomorphic composable data exchange systems* or *strong composable data exchange systems*. The former system account for composition under UCQ-equivalence, covering most of the current data exchange setting requirements. The latter takes into account the composition under a closed world assumption semantics allowing for non-monotonic queries.

In the mapping composition introduced by Fagin et al. [16], two mappings M and M' are considered "composable" if the target schema of the first mapping matches the source schema of the second mapping. We refer to composing under this composability criterion as *standard composition*. Several classes of mappings have recently been shown to be closed under standard composition [16, 3, 7]. Unfortunately, the class of mappings specified by source-to-target tgd's, the most prominent in data exchange setting, has been shown not to be closed under standard composition. Even more, it is shown in [16] that one may need second order dependencies to represent the standard composition of two mappings specified by source-to-target tgd's. Second order dependencies are a known cause of intractability. As a remedy we introduce new composability criteria for which the class mappings specified by source-to-target tgd's is closed. The new composability criteria, called *semi-LAV composability*, properly subsumes the results on closure under standard composition for the class of mappings specified by sets of source-to-target full tgd's [16] and the class specified by true-LAV source-to-target tgd's [7].

The rest of this paper is organized as follows: In Section 2 we review incomplete information and representation systems. In Section 3 we add classes of mappings to the representation systems, and obtain *data exchange systems*. We unearth both homomorphic and strong data exchange systems. Section 4 focuses on the composition of mappings and introduces new notions of compositions, *weak composition* for homomorphic systems and *strong composition* for strong systems. In Section 5 we introduce the semi-LAV composability criteria. These criteria ensure closure under composition for larger classes of mappings, still with tractable membership problem. Conclusions are drawn and further work is envisioned in Section 5.

In order to emphasize what we consider "canonical" data exchange systems in each case, we have highlighted the corresponding theorems with a surrounding box. By a canonical system we mean a system where every component is optimal, in the sense that "improving" any component no longer results in a data exchange system. We have however not included any negative results to substantiate the optimality, as there are a plethora of such propositions.

2. INCOMPLETE INFORMATION AND REPRESENTATION SYSTEMS

For basic definitions and concepts we refer to [1]. We assume familiarity with relational query languages. We use UCQ for the class of unions of conjunctive queries, and FO for the class of first order queries. We use the symbol $\subseteq$ for subset inclusion, and $\subset$ for proper set inclusion.

Relational schemas and instances. A *relational schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_n\}$ of relation names, each with an associated *arity* in $\mathbb{N}$. The arity of R_i is denoted $arity(R_i)$. Let Cons be a countably infinite set of *constants*, usually denoted $a, b, c, \ldots$, possibly subscripted. From the domain Cons and a schema $\mathbf{R}$ we build up a Herbrand structure consisting of all expressions of the form $R(a_1, a_2, \ldots, a_k)$, where $R \in \mathbf{R}$, $arity(R) = k$, and the a_i's are values in Cons. Such expressions are called *ground atoms*, or *ground tuples*. A database instance I over $\mathbf{R}$ is then simply a finite set of ground tuples over over $\mathbf{R}$, e.g. $\{R_1(a_1, a_3), R_2(a_2, a_3, a_4)\}$. We denote the set of constants occurring in an instance I with $dom(I)$. The set of all instances over $\mathbf{R}$ is denoted $Inst(\mathbf{R})$.

Incomplete instances. An incomplete database (over a schema $\mathbf{R}$) is conceptually a set $\mathcal{X}$ of complete instances (over the schema $\mathbf{R}$), or *possible worlds* I. Given a query Q and an incomplete database $\mathcal{X}$, the result of Q on $\mathcal{X}$ is $Q(\mathcal{X}) = \{Q(I) : I \in \mathcal{X}\}$. To this *exact answer* [19] there are two approximations [24], namely

$$cert_Q(\mathcal{X}) = \bigcap\{Q(I) : I \in \mathcal{X}\}, \text{ and}$$
$$poss_Q(\mathcal{X}) = \bigcup\{Q(I) : I \in \mathcal{X}\},$$

called the *certain* and the *possible answers*, respectively. We now introduce the notion of *co-initial* sets of possible worlds. This notion will provide us with a characterization of when two incomplete instances are "certain-answer equivalent" for *monotone queries*. We will couple this with a characterization of "certain answer equivalence" with respect to *first order queries*.

DEFINITION 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets of possible worlds. Then we say that $\mathcal{X}$ **precedes** $\mathcal{Y}$, if for all $J \in \mathcal{Y}$, there is an $I \in \mathcal{X}$, such that $I \subseteq J$. This is denoted $\mathcal{X} \Subset \mathcal{Y}$. If both $\mathcal{X} \Subset \mathcal{Y}$, and $\mathcal{Y} \Subset \mathcal{X}$, we say that $\mathcal{X}$ and $\mathcal{X}$ are **co-initial**, and denote it $\mathcal{X} \simeq \mathcal{Y}$.*

It is easy to see that co-initiality is an equivalence relation. We now have two partial orders on sets of possible worlds, namely $\mathcal{X} \Subset \mathcal{Y}$, and the more obvious $\mathcal{X} \subseteq \mathcal{Y}$. Intuitively, $\mathcal{X} \Subset \mathcal{Y}$ means that $\mathcal{X}$ has *less* positive or monotone information than $\mathcal{Y}$. The order $\mathcal{X} \subseteq \mathcal{Y}$ means that $\mathcal{X}$ has *more* absolute information than $\mathcal{Y}$. This intuition is justified by the following theorem.

THEOREM 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be denumerable sets of finite instances. Then*

1. *$\mathcal{X} \Subset \mathcal{Y}$ ($\mathcal{X} \simeq \mathcal{Y}$) if and only if $cert_Q(\mathcal{X}) \subseteq cert_Q(\mathcal{Y})$ ($cert_Q(\mathcal{X}) = cert_Q(\mathcal{Y})$) for all monotone queries Q.*

2. *$\mathcal{X} \subseteq \mathcal{Y}$ ($\mathcal{X} = \mathcal{Y}$) if and only if $cert_Q(\mathcal{Y}) \subseteq cert_Q(\mathcal{X})$ ($cert_Q(\mathcal{Y}) = cert_Q(\mathcal{X})$) for all first order queries Q.*

The only if part of the first claim was proved in [24]. The if part of the first claim as well as the second claim is from [18]. Note that the class of monotone queries includes $\mathsf{UCQ}^{\neq}$, that is, unions of conjunctive queries with inequalities. It is well known that allowing inequalities leads coNP-completeness of query evaluation and other reasoning tasks (see e.g. [2]). A more practical and also popular notion is *UCQ-equivalence*, requiring only that the two sets of instances are indistinguishable as far as certain answers to unions of conjunctive queries are concerned. In order to characterize this notion we need a few extra concepts.

A mapping from $dom(I)$ to $dom(J)$ is said to be a *(instance) homomorphism* from an instance I to an instance J, if $R(a_1,\ldots,a_k) \in I$ implies that $R(h(a_1),\ldots,h(a_k)) \in J$, which we usually denote $h(I) \subseteq J$. Let $D \subseteq dom(I)$. If h is the identity on D we say that h is a D-homomorphism. An instance I *D-homomorphically precedes* an instance J, if there is a D-homomorphism h, such that $h(I) \subseteq J$. Let $\mathcal{X}$ be a denumerable set of finite instances. We say that $\bigcap\{dom(I) : I \in \mathcal{X}\}$ is the set of *visible constants* in $\mathcal{X}$.

Definition 2. *Let $\mathcal{X}$ and $\mathcal{Y}$ be denumerable sets of finite instances, and let D denote the set of visible constants in $\mathcal{X}$. We say that $\mathcal{X}$* ***homomorphically precedes*** *$\mathcal{Y}$ if for each $J \in \mathcal{Y}$ there exists an instance $I \in \mathcal{X}$ and a D-homomorphism h, such that $h(I) \subseteq J$. This is denoted $\mathcal{X} \leq \mathcal{Y}$. If both $\mathcal{X} \leq \mathcal{Y}$ and $\mathcal{Y} \leq \mathcal{X}$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are* ***homomorphically co-initial****, and denote it $\mathcal{X} \approx \mathcal{Y}$. Note that if $\mathcal{X}$ and $\mathcal{Y}$ are homomorphically co-initial, then they have the same set of visible constants.*

Finally, we say that a set $\mathcal{X}$ has *finite index* if it contains a finite number of isomorphically non-equivalent instances[1]. We can now give the promised characterization of UCQ-equivalence.

Theorem 2. *Let $\mathcal{X}$ and $\mathcal{Y}$ have finite index. Then $\mathcal{X} \leq \mathcal{Y}$ ($\mathcal{X} \approx \mathcal{Y}$) if and only if $cert_Q(\mathcal{X}) \subseteq cert_Q(\mathcal{Y})$ ($cert_Q(\mathcal{X}) = cert_Q(\mathcal{Y})$) for all UCQ-queries Q.*

Relational tableaux. Since an incomplete instance $\mathcal{X}$ in general contains infinitely many possible worlds, we need a mechanism to finitely represent them. The two most important such mechanisms are *tableaux* (naive tables) and *conditional tables* [24].

Relational tableaux are defined as follows. Let Vars be a countably infinite set, disjoint from the set of constants. Elements in Vars are called *variables*, and are denoted $X, Y, Z, \ldots$, possibly subscripted. We can then also allow *non-ground atoms*, i.e. expressions of the form $R(a, X, b, \ldots, X, Y)$. A *tableau* T is a finite set of atoms (ground, or non-ground). A non-ground atom represents a sentence where the variables are existentially quantified. The set of variables and constants in a tableau is denoted $dom(T)$.

Let T and U be tableaux. A mapping h from $dom(T)$ to $dom(U)$, that is the identity on Cons, is called a *(tableau) homomorphism* from T to U if $h(T) \subseteq U$. This is denoted $T \rightarrow U$. If $h(T) = U$ we denote it $T \hookrightarrow U$. If there is a homomorphism from T to U and a homomorphism from U to T we say that T and T are *homomorphically equivalent*, and denote it $T \leftrightarrow U$. If there is an injective homomorphism h such that $h(T) = U$, and $h^{-1}(U) = T$, we say that T and U are *isomorphic*, which we denote by $T \cong U$. An *endomorphism* on a tableau T is a mapping on $dom(T)$ such that $h(T) \subseteq T$. Finally, a *valuation* is a homomorphism whose image is contained in Cons.

There are two interpretations of a tableau T as a representative of incomplete instances, namely the *open world* (OWA) interpretation and the *closed world* (CWA) interpretation, denoted $Rep^{\mathsf{owa}}(T)$ and $Rep^{\mathsf{cwa}}(T)$, respectively. Formally:

$$
\begin{aligned}
Rep^{\mathsf{owa}}(T) &= \{I : v(T) \subseteq I \text{ for some valuation } v\},\\
Rep^{\mathsf{cwa}}(T) &= \{I : v(T) = I \text{ for some valuation } v\}.
\end{aligned}
$$

The *Rep*-functions are related to co-initiality, equality, and homomorphisms as follows.

Theorem 3. [21, 27] *Let T and U be tableaux. Then*

1. *$Rep^{\mathsf{owa}}(T) \leq Rep^{\mathsf{owa}}(U)$ iff $T \rightarrow U$, and*
   $Rep^{\mathsf{owa}}(T) \approx Rep^{\mathsf{owa}}(U)$ iff $T \leftrightarrow U$.

2. *$Rep^{\mathsf{cwa}}(T) \subseteq Rep^{\mathsf{cwa}}(U)$ iff $U \hookrightarrow T$, and*
   $Rep^{\mathsf{cwa}}(T) = Rep^{\mathsf{cwa}}(U)$ iff $T \cong U$.

Note that the set of visible constants in $Rep^{\mathsf{owa}}(T)$ is exactly the constants appearing in T, and that $Rep^{\mathsf{owa}}(T) \simeq Rep^{\mathsf{owa}}(U)$ iff $Rep^{\mathsf{owa}}(T) \approx Rep^{\mathsf{owa}}(U)$.

The *core* [15] of a tableau T is a tableau denoted $core(T)$, such that $core(T) \subseteq T$, $T \rightarrow core(T)$, and there is no endomorphism g, such that $g(core(T)) \subset core(T)$. The core is unique up to isomorphism. Note that $core(T) \cong core(U)$ iff $Rep^{\mathsf{owa}}(T) \approx Rep^{\mathsf{owa}}(U)$, but it can be that $core(T) \cong core(U)$ although $Rep^{\mathsf{cwa}}(T) \neq Rep^{\mathsf{cwa}}(U)$. For an example of the last point, let $T = \{R(a,b)\}$ and $U = \{R(a,b), R(a,X)\}$.

Representation systems. One of the salient features of the relational model, taken for granted these days, is that relations are closed under first order queries, meaning that the result of an FO-query on a relational instance is another relational instance. When it comes to incomplete databases we should expect the same. In their seminal paper [24], Imielinski and Lipski showed that for any tableau (naive table) T and query Q in UCQ, one can find a tableau U, such that $Rep^{\mathsf{owa}}(U) \approx Q(Rep^{\mathsf{owa}}(T))$. From this it follows that $cert_Q(Rep^{\mathsf{owa}}(T)) = Q^{\mathsf{naive}}(T)$, where $Q^{\mathsf{naive}}(T)$ is computed using "naive evaluation," meaning that the variables in T are treated as pairwise distinct constants, distinct from all constants in T, and that from the thus obtained result tuples with variables are removed. Imielinski and Lipski called the triple consisting of the class of naive tables (we use "naive table" and "tableau" interchangeably) with OWA-interpretation together with the class UCQ a *representation system.* In light of Theorem 2 above, we reformulate their notion of representation systems equivalently as follows.

[1] Instances I and J are isomorphic if there is an injective homomorphism h such that $h(I) = J$, and $h^{-1}(J) = I$.

Definition 3. *Let $\mathscr{T}$ be a class of tables, Rep a function that assigns a set of possible worlds to each $T \in \mathscr{T}$, and $\mathscr{Q}$ a class of queries. Then a triple $(\mathscr{T}, \mathit{Rep}, \mathscr{Q})$ is called a* ***homomorphic representation system****, if for each $T \in \mathscr{T}$ and $Q \in \mathscr{Q}$, there exists a table $U \in \mathscr{T}$, such that*

$$\mathit{Rep}(U) \approx Q(\mathit{Rep}(T)).$$

Let TBL denote the class of tableaux. The result of Imielinski and Lipski can now be stated as follows.

Theorem 4. [24] $(\mathsf{TBL}, \mathit{Rep}^{\mathsf{owa}}, \mathsf{UCQ})$ *is a homomorphic representation system.*

This means that the class of tableaux (naive tables) is closed (modulo $\approx$) under unions of conjunctive queries.

Strong representation systems. We now turn our attention to representation systems that allow for non-monotonic features such as full FO-queries and possible (maybe) answers. As can be expected, neither monotone nor homomorphic representation systems will be adequate. This can be seen from the following next two lemmas.

Lemma 1. *For any tableaux T (even without variables) and satisfiable conjunctive query Q with at least one negated atom, it holds that $\mathit{cert}_Q(\mathit{Rep}^{\mathsf{owa}}(T)) = \varnothing$.*

This means that the answer to any satisfiable conjunctive query with at least one negated atom consists of the empty instance. A similar phenomenon was recently also noted by Fagin et al. [17]. It seems that many of the anomalies stemming from applying non-monotonic queries in data exchange can be traced back to this mismatch.

If we also are interested in retrieving *possible answers* we run into a similar anomaly. The symbol ∞ stands for $\mathsf{Cons}^{arity(Q)}$.

Lemma 2. *For any naive table T and non-Boolean UCQ-query Q, we have $\mathit{poss}_Q(\mathit{Rep}^{\mathsf{owa}}(T)) = \infty$.*

It is easy too see that such phenomena cannot occur in closed world systems. This motivates the following definition.

Definition 4. *Let $\mathscr{T}$ be a class of tables, Rep a function that assigns a set of possible worlds to each $T \in \mathscr{T}$, and $\mathscr{Q}$ a class of queries. Then a triple $(\mathscr{T}, \mathit{Rep}, \mathscr{Q})$ is a* ***strong representation system****, if for each $T \in \mathscr{T}$ and $Q \in \mathscr{Q}$, there exists a table $U \in \mathscr{T}$, such that*

$$\mathit{Rep}(U) = Q(\mathit{Rep}(T)).$$

It is well known [24, 1] that there are first order queries Q and tableaux T, such that there is no tableau U with $\mathit{Rep}^{\mathsf{cwa}}(U) = Q(\mathit{Rep}^{\mathsf{cwa}}(T))$. It turns out that we need *conditional tables* in order to obtain a strong representation system for first order queries. Formally, a conditional table [24] is a pair (T, φ), where T is a tableau, and φ is a mapping that associates a *local condition* $\varphi(t)$ with each tuple $t \in T$. A (local) condition is a Boolean formula built up from atoms of the form $X = Y$, $X = a$, and $a = b$ for $X, Y \in \mathsf{Vars}$, and $a, b \in \mathsf{Cons}$. An atomic equality of the form $a = a$, for $a \in \mathsf{Cons}$ is interpreted as "true," which we will denote $\top$. Likewise, for any two distinct constants a and b, the equality $a = b$ is interpreted as "false," denoted $\bot$. A conditional table (T, φ) *represents* a set of possible worlds (ground instances, complete databases). For this, let v be a *valuation*, that is, a ground homomorphism. Then

$$v(T, \varphi) = \{v(t) \;:\; t \in T, \text{ and } v\big(\varphi(t)\big) = \top\}.$$

The set of possible worlds represented by (T, φ) under the closed world and open world interpretations are

$$\begin{array}{lcl} \mathit{Rep}^{\mathsf{cwa}}(T, \varphi) & = & \{I : I = v(T, \varphi) \text{ for some valuation } v\}, \\ \mathit{Rep}^{\mathsf{owa}}(T, \varphi) & = & \{I : I \supseteq v(T, \varphi) \text{ for some valuation } v\}. \end{array}$$

Below is an example of a conditional table (T, φ) displayed in tabular form:

t	$\varphi(t)$
$R(X, b)$	$X = Y$
$R(b, c)$	$\top$
$R(Y, d)$	$X = d$
$T(X, c)$	$\top$

In "in-line" notation we would write above table (T, φ) as $\{R(X, b); X = Y,\ R(b, c); \top,\ R(Y, d); X = d,\ T(X, c); \top\}$. Below are some instances from $\mathit{Rep}^{\mathsf{cwa}}(T, \varphi)$. For example, I_1 is obtained by considering valuation v where $v(X) = v(Y) = a$.

I_1	I_2	I_3	I_4
$R(a, b)$	$R(d, b)$	$R(b, c)$	$R(b, c)$
$R(b, c)$	$R(b, c)$	$R(a, d)$	$T(a, c)$
$T(a, c)$	$R(d, d)$	$T(d, c)$	
	$T(d, c)$		

Imielinski and Lipski [24] showed that conditional tables under closed world interpretation is a strong representation for FO-queries. We denote the class of conditional tables as COND.

Theorem 5. [24] $(\mathsf{COND}, \mathit{Rep}^{\mathsf{cwa}}, \mathsf{FO})$ *is a strong representation system.*

3. ADDING MAPPINGS

We now extend the notion of representation systems to also account for *data exchange mappings* [13]. First we need to review a few concepts.

Data exchange mappings and systems. The concept of mappings between schemas was introduced by Bernstein et al. in [8], and in the relational context by Fagin et al. in their seminal paper [13]. Let $\mathbf{R}$ and $\mathbf{S}$ be two disjoint schemas, which we call the *source schema* and the *target schema*, respectively. A *data exchange mapping* M from source $\mathbf{R}$ to target $\mathbf{S}$ is a subset of $\mathit{Inst}(\mathbf{R}) \times \mathit{Inst}(\mathbf{S})$. The intended meaning of M is that a source instance I is mapped to the set of instances, also called *solutions*,

$$\mathit{Sol}_M(I) = \{J : (I, J) \in M\}.$$

In the sequel we shall, depending on the context, regard a mapping M as a function from source instances to sets of target instances, or as a relation between source and target instances.

Given a mapping M from source $\mathbf{R}$ to target $\mathbf{S}$, a source instance I over $\mathbf{R}$, and a query Q over $\mathbf{S}$ (a target query), the *certain answer to query Q on I through M* is defined as

$$cert_{Q,M}(I) = \bigcap\{Q(J) : J \in Sol_M(I)\}.$$

One would therefore be interested in scenarios where the target instances can be (preferably efficiently) materialized by an approximation that gives the correct certain answer for all target queries. We are thus led to the following definition.

DEFINITION 5. *Let $\mathscr{T}$ be a class of tables, and Rep a function that assigns a set of possible worlds to each $T \in \mathscr{T}$. Let $\mathscr{Q}$ be a class of queries, and $\mathscr{M}$ a class of mappings. Then a quadruple $(\mathscr{T}, Rep, \mathscr{Q}, \mathscr{M})$ is said to be a* ***homomorphic data exchange system*** *if for each $T \in \mathscr{T}, M \in \mathscr{M}$, and $Q \in \mathscr{Q}$, there exists a table $U \in \mathscr{T}$, such that*

$$Rep(U) \approx \{Q(J) : J \in Sol_M(I),\ I \in Rep(T)\}.$$

Embedded dependencies. A mapping M has to be specified (finitely) using some formalism. The *embedded dependencies* [11] are the class of choice. Embedded dependencies consist of tuple generating dependencies and equality generating dependencies. A *tuple generating dependency (tgd)* is a first order formula of the form

$$\forall \bar{x}, \bar{y} : \alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z}),$$

where α and β are conjunctions of relational atoms, and $\bar{x}$,$\bar{y}$ and $\bar{z}$ are sequences of variables. We assume that the variables occurring in dependencies are disjoint from all variables occurring in any tableaux under consideration. To emphasize this, we use lowercase letters for variables in dependencies. When α is the antecedent of a tgd, we shall sometimes conveniently regard the set of atoms in it as a tableau, and refer to it as the *body* of the tgd. Similarly we refer to β as the *head* of the tgd. If there are no existentially quantified variables the dependency is said to be *full*, otherwise it is *embedded*. In case a body of a tgd contains only one atom, it is called a *Local-As-View tgd (LAV-tgd)*. If a LAV-tgd does not contain repeated variables in the body it is called a *true-LAV tgd*.

Frequently, we omit the universal quantifiers in tgd formulae. Also, when the variables are not relevant in the context, we denote a tgd $\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z})$ simply as $\alpha \to \beta$.

Let $\xi = \alpha \to \beta$ be a tuple generating dependency, and I an instance. Then we say that I *satisfies* ξ, if $I \vDash \xi$ in the standard model theoretic sense, or equivalently, if for every homomorphism h, such that $h(\alpha) \subseteq I$, there is an extension h' of h, such that $h'(\beta) \subseteq I$.

An *equality generating dependency (egd)* is a first order formula of the form

$$\forall \bar{x} : \alpha(\bar{x}) \to y = z$$

where α is a conjunction of relational atoms, $\bar{x}$ a sequence of variables, and y, z two variables from $\bar{x}$.

We say that an instance I *satisfies* an egd ξ, if $I \vDash \xi$ in the model theoretic sense, or equivalently, if for every homomorphism h, such that $h(\alpha) \subseteq I$, it holds that $h(x) = h(y)$.

In order to use embedded dependencies as mappings, we need some definitions. Let $\mathbf{R}$ and $\mathbf{S}$ be two disjoint schemas. A tgd $\alpha \to \beta$ is said to be a *a source-to-target tgd* (*st-tgd*) from $\mathbf{R}$ to $\mathbf{S}$ if all the relation symbols in α belong to $\mathbf{R}$, and all the relation symbols in β belong to $\mathbf{S}$, A *target tgd* is a tgd where all relation symbols are from $\mathbf{S}$. *Target egd's* are defined similarly. We abbreviate target dependencies with t-tgd's and t-egd's, respectively. Let $\Sigma = \Sigma_{st} \cup \Sigma_t$, where Σ_{st} is a (finite) set of st-tgd's, Σ_t is a (finite) set of t-tgd's and t-egd's. The mapping specified by Σ is

$$M_\Sigma = \{(I, J) \in Inst(\mathbf{R}) \times Inst(\mathbf{S}) : (I, J) \vDash \Sigma\}.$$

The Chase on tableaux. Let Σ be a set of embedded dependencies and T a tableau. A *trigger* for Σ on T is a pair (ξ, h), where $\xi = \alpha \to \beta \in \Sigma$ is a tgd, and h is a homomorphism such that $h(\alpha) \subseteq T$ and for no extension h' of h, does it hold that $h'(\beta) \subseteq T$. The *standard chase step* [13, 10] on a tableau T with such a trigger results in a tableau $U = T \cup \{h'(\beta)\}$, where h' is an extension of h that assigns new fresh (uppercase) variables to the existential variables in β. In case ξ is an egd $\alpha(\bar{x}) \to y = z$, the pair (ξ, h) is called a trigger, if $\alpha(h(\bar{x})) \subseteq T$, and $h(y) \neq h(z)$. In this case the standard chase step on tableau T with such a trigger results in the tableau obtained from T by substituting each occurrence of one of $\{h(y), h(z)\}$ with the other, following the rule that the variable is substituted with the constant, or in case both are variables, then X_j is substituted with X_i, if $i < j$ in some fixed enumeration of Vars. In case both of $h(y)$ and $h(z)$ are constants, the chase step results in an inconsistent state, denoted $\perp$. The *standard chase* on a tableau T is defined as a sequence $T_0, T_1, T_2, \ldots, T_n, \ldots$ where $T_0 = T$, and for each i we have that T_{i+1} was obtained from T_i in one chase step. If a chase step ever results in $\perp$, the sequence is said the be *unsuccessful*, and the process aborted. A successful process, on the other hand, might not terminate. However if there is an integer i such that there is no trigger for T_i, the chase is said to terminate. In this case we call the tableau T_i the *result* of the chase, and denote it $Chase_\Sigma(T)$.

Fagin et al. [13] showed that when Σ is a set of embedded dependencies, I an instance, and $Chase_\Sigma(I)$ terminates, then

$$Rep^{\mathsf{owa}}(Chase_\Sigma(I)) \approx Sol_{M_\Sigma}(Rep^{\mathsf{owa}}(I)).$$

This fundamental result was later generalized in [17] to allow the input to be tableaux.

Weak acyclicity and beyond. Deutsch et al. [10] showed that it is undecidable whether the chase terminates with a set Σ of embedded dependencies on an instance I. Several classes of dependencies ensuring termination of the chase have been defined. The most important such classes are the weakly acyclic sets of tgd's [13], and their subclass called richly acyclic tgd's [23]. In the following we define these concepts.

For a set Σ of tgd's over a database schema $\mathbf{R}$ the *dependency graph* of Σ is the directed graph that has as vertices (R, i), where $R \in \mathbf{R}$, and $i \in \{1, \ldots, arity(R)\}$. The graph has two types of edges:

1. *Regular edges.* There is a regular edge between vertices (R, i) and (S, j) if there is a tgd in Σ that has a variable y that appears both in position (R, i) in the body, and in position (S, j) in the head.

2. *Existential edges.* There is an existential edge between vertices (R, i) and (S, j) if there is a tgd in Σ that has a variable y that appears in position (R, i) of the body and in some position in the head, and an existentially quantified variable z that appears in position (S, j) in the head.

A set Σ of tgd's is said to be *weakly acyclic* if the dependency graph of Σ does not have any cycles containing an existential edge [13].

Hernich and Schweikardt [23] defined a slight restriction on the weakly acyclic sets of dependencies as follows: If the previous dependency graph is extended by also adding an existential edge from position (R, i) and (S, j) whenever there is a dependency in Σ that has a variable x that appears in position (R, i) of the body and an existentially quantified variable z appears in position (S, j) in the head. The newly created graph is called *extended dependency graph.* A set Σ of tgd's is said to be *richly acyclic* if the extended dependency graph of Σ does not have any cycles containing an existential edge.

Let STAND be the class of mappings defined by M_Σ, where Σ consists of source-to-target tgd's, a weakly acyclic set of target tgd's, and target egd's. We can now integrate data exchange and representation systems according to the following theorem.

Theorem 6. [13, 17] $(\mathsf{TBL}, Rep^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{STAND})$ *is a homomorphic data exchange system.*

Second order dependencies. The class of st-tgd's was generalized by Fagin et al. in [16] to *Second Order* tgd's (SO-tgd's) which are source-to-target tgd's extended with existentially quantified function symbols, terms using these function symbols, and atomic facts over terms. Equalities between terms are allowed in the body. The function symbols act as Skolemized existential quantifiers. Examples of SO-tgd's can be found in Example 1. We refer to the Appendix and to [16] for a formal definition of SO-tgd's. Note that SO-tgd's are source-to-target, by their definition. The SO-tgd's emerged as the natural closure of st-tgd's under composition of mappings, a topic we return to in the next section. It was shown in [16] that the chase can be generalized to compute (in polynomial time) a tableau $Chase_\Sigma(T)$, such that

$$Rep^{\mathsf{owa}}(Chase_\Sigma(T)) \approx Sol_{M_\Sigma}(Rep(T)),$$

when Σ consists of SO-tgd's. Let normSO be the class of mappings defined by sets of SO-tgd's, a weakly acyclic set of target tgd's and target egd's. Then the previous theorem can be strengthened as follows.

Theorem 7. [16, 17] $(\mathsf{TBL}, Rep^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{normSO})$ *is a homomorphic data exchange system.*

Later Arenas et al. showed in [5] that if one is only concerned with UCQ-equivalence, then every set of SO-tgd's can be written in a simplified form, called *plain* SO-tgd's, while preserving UCQ-equivalence. Plain SO-tgd's are SO-tgd's where equalities over terms are disallowed. Let simSO be the class of mappings defined by M_Σ, where Σ is a set of plain SO-tgd's, a weakly acyclic set of t-tgd's, and t-egd's. We can thus replace the class normSO with the smaller class simSO, since the result of Arenas et al. tells us that for any mapping M_Σ, where Σ is a set of SO-tgd's there is a mapping $M_{\Sigma'}$, where Σ' is a set of plain SO-tgd's, such that $Sol_{M_\Sigma}(I) \approx Sol_{M_{\Sigma'}}(I)$, for all instances I. This gives us the following "canonical" open world data exchange system.

Theorem 8. [16, 17, 5] $(\mathsf{TBL}, Rep^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{simSO})$ *is a homomorphic data exchange system.*

This leaves open the question of the usefulness of the full power of SO-tgd's. Consider the following example.

Example 1. *Let the source schema have two relations Emp (employees) and OutWorker (outsourced workers). Let the target schema have relations EmpMgr (employees with their managers), SelfMgr (employees that are self managers) and Consult (consultants). Consider also the source to target dependencies:*

$$\begin{aligned}\Sigma \;=\; \{&OutWorker(x) \to Consult(x),\\ &Emp(e) \to EmpMgr(e, f(e)),\\ &Emp(y), y = f(y) \to SelfMgr(y)\}.\end{aligned}$$

Note that Σ is a set of SO-tgd's. Technically, Σ should be read as $\exists f : \bigwedge\Sigma$, where $\bigwedge\Sigma$ is the conjunction of the sentences in Σ, and all first order variables and quantification inside Σ are local to each sentence, see [16]. Intuitively, the dependencies specify that all outsourced workers are consultants, each employee has at least one manager and each employee who is his own manager is a self-manager. Let us consider instance $I = \{Emp(joe), OutWorker(ann)\}$. Then $Chase_\Sigma(I) = \{EmpMgr(joe, Z), Consult(ann)\}$. Now the UCQ-equivalent set of plain SO-tgd's is

$$\begin{aligned}\Sigma' \;=\; \{&OutWorker(y) \to Consult(y),\\ &Emp(x) \to EmpMgr(x, f(x))\},\end{aligned}$$

and $Chase_{\Sigma'}(I) = Chase_\Sigma(I)$. In any case, the assumption that only employees can be self-managers, that is "ann" can't be her own self-manager, is lost. More formally, let Q be the query $\{x : Consult(x) \wedge \neg SelfMgr(x)\}$ Then the set of certain answers is $cert_{Q,M_\Sigma}(I) = \emptyset$, even if one may expect that $cert_{Q,M_\Sigma}(I) = \{ann\}$.

We will show that if we allow conditional tables instead of tableaux as representation mechanism, and suitably extend the chase, the result of applying the extended chase gives the conditional table

t	$\varphi(t)$
$EmpMgr(joe, Z)$	$\top$
$SelfMgr(joe)$	$Z = joe$
$Consult(ann)$	$\top$

Materializing the target as this conditional table with closed world interpretation will correctly give us $\{ann\}$ as the certain answer for Q.

Constructible solutions. Before admitting conditional tables into data exchange, we need to determine an appropriate closed world assumption and know how to apply it to mappings. This was recently solved for source-to-target tgd's by Libkin [26], who proposed the *closed world solutions* for data exchange. The approach of Libkin is based on the intuition that any (existential) facts in the target instance should follow logically from the source instance and the dependencies, and that no two nulls in the target should be gratuitously equated. Libkin's solution was subsequently generalized to also include target tgd's and egd's [23, 22]. It was however observed in [21] that these generalizations suffered from some anomalies, and a new closed world semantics, called *constructible solutions* was proposed. This is the closed world semantics we adopt here.

The constructible solutions semantics can intuitively be described as follows. Let I be a ground instance and Σ a set of tgd's. Consider a chase step on I with a tgd $\xi = \alpha(\bar{x},\bar{y}) \to \exists\bar{z}\ \beta(\bar{x},\bar{z})$ and trigger (ξ,h). Instead of adding the tuple $\beta(h(\bar{x}),\bar{Z}')$, where $\bar{Z}'$ is a sequence of new fresh variables, we create a branch for each sequence $\bar{c}$ of constants and add the ground tuple $\beta(h(\bar{x}),\bar{c})$ to I. Continuing in this fashion we will have a *chase tree*, instead of a chase sequence. (There is a caveat that each trigger (ξ,h) is fired only once in each branch.) Notably, all nodes in the tree will be ground instances. The leaves (or limits of branches) of the chase tree form the set of constructible solutions for I and Σ, denoted $\Sigma(I)$. A formal definition of $\Sigma(I)$ can be found in [21]. If $\Sigma = \Sigma_{tgd} \cup \Sigma_{egd}$ we define $\Sigma(I) = \Sigma_{tgd}(I) \cap Sat(\Sigma_{egd})$, where $Sat(\Sigma)$ denotes the set of all instances I, such that $I \vDash \Sigma$.

We still need to extend the constructible solutions to the extended class *st-SO dependencies* [3], that is SO-tgd's where equalities over functional terms also can occur in the head of the tgd's. Let Σ be a set of st-SO dependencies, and F the set of function symbols therein. Let $F^\flat$ be an interpretation of F in the universe Cons.

Given an instance I, we first create a subtree for each interpretation $F^\flat$. In each subtree $F^\flat$ we construct $\Sigma(I)$. In doing this, when applying a homomorphism h we of course use the rule $h(f(\bar{t})) = f^\flat(h(\bar{t}))$. Note that even if we are not instantiating existential variables we still get a three due to the fact that the chase is non-deterministic. At each point, we get a subtree for each choice of trigger (ξ,h).

Definition 6. *Let Σ be a set of st-SO tgd's, target tgd's and egd's. The* ***closed world mapping*** *defined by Σ is*

$$M_\Sigma^{\mathsf{cwa}} = \{(I,J) : J \in \Sigma(I)\}.$$

Consequently we have $Sol_{M_\Sigma^{\mathsf{cwa}}}(I) = \{J : J \in \Sigma(I)\}$.

We also define the certain and possible CWA-answers.

Definition 7.

$$\begin{array}{rcl} cert_{Q,M_\Sigma^{\mathsf{cwa}}}(I) & = & \bigcap\{Q(J) : J \in Sol_{M_\Sigma^{\mathsf{cwa}}}(I)\} \\ poss_{Q,M_\Sigma^{\mathsf{cwa}}}(I) & = & \bigcup\{Q(J) : J \in Sol_{M_\Sigma^{\mathsf{cwa}}}(I)\} \end{array}$$

We can now formulate the requirements for a closed world data exchange system.

Definition 8. *Let $\mathscr{T}$ be a class of tables, Rep a function that assigns a set of possible worlds to each $T \in \mathscr{T}$. Let $\mathscr{Q}$ be a class of queries and $\mathscr{M}$ a class of mappings. Then a quadruple $(\mathscr{T}, Rep, \mathscr{Q}, \mathscr{M})$ is a* ***strong data exchange system****, if for each $T \in \mathscr{T}$, $M_\Sigma \in \mathscr{M}$, and $Q \in \mathscr{Q}$, there exists a table $U \in \mathscr{T}$, such that*

$$Rep(U) = \{Q(J) : J \in Sol_{M_\Sigma^{\mathsf{cwa}}}(I),\ I \in Rep(T)\}.$$

To be able to obtain strong data exchange systems, we will extend the chase from tableaux to conditional tables. This *conditional chase* requires that we generalize the notion of a trigger. Recall that for tableau T and a tgd $\xi = \alpha \to \beta$, a trigger is a pair (ξ,h), where h is a homomorphism such that $h(\alpha) \subseteq T$. For a conditional table (T,φ) we are looking for a subset T' of T, and mappings h_1 and h_2, such that $h_1(\alpha) = h_2(T')$. The most general such pair is called an *mgu* (most general unifier). A *trigger* is then a triple $(\xi,(h_1,h_2),T')$. Applying a trigger will generate new tuples $h_1(\beta)$ each with a local condition determined by h_2 and the local conditions from T'. For example, if $T = \{R(a,Y); Z = 1, R(b,c); \top\}$ and $\xi = R(x,x) \to S(x)$, then $(\xi,(\{x/a\},\{Y/a\},\{R(a,Y)\}$ is a trigger. Applying this trigger results in the addition of the tuple $S(a)$ with local condition $Z = 1 \wedge Y = a$. For details, see [21].

In order to account for egd's as well, we need to extend the notion of conditional tables by allowing a set of global conditions [18]. A global condition is syntactically like a local condition, but it is used to restrict the set of possible worlds, rather that restricting the set of tuples within a possible world as the local conditions do. For a conditional table (T,φ) we denote the global condition with $\varphi(T)$. We can now extend the open world and closed world interpretations of conditional tables as follows:

$$\begin{array}{l} Rep^{\mathsf{owa}}(T,\varphi) = \{I : I \supseteq v(T,\varphi) \text{ for } v \text{ such that } v(\varphi(T)) = \top\} \\ Rep^{\mathsf{cwa}}(T,\varphi) = \{I : I = v(T,\varphi) \text{ for } v \text{ such that } v(\varphi(T)) = \top\} \end{array}$$

The global condition is needed in order to enforce equalities in the representation function. For example, let $(T,\varphi) = \{R(a,X,c); \top,\ R(a,b,d); \top\}$, and $\xi = R(x,y,z), R(x,y',z') \to y = y'$. Then ξ can be enforced in the chase by adding the global condition $\varphi(T) = \{X = b\}$. If we instead have had $(T,\varphi) = \{R(a,b,c); \top,\ R(X,Y,d); \top\}$, we would have added $\varphi(T) = \{X \neq a \vee Y = b\}$ instead.

We now have the ingredients for a strong data exchange systems. Let ordSO be the class of mappings M_Σ^{cwa}, where Σ consists of SO-tgd's, a richly acyclic set of target dependencies, and target egd's. Let GCOND be the class of global conditional tables, and FO the class of first order queries. We then have

Theorem 9. $(\mathsf{GCOND}, Rep^{\mathsf{cwa}}, \mathsf{FO}, \mathsf{ordSO})$ *is a strong data exchange system.*

We note that Arenas et al. [4] showed that for any "positive" conditional table (T,φ), where the global condition is $\top$ and all local conditions are conjunctions of equalities, and set Σ of st-SO dependencies, one can find a positive conditional table (U,ψ), such that

$$Rep^{\mathsf{owa}}(U,\psi) = \bigcup\{Sol_\Sigma(I) : I \in Rep^{\mathsf{owa}}(T,\varphi)\}.$$

This would yield a strong data exchange system, when coupled with target UCQ's, except that the same power can be achieved using naive tables. To see this, note that for any UCQ Q, we have

$$cert_Q(Rep^{\mathtt{owa}}(U,\psi)) = Q^{\mathtt{naive}}(U).$$

The proof of Theorem 9, omitted here due to space restrictions, is based on the fact that the chase procedure can be generalized to the conditional chase, denoted $Chase^{\mathtt{cond}}$, with the property

$$Rep^{\mathtt{cwa}}(Chase^{\mathtt{cond}}_{\Sigma}(T,\varphi)) = \bigcup\{\Sigma(I) : I \in Rep^{\mathtt{cwa}}(T,\varphi)\},$$

for all global conditional tables (T,φ) and finite sets Σ from ordSO.

The rich acyclicity condition in the ordSO definition ensures that the new conditional chase terminates. Here we illustrate the conditional chase process by an example. Consider the source-to-target SO dependency $\exists f\ \exists g\ \exists k : \bigwedge\Sigma$, where

$$\begin{aligned}\Sigma \;\; = \;\; & \{R(x,y), R(y,z), z = f(x,y) \;\rightarrow\; S(x,y,g(x,y)),\\ & R(y,y), x = f(y,y) \;\rightarrow\; x = k(y)\}\end{aligned}$$

Let (T,φ) be the following conditional table:

t	φ
$R(a,X)$	$\top$
$R(b,c)$	$\top$
$R(d,d)$	$Y = b$

The conditional chase process for the given st-SO dependency, consists of conditionally chasing each implication from Σ separately. First we chase the conditional table with the implication $R(x,y), R(y,z), z = f(x,y) \;\rightarrow\; S(x,y,g(x,y))$. To achieve this we first apply the conditional chase steps described in [21] with the dependency obtained by eliminating the equalities from the body of the implication, that is the dependency

$$R(x,y), R(y,z) \;\rightarrow\; S(x,y,g(x,y)).$$

These three mgu's match tuples of the conditional table with the body of the dependency: $(\{x/a, y/b, z/c\}, \{X/b\})$, $(\{x/a, y/d, z/d\}, \{X/d\})$, and $(\{x/d, y/d, z/d\}, \{\})$.

The first mgu will generate conditional tuple $S(a,b,g(a,b))$ with local condition $X = b$. To the local condition of this tuple we add the image of the eliminated equality $z = f(x,y)$ under the mapping $\{x/a, y/b, z/c\}$, namely $c = f(a,b)$. By applying the triggers corresponding to these three mgu's we obtain the following conditional table (T_1,φ_1):

t	$\varphi_1(t)$
$R(a,X)$	$\top$
$R(b,c)$	$\top$
$R(d,d)$	$Y = b$
$S(a,b,g(a,b))$	$X = b \wedge (c = f(a,b))$
$S(a,d,g(a,d))$	$Y = b \wedge X = d \wedge (d = f(a,d))$
$S(d,d,g(d,d))$	$Y = b \wedge (d = f(d,d))$

Next, we move to the second implication

$$\xi \;\; = \;\; R(y,y), x = f(y,y) \;\rightarrow\; x = k(y)$$

in Σ. Note that ξ is equality generating, and will therefore affect the global condition only. The sole trigger for ξ is

$$(\xi, (\{x/a, y/d\}, \{X/d\}), \{R(a,X), R(d,d)\}).$$

The equality in the head of ξ will create a condition that needs to be satisfied whenever the body is satisfied. In our case the body is satisfied when the local condition $Y = b$ and the condition $X = d$ from the mgu of the trigger, as well as the equality $x = f(y,y)$ from the body of ξ is satisfied. This will create the following global condition that needs to be added to the resulting conditional table: $(Y = b) \wedge (X = d) \wedge (a = f(d,d)) \;\rightarrow\; a = k(d)$. The resulting target conditional table, with terms, can be represented as (T_2,φ_2):

$\varphi_2(T): \quad (Y = b) \wedge (X = d) \wedge (a = f(d,d)) \rightarrow a = k(d)$

t	$\varphi_2(t)$
$R(a,X)$	$\top$
$R(b,c)$	$\top$
$R(d,d)$	$Y = b$
$S(a,b,g(a,b))$	$X = b \wedge (c = f(a,b))$
$S(a,d,g(a,d))$	$Y = b \wedge X = d \wedge (d = f(a,d))$
$S(d,d,g(d,d))$	$Y = b \wedge (d = f(d,d))$

Finally, we eliminate all Skolem functions by replacing each distinct Skolemized term with a new fresh variable[2] yielding the following conditional table (T_2,φ_2):

$\varphi_3(T): \quad (Y = b) \wedge (X = d) \wedge (a = Z_1) \rightarrow a = Z_2$

t	$\varphi_3(t)$
$R(a,X)$	$\top$
$R(b,c)$	$\top$
$R(d,d)$	$Y = b$
$S(a,b,Z_3)$	$X = b \wedge (c = Z_4)$
$S(a,d,Z_5)$	$Y = b \wedge X = d \wedge (d = Z_6)$
$S(d,d,Z_7)$	$Y = b \wedge (d = Z_1)$

4. COMPOSABLE SYSTEMS

Data exchange is naturally not limited only to one mapping and source-target pair. If the target schema is a source schema for a second mapping, it is of interest to be able to compose the two mappings into a third one that goes directly from the source of the first mapping to the target of the second mapping. Such a composition is defined formally as follows.

Definition 9. *Let $M_{12} \subseteq Inst(\mathbf{R}_1) \times Inst(\mathbf{R}_2)$ and $M_{23} \subseteq Inst(\mathbf{R}_2) \times Inst(\mathbf{R}_3)$ be mappings, where $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$ are pairwise disjoint schemas. Then M_{12} and M_{23} are said to be* ***composable*** *and their* ***composition*** *is*

$$\begin{aligned} M_{12} \circ M_{23} \;\; = \;\; & \{(I,J) \in Inst(\mathbf{R}_1) \times Inst(\mathbf{R}_3) \; : \\ & (I,K) \in M_{12} \text{ and } (K,J) \in M_{23}, \\ & \text{for some } K \in Inst(\mathbf{R}_2)\}. \end{aligned}$$

As emphasized by several authors [16, 29, 7, 3], it is desirable that a class of mappings should be closed under composition, as in the following definition.

[2] If the table contains nested Skolem terms, we need to take into account their possible interdependence similarly to [3]. For example, if $t_1 = f(a,b)$ and $t_2 = f(a,g(d))$, then in case $g(d) = b$, necessarily $t_1 = t_2$. If we now replace t_1 with x, t_2 with y, and $g(d)$ with z, we need to add $z = b \rightarrow x = y$ to the global condition.

Definition 10. *Let $\mathscr{M}$ be a class of mappings. We say that $\mathscr{M}$ is* ***closed under composition****, if for any two composable mappings $M_{12}, M_{23} \in \mathscr{M}$, there exists a mapping $M_{13} \in \mathscr{M}$ such that $M_{13} = M_{12} \circ M_{23}$.*

Recently there has been significant advances is finding classes of mappings that are closed under composition. The largest hitherto known such class is standSO [3], which is the class of mappings M_Σ, where Σ consists of an st-SO dependency, a weakly acyclic set of target tgd's, and target egd's.

In the next two subsections we shall see how to obtain homomorphic and strong data exchange systems that compose.

4.1 Composable homomorphic systems

As mentioned in connection with Theorems 7 and 8, using full SO-tgd's in a homomorphic data exchange system is an "over-kill," since Arenas et al. showed in [5] that for any such mapping M there is a mapping M' in the smaller class plainSO such that $Sol_M(I) \approx Sol_{M'}(I)$, for all instances I. By plainSO we mean the class of mappings M_Σ where Σ is a set of plain SO-tgd's (no equalities between function terms). This motivates the following definitions.

Definition 11. *A class of mappings $\mathscr{M}$ is said to be* ***weakly closed under composition*** *if for any two composable mappings $M_{12}, M_{23} \in \mathscr{M}$, there exists a mapping $M_{13} \in \mathscr{M}$, such that*

$$Sol_{M_{13}}(I) \approx Sol_{M_{23}}(Sol_{M_{\Sigma_{12}}}(I)),$$

for all instances I.

Definition 12. *A quadruple $(\mathscr{T}, Rep, \mathscr{Q}, \mathscr{M})$ is said to be a* ***composable homomorphic exchange system*** *if $(\mathscr{T}, Rep, \mathscr{Q}, \mathscr{M})$ is a homomorphic data exchange system and $\mathscr{M}$ is weakly closed under composition.*

The aforementioned result in [5] thus says that plainSO is a weakly composable class. It gives us

Theorem 10. [5] $(\mathsf{TBL}, Rep^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{plainSO})$ *is a composable homomorphic data exchange system.*

Example 2. *Consider the following example from* [16].

$$\begin{aligned}
\Sigma_{12} &= \{Emp(e) \to \exists m\ Mgr_1(e,m)\}\\
\Sigma_{23} &= \{Mgr_1(e,m) \to Mgr(e,m),\\
&\qquad Mgr_1(e,e) \to SelfMgr(e)\}\\
\sigma_{13} &= \{Emp(e) \to Mgr(e,f(e)),\\
&\qquad Emp(e), e = f(e) \to SelfMgr(e))\}\\
\Sigma_{13} &= \{Emp(e) \to Mgr(e,f(e))\}
\end{aligned}$$

Note that σ_{13} is an SO-tgd, while the tgd in Σ_{13} is plain SO. On input $\Sigma_{12} \cup \Sigma_{23}$ the result returned by the composition algorithm in [16] *is σ_{13}, while the transformation algorithm in* [5] *returns Σ_{13}. Note that $M_{\sigma_{13}} = M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$, and that*[3] *$M_{\Sigma_{13}} \approx M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$.*

4.2 Composable strong systems

In the previous section we considered the case when data exchange and query answering was modulo $\approx$. If we now want to admit queries in FO, we need to move to a closed world environment in order to avoid the pitfalls of Lemmas 1 and 2. We therefore need to interpret the dependencies Σ as $M^{\mathsf{cwa}}_\Sigma = \{(I,J) : J \in \Sigma(I)\}$, as in Definition 6. We then have to be careful when composing Σ_{12} with Σ_{23}. The composition mapping we wish to obtain is $M^{\mathsf{cwa}}_{\Sigma_{12}} \circ M^{\mathsf{cwa}}_{\Sigma_{23}}$, which is different from $M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$. An example of this will follow.

Definition 13. *A class $\mathscr{M}$ of mappings that are specified by a set of embedded dependencies from a class $\mathscr{D}$, is said to be* ***strongly closed under composition*** *if for all composable Σ_{12} and Σ_{23} in $\mathscr{D}$, there exists a $\Sigma_{13} \in \mathscr{D}$, such that*

$$M^{\mathsf{cwa}}_{\Sigma_{13}} = M^{\mathsf{cwa}}_{\Sigma_{12}} \circ M^{\mathsf{cwa}}_{\Sigma_{23}}.$$

We can now state the criteria for strong composable data exchange systems.

Definition 14. *A system $(\mathscr{T}, Rep, \mathscr{Q}, \mathscr{M})$ is said to be a* ***composable strong data exchange system****, if it is a strong data exchange system and $\mathscr{M}$ is strongly closed under composition.*

It follows from [21] that if Σ is a set of source-to-target tgd's, M^{cwa}_Σ coincides with the CWA notion introduced by Libkin in [26]. Later Libkin and Sirangelo [28] showed that the class of mappings specified by SO-tgd's (source-to-target only) is strongly closed under composition. It would be very important to generalize this strong closure property to larger classes, such as the sets of dependencies used in the definition of the mapping class standSO, namely sets consisting of source-to-target SO-tgd's, a weakly acyclic set of target tdg's, and target egd's. Unfortunately, as shall see below, this class is not strongly closed under composition, even though it is closed under composition. We are thus forced to add some mild restrictions to the class standSO dependencies, yielding the class of ordSO defined in Section 3. The following theorem assures us that the mild restriction works.

Theorem 11. *The class* ordSO *is strongly closed under composition.*

Next we describe, based on an example, how the strong composition is constructed in the proof of Theorem 11. Consider the following sets of dependencies:

$$\begin{aligned}
\Sigma_{12} &= \{S(x) \to R(f(x),x)\}\\
\Sigma_2 &= \{R(x,y) \to T(x),\quad T(x) \to \exists y\, R(x,y)\}\\
\Sigma_{23} &= \{R(x,y) \to V(x,y,g(x,y))\}\\
\Sigma_3 &= \{V(x,y,x) \to \exists z\, V(y,z,y)\}
\end{aligned}$$

We now construct dependency sets Σ_{13} and Σ_3, such that $M^{\mathsf{cwa}}_{\Sigma_{13} \cup \Sigma_3} = M^{\mathsf{cwa}}_{\Sigma_{12} \cup \Sigma_2} \circ M^{\mathsf{cwa}}_{\Sigma_{13} \cup \Sigma_3}$. First we Skolemize all tgd's in Σ_2 obtaining the set:

$$\Sigma^{\mathsf{sk}}_2 = \{R(x,y) \to T(x);\quad T(x) \to R(x,k(x))\}.$$

Next, similarly to the composition algorithm for standard SO-dependencies [16], we introduce two new Skolem functions, $f_R[.]$ and $g_R[.]$, for each relation name R in Σ^{sk}_2. The

[3] When writing $M \approx M'$, we mean that $Sol_M(I) \approx Sol_{M'}(I)$, for all instance I.

$$
\begin{aligned}
\sigma'_{13} \quad = \quad & \exists f_R\ \exists g_R\ \exists f_T\ \exists g_T\ \exists f\ \exists g\ \exists h\ : \\
& (\forall x\ S(x) \to f_R[f(x), x] = g_R[f(x), x]) \wedge \\
& (\forall x\ S(x), f_R[f(x), x] = g_R[f(x), x] \to f_R[f(x), h(f(x), x)] = g_R[f(x), h(f(x), x)]) \wedge \\
& (\forall x\ S(x), f_R[f(x), h(f(x), x)] = g_R[f(x), h(f(x), x)] \to f_R[f(x), h(f(x), h(f(x), x))] = g_R[f(x), h(f(x), h(f(x), x))]) \wedge \\
& \qquad\qquad \ldots\ldots \\
& (\forall x\ S(x), f_R[f(x), x] = g_R[f(x), x] \to V(f(x), x, g(f(x), x))) \wedge \\
& (\forall x\ S(x), f_R[f(x), h(f(x), x)] = g_R[f(x), h(f(x), x)] \to V(f(x), h(f(x), x), g(f(x), h(f(x), x)))) \wedge \\
& (\forall x\ S(x), f_R[f(x), h(f(x), h(f(x), x))] = g_R[f(x), h(f(x), h(f(x), x))] \to \\
& \qquad\qquad\qquad V(f(x), h(f(x), h(f(x), x)), g(f(x), h(f(x), h(f(x), x))))) \wedge \\
& \qquad\qquad \ldots\ldots \\
& \qquad\qquad \ldots\ldots
\end{aligned}
$$

Figure 1: Infinite st-SO dependency

intended meaning of these functions is that an equality of the form $f_R[a, b] = g_R[a, b]$ holds if and only if $R(a, b)$ holds.

From the set Σ_{12}, we know that $R(f(a), a)$ holds whenever $S(a)$ holds. Based on this, and the first tgd in Σ_2^{sk}, we add to Σ_{13} the following tgd:

$$
\begin{aligned}
S(x), \big(f_R[f(x), x] = g_R[f(x), x]\big) & \\
& \to \big(f_T[f(x)] = g_T[f(x)]\big).
\end{aligned}
$$

Similarly, from the second tgd in Σ_2^{sk} we construct the following tuple generating dependency:

$$
\begin{aligned}
S(x), \big(f_R[f(x), x] = g_R[f(x), x]\big) \to & \\
& \big(f_R[f(x), h(f(x))] = g_R[f(x), h(f(x))]\big).
\end{aligned}
$$

Note that in the standard SO mapping composition algorithm [3] this last tgd is not added. Intuitively, this is because in the standard algorithm the second tgd in Σ_2^{sk} is always satisfied, as $T(a)$ holds iff $R(a, b)$ holds for some value b. As the strong composition is based on the constructible solution semantics, it follows that the dependency will be applied as long as there are new values x for which the tgd was not applied before (even if it is satisfied). This reminiscent of the oblivious chase [9].
Finally, from the first dependency in Σ_2^{sk} and the new Skolem functions $f_T[.]$ and $g_T[.]$ we get:

$$
\begin{aligned}
S(x), \big(f_R[f(x), h(f(x))] = g_R[f(x), h(f(x))]\big) \to & \\
& \big(f_T[f(x)] = g_T[f(x)]\big).
\end{aligned}
$$

From this point on, no new dependency needs to be added as $T(f(x))$ was already "applied." We end up with σ_{13}, which we show below with all quantification explicitly included. With Σ_3 remaining unmodified, it can be verified that

$$
M^{\mathsf{cwa}}_{\sigma_{13} \cup \Sigma_3} = M^{\mathsf{cwa}}_{\sigma_{12} \cup \Sigma_2} \circ M^{\mathsf{cwa}}_{\Sigma_{23} \cup \Sigma_3}.
$$

$$
\begin{aligned}
\sigma_{13} = \ & \exists f_R\ \exists g_R\ \exists f_T\ \exists g_T\ \exists f\ \exists g\ \exists h\ : \\
& (\forall x\ S(x) \to f_R[f(x), x] = g_R[f(x), x]) \wedge \\
& (\forall x\ S(x), f_R[f(x), x] = g_R[f(x), x] \to \\
& \qquad\qquad\qquad f_T[f(x)] = g_T[f(x)]) \wedge \\
& (\forall x\ S(x), f_T[f(x)] = g_T[f(x)] \to \\
& \qquad\qquad f_R[f(x), h(f(x))] = g_R[f(x), h(f(x))]) \wedge \\
& (\forall x\ S(x), f_R[f(x), h(f(x))] = g_R[f(x), h(f(x))] \to \\
& \qquad\qquad\qquad f_T[f(x)] = g_T[f(x)]) \wedge \\
& (\forall x\ S(x), f_R[f(x), x] = g_R[f(x), x] \to \\
& \qquad\qquad\qquad h(f(x), x, g(f(x), x))) \wedge \\
& (\forall x\ S(x), f_R[f(x), h(f(x))] = g_R[f(x), h(f(x))] \to \\
& \qquad\qquad V(f(x), h(f(x)), g(f(x), h(f(x))))).
\end{aligned}
$$

Note that by loosening the rich acyclicity restriction on the target dependency set to weak acyclicity, the previous composition algorithm may not terminate. If, for example, we change the set Σ_2 to the following weakly, but not richly, acyclic set of dependencies

$$
\Sigma_2 = \{\forall x, y\ R(x, y) \to \exists z\ R(x, y)\},
$$

then our composition algorithm doesn't terminate and the result is the infinite SO-tgd σ'_{13} shown in Figure 1. On the other hand, in this case the standard SO composition algorithm [3] returns the following st-SO dependency.

$$
\begin{aligned}
& \sigma''_{13} = \ \exists f_R\ \exists g_R\ \exists f\ \exists g\ : \\
& (\forall x\ S(x) \to f_R[f(x), x] = g_R[f(x), x]) \wedge \\
& (\forall x\ S(x), f_R[f(x), x] = g_R[f(x), x] \to \\
& \qquad\qquad\qquad V(f(x), x, g(f(x), x)))
\end{aligned}
$$

Based on the above, we can now deliver the promised strong composable data exchange system.

Theorem 12. (GCOND, Rep^{cwa}, FO, ordSO) *is a strong composable data exchange system.*

5. SEMI-LAV COMPOSABILITY

As mentioned in the beginning of Section 4, there has recently been significant advances is finding classes of mappings that are closed under composition. Some of these classes are:

- tgdFULL. The class of mappings M_Σ, where Σ is a set of full source-to-target tgd's [16].
- tLAV. The class of mappings M_Σ, where Σ is a set of of source-to-target true-LAV tgd's [7].
- tgdSO. The class of mappings M_Σ, where Σ is a set of source-to-target SO-tgd's [16].
- standSO. The class of mappings M_Σ, where Σ is a st-SO dependency, a weakly acyclic set of target tgd's, and target egd's [3].

We have tgdFULL $\cap$ tLAV $\neq \emptyset$, and (tgdFULL $\cup$ tLAV) $\subset$ tgdSO $\subset$ standSO. It turned out [16] that the first order classes can be separated from the second order one's by the following decision problem.

Definition 15. *The schema mapping* ***membership*** *problem* Memb(Σ) *is: Given (I, J), is $(I, J) \in M_\Sigma$?*

It is well know that the membership problem for the mappings in the first two classes is in LOGSPACE. Fagin et al. showed in [16] that for mappings in tgdSO the membership problem is NP-complete. It follows that the same complexity result applies for the mappings in standSO.

Notably absent from the list of classes that are closed under composition is the class of mappings M_Σ, where Σ is a set of tgd's. As shown in [16], closing the class of tgd's under composition takes us to SO-tgd's. Since the SO-tgd's have a hard membership problem, it would be of interest to find ways of composing first order tgd's without ending up with with a second order result. In order to achieve this we restrict the notion of composability. First we need the concept of *semi-LAV* sets of tgd's. In [20] it was shown that the class of semi-LAV tgd's have nice tractable properties when it comes to data repair and correspondence checking.

The reader is asked to recall the notion of a *dependency graph* of a set of tgd's from Section 3. Let Σ be a set of tgd's over a schema $\mathbf{R}$. Let (R, i) be a vertex in the dependency graph of Σ. Then $rank(R, i)$ is the maximum number of existential edges along any path in the graph ending in (R, i). For all relational symbols $R \in \mathbf{R}$ that occur in Σ, we say that a position (R, i) is *unsafe* if $rank(R, i) > 0$. Any relational symbol R that contains an unsafe position is said to be unsafe. A set Σ of weakly acyclic dependencies is said to be *semi-LAV* if all unsafe relational symbols occur in the body of LAV-dependencies only.

Definition 16. *Let $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ be two composable mappings that may contain target dependencies, such that the following hold.*

1. *$\Sigma_{12} \cup \Sigma_{23}$ is a semi-LAV set.*
2. *All variables occurring in an unsafe position (relative to $\Sigma_{12} \cup \Sigma_{23}$) in the body of a dependency in Σ_{23} occur only once.*

Then $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ are ***semi-LAV composable****. The mapping $M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$ is called the* ***semi-LAV composition*** *of $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$.*

Consider for example mappings $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ as specified in Figure 2. Clearly $\Sigma_{12} \cup \Sigma_{23}$ is a semi-LAV set of tgd's with only one unsafe position $(Contracts, 3)$. As the variable *contract* in the last dependency occurs only in that unsafe position in the body, it makes $\Sigma_{12} \cup \Sigma_{23}$ also semi-LAV composable. Note that this properly extends the true-LAV tgd's of [7], by also allowing repeated variables in the body of the dependencies, as long as they don't occur in any unsafe position. We now have the following rather nice state of affairs.

Theorem 13. TGD*, the class of mappings specified by source-to-target tuple generating dependencies, is closed under semi-LAV composition.*

Next we describe the algorithm for semi-LAV composition. Due to space constraints the full proof of Theorem 13 is not included.

The input to the algorithm will be a pair $(\Sigma_{12}, \Sigma_{23})$ of tgd's, such that $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ are semi-LAV composable. Such a pair will be called *compatible.*

We shall use the fact that a tgd $\alpha \to \beta$ can be equivalently expressed by $\{\alpha \to \beta_1, \alpha \to \beta_2\}$, where β_1 contains the subset of atoms from β that are over safe predicates, and β_2 the atoms with existential variables. We can thus assume that Σ_{12} can be split as $\Sigma_{12}^{\forall} \cup \Sigma_{12}^{\exists}$, where the first set contains all the full tgd's, and the second set contains the rest.

Somewhat similarly we split Σ_{23} as $\Sigma_{23}^{\forall} \cup \Sigma_{23}^{\exists}$, where $\Sigma_{23}^{\forall}$ contains all the tgd's that has only safe predicates in the body, and $\Sigma_{23}^{\exists}$ contains the rest. Clearly $\Sigma_{23}^{\exists}$ will contain only LAV-tgd's.

In the algorithm, we freely switch between viewing a conjunction of atoms as a tableau, and vice versa. We are now ready to describe the algorithm:

```
Algorithm Semi-LAV Composition
Input: A compatible pair $(\Sigma_{12}, \Sigma_{23})$ of tgd's.
Output: $\Sigma_{13}$, such that $M_{\Sigma_{13}} = M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$.
  Let $\Sigma_{13}^{\forall}$, such that $M_{\Sigma_{13}^{\forall}} = M_{\Sigma_{12}^{\forall}} \circ M_{\Sigma_{23}^{\forall}}$.
  Let $\mathbf{S}$ = set of rel. symbols in heads of $\Sigma_{23}$
  Let $\Sigma_{13}^{\exists} = \emptyset$;
  For all $\alpha(\bar{x}) \to \exists \bar{y}\, \beta(\bar{x}, \bar{y}) \in \Sigma_{12}^{\exists}$ do
      For all endomorphisms $h$ on $\bar{x}$ do
          Let $\gamma(h(\bar{x}), \bar{z}) = Chase_{\Sigma_{12}^{\exists} \cup \Sigma_{23}^{\exists}}\big(\alpha(h(\bar{x}))\big)|_{\mathbf{S}}$
          Add tgd $\alpha(h(\bar{x})) \to \exists \bar{z}\, \gamma(h(\bar{x}), \bar{z})$ to $\Sigma_{13}^{\exists}$;
      End For
  End For
  Return $\Sigma_{13} = \Sigma_{13}^{\forall} \cup \Sigma_{13}^{\exists}$
```

The output Σ_{13} is a set of st-tgds. The algorithm proceeds by first computing the composition of $\Sigma_{12}^{\forall}$ with $\Sigma_{23}^{\forall}$. Since the former is a set of full tgd's, the algorithm of Fagin et al. in [16] can be used. For computing the composition of the two mappings specified by the remaining tgd's, we use the algorithm of Arocena et al. [7], with the difference that

$$\begin{aligned}\Sigma_{12} &= \big\{\mathit{Manage}(\mathit{emp},\mathit{mgr}) \to \exists \mathit{contract}\ \mathit{Employee}(\mathit{emp}), \mathit{Contracts}(\mathit{emp},\mathit{mgr},\mathit{contract})\big\}\\ \Sigma_{23} &= \big\{\mathit{Employee}(\mathit{emp}_1), \mathit{Employee}(\mathit{emp}_2) \to \mathit{Coworker}(\mathit{emp}_1,\mathit{emp}_2);\\ &\qquad \mathit{Contracts}(\mathit{emp},\mathit{emp},\mathit{contract}) \to \mathit{SelfMgrContract}(\mathit{emp},\mathit{contract})\big\}\end{aligned}$$

Figure 2: Semi-LAV dependencies

we run it for all partitionings of the sets of variables in the bodies of $\Sigma_{12}^{\exists}$, as expressed by the set of endomorphisms h on line 5. There is a bit of redundancy here (for the sake of conciseness) since there are fewer partitions on a finite set of variables than there are endomorphisms on it. On line 6, the sequence $\bar{z}$ represents the variables (null values) generated by the chase. We also use the notation $T|_{\mathbf{S}}$ for the subset of the atoms from T that are over relation symbols from $\mathbf{S}$.

5.1 Semi-LAV in homomorphic systems

If we want to consider homomorphic systems that are closed under composition under the stricter criteria of semi-LAV composability, we should naturally look at the notion of weak composition (UCQ-equivalent composition) from Definition 11. Using weak composition allows us to loosen the requirements for composability by omitting Clause 2 in Definition 16. We shall call a pair of mappings $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ fulfilling only Clause 1 in Definition 16 *relaxed semi-LAV composable.* It turns out that the class TGD of mappings is weakly closed under relaxed semi-LAV composition.

THEOREM 14. *The class* TGD *is weakly closed under relaxed semi-LAV composition.*

The proof, omitted here due to space restriction, is based on an algorithm similar to Algorithm Semi-LAV Composition. As a direct consequence of Theorem 14 we have

THEOREM 15. $(\mathsf{TBL}, \mathit{Rep}^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{TGD})$ *is a composable homomorphic data exchange system, with respect to relaxed semi-LAV composition.*

So far we have only discussed composition of mappings specified by source to target tgd's. The natural question then arises: can we achieve weak closure under semi-LAV composition for mappings that also contain target dependencies? In [14] it is shown that this is not possible, even when considering full tgd's. The following example is sufficient for the current context.

EXAMPLE 3. *Consider sets*

$$\begin{aligned}\Sigma_{12} &= \{R(x,y) \to S(x,y)\}\\ \Sigma_{2} &= \{S(x,y), S(y,z) \to S(x,z)\}\\ \Sigma_{23} &= \{S(x,x) \to T(x)\}.\end{aligned}$$

Clearly $\Sigma_{12} \cup \Sigma_2 \cup \Sigma_{23}$ *is a semi-LAV set. It can be easily shown that there is no finite set* Σ *of tgd's, such that* $M_\Sigma \approx M_{\Sigma_{12}\cup\Sigma_2} \circ M_{\Sigma_{23}}$.

On the other hand, Fagin et al. showed in [14] that for any set $\Sigma_{12} \cup \Sigma_2$, where Σ_{12} is a set of source-to-target tgd's and Σ_2 is a set of target dependencies with bounded core chase and bounded fact-block size, there exists a set Σ'_{12}, where Σ'_{12} is a finite set of source-to-target dependencies, such that

$$\mathit{Sol}_{M_{\Sigma_{12}\cup\Sigma_2}}(I) \approx \mathit{Sol}_{M_{\Sigma'_{12}}}(I),$$

for all instances I. The *bounded core chase* property means that there exists a constant c, depending only on the set of dependencies, such that the core chase [10] on any instance terminates in maximum c steps. *Bounded fact-block size* means that there exists a constant f depending only on the set of dependencies, such that number of tuples connected by common variables in the in the result of the core chase of any instance is bounded by f (for the exact definition of these notions see [14]).

The following lemma follows directly from the property of semi-LAV dependencies that the result of chasing any ground instance with a semi-LAV set of dependencies will result in an instance that has bounded fact-block size [20].

LEMMA 3. *Let* Σ *be a semi-LAV set of tgd's. Then* Σ *has the bounded core chase property and bounded fact-block size.*

Let cfTGD be the class of mappings specified by sets of source-to-target tgd's and target tgd's with bounded core chase property and bounded fact-block size. Then the previous lemma and Theorem 4.14 from [14] entail that cfTGD is weakly closed under semi-LAV composition. We thus have

THEOREM 16. $(\mathsf{TBL}, \mathit{Rep}^{\mathsf{owa}}, \mathsf{UCQ}, \mathsf{cfTGD})$ *is a composable homomorphic data exchange system, with respect to relaxed semi-LAV composition.*

5.2 Semi-LAV in strong systems

We now turn our attention to semi-LAV composition in strong systems. This means that we have to consider the notion of CWA strong composability from Definition 13. First let us see an example why the semi-LAV composability needs to be narrowed in order to obtain strong closure under composition.

EXAMPLE 4. *Consider the sets*

$$\begin{aligned}\Sigma_{12} &= \{R(x,y) \to \exists z\, S(x,y,z)\}\\ \Sigma_{23} &= \{S(x,x,z) \to T(x,z),\\ &\qquad S(x,y,z) \to M(x,y,z)\}\end{aligned}$$

Clearly $M_{\Sigma_{12}}$ *and* $M_{\Sigma_{23}}$ *are semi-LAV composable. The semi-LAV Composition algorithm will return*

$$\begin{aligned}\Sigma_{13} &= \{R(x,x) \to \exists z\, T(x,z), M(x,x,z),\\ &\qquad R(x,y) \to \exists z\, M(x,y,z)\}\end{aligned}$$

Now $M_{\Sigma_{13}} = M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$. Consider instance $I_1 = \{R(a,a)\}$ and instance $I_3 = \{T(a,c), M(a,a,c), M(a,a,d)\}$. We have $I_3 \in \Sigma_{13}(I_1)$, but all instances I_2 such that $I_2 \in \Sigma_{12}(I_1)$ have exactly one tuple, meaning that all instances in $\Sigma_{23}(I_2)$ have at most two tuples. Therefore $I_3 \notin \Sigma_{23}(I_2)$, for any $I_2 \in \Sigma_{12}(I_1)$. In other words, $M_{13}^{\mathtt{cwa}} \neq M_{12}^{\mathtt{cwa}} \circ M_{23}^{\mathtt{cwa}}$. It is also easy to see that no other finite set of source-to-target tgd's can cover the previous case.

Motivated by the previous example we narrow the notion of semi-LAV composability.

Definition 17. *Let $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ be semi-LAV composable mappings. They are said to be* ***restricted semi-LAV composable*** *if no unsafe predicate in the body of Σ_{23} contains a repeated variable.*

The restricted semi-LAV composition does not allow repeated variables occurring in atoms over unsafe predicates in the body. For example, the mappings M_{12}, M_{23} from Example 4 are not restricted semi-LAV composable because the unsafe predicate S appears in the body of a dependency containing the repeated variable x.

Theorem 17. *The class TGD is strongly closed under restricted semi-LAV composition.*

This gives us

Theorem 18. *$(\mathsf{COND}, Rep^{\mathtt{cwa}}, \mathsf{FO}, \mathsf{TGD})$ is a strong composable data exchange system, with respect to restricted semi-LAV composition.*

The algorithm for computing the strong composition of two restricted semi-LAV composable mappings differs from the *Semi-LAV Composition* algorithm in that it chases the tgd's that contains unsafe predicate (called unsafe tgd's) only with the identity endomorphism on the sequence of variables, and not all endomorphisms. This us due to the fact that the restricted semi-LAV composable sets do not contain any repeated variables in in body of unsafe tgd's. Using similar notation as in the previous algorithm we get the following algorithm that computes the strong composition of two restricted semi-LAV composable mappings.

```
Algorithm Strong Restricted Semi-LAV Composition
Input: A compatible pair $(\Sigma_{12}, \Sigma_{23})$ of tgd's.
Output: $\Sigma_{13}$, such that $M_{\Sigma_{13}} = M_{\Sigma_{12}} \circ M_{\Sigma_{23}}$.
1.  Let $\Sigma_{13}^{\forall}$, such that $M_{\Sigma_{13}^{\forall}} = M_{\Sigma_{12}^{\forall}} \circ M_{\Sigma_{23}^{\forall}}$.
2.  Let $\mathbf{S}$ = set of rel. symbols in heads of $\Sigma_{23}$
3.  Let $\Sigma_{13}^{\exists} = \varnothing$;
4.  For all $\alpha(\bar{x}) \to \exists\bar{y}\,\beta(\bar{x},\bar{y}) \in \Sigma_{12}^{\exists}$ do
5.          Let $\gamma(\bar{x},\bar{z}) = Chase_{\Sigma_{12}^{\exists} \cup \Sigma_{23}^{\exists}}\big(\alpha(\bar{x})\big)|_{\mathbf{S}}$
6.          Add tgd $\alpha(\bar{x}) \to \exists\bar{z}\,\gamma(\bar{x},\bar{z})$ to $\Sigma_{13}^{\exists}$;
7.  End For
8.  Return $\Sigma_{13} = \Sigma_{13}^{\forall} \cup \Sigma_{13}^{\exists}$
```

The following example gives the intuition behind the algorithm.

Example 5. *Consider sets*

$$\begin{aligned}
\Sigma_{12} &= \{R(x,x) \to \exists z\, S(x,z,z), T(x),\\
&\qquad R(x,y) \to T(x), T(y)\}\\
\Sigma_{23} &= \{S(x,y,z) \to M(x,z);\ T(x), T(y) \to M(x,y)\}.
\end{aligned}$$

Clearly $M_{\Sigma_{12}}$ and $M_{\Sigma_{23}}$ are restricted semi-LAV composable. The algorithm will return

$$\begin{aligned}
\Sigma_{13} = \{&R(x,x), R(y,z) \to M(x,y),\\
&R(x,x), R(y,z) \to M(x,z),\\
&R(x,y), R(z,z) \to M(x,z),\\
&R(x,y), R(z,z) \to M(y,z),\\
&R(x,y), R(z,v) \to M(x,z),\\
&R(x,y), R(z,v) \to M(x,v),\\
&R(x,y), R(z,v) \to M(y,z),\\
&R(x,y), R(z,v) \to M(y,v),\\
&\qquad R(x,x) \to \exists z\, M(x,z)\}
\end{aligned}$$

It can easily be verified that $M_{13}^{\mathtt{cwa}} = M_{12}^{\mathtt{cwa}} \circ M_{23}^{\mathtt{cwa}}$.

6. CONCLUSION

We have undertaken a systematic study of the application of representation systems to data exchange, revealing two major classes of representation systems for data exchange, namely homomorphic data exchange systems and strong data exchange systems. We have also integrated composition of mappings into the framework, and obtained a number of results concerning composable homomorphic and composable strong data exchange systems. We introduced the notion of semi-LAV composability, and showed that the class of first order tgd's is closed under this stricter notion. We showed how to integrate semi-LAV composability into homomorphic and strong composable data exchange systems. We believe that broadening the approach taken in this paper to include further operations on mappings, such as inversion [12] and recovery [6], will be a fruitful and challenging area of research. A step in this direction has already been taken by Karvounarakis and Tannen [25].

Acknowledgements

We are grateful to the anonymous referees for detailed and constructive comments.

7. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

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

[3] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. In *ICDT*, pages 129–142, 2010.

[4] M. Arenas, J. Pérez, and J. L. Reutter. Data exchange beyond complete data. In *PODS*, pages 83–94, 2011.

[5] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009.

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

[7] P. C. Arocena, A. Fuxman, and R. J. Miller. Composing local-as-view mappings: closure and applications. In *ICDT*, pages 209–218, 2010.

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

[9] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *KR*, pages 70–80, 2008.

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

[11] R. Fagin. Horn clauses and database dependencies. *J. ACM*, 29(4):952–985, 1982.

[12] R. Fagin. Inverting schema mappings. *ACM Trans. Database Syst.*, 32(4), 2007.

[13] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[14] R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a theory of schema-mapping optimization. In *PODS*, pages 33–42, 2008.

[15] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. In *PODS*, pages 90–101, 2003.

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

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

[18] G. Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer, 1991.

[19] G. Grahne and V. Kiricenko. Towards an algebraic theory of information integration. *Inf. Comput.*, 194(2):79–100, 2004.

[20] G. Grahne and A. Onet. Data correspondence, exchange and repair. In *ICDT*, pages 219–230, 2010.

[21] G. Grahne and A. Onet. Closed world chasing. In *LID*, pages 7–14, 2011.

[22] A. Hernich. Answering non-monotonic queries in relational data exchange. In *ICDT*, pages 143–154, 2010.

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

[24] T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[25] G. Karvounarakis and V. Tannen. Conjunctive queries and mappings with unequalities. In *Technical Reports (CIS)*, pages 1–15, 2008.

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

[27] L. Libkin. Incomplete information and certain answers in general data models. In *PODS*, pages 59–70, 2011.

[28] L. Libkin and C. Sirangelo. Open and closed world assumptions in data exchange. In *Description Logics*, 2009.

[29] A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. *ACM Trans. Database Syst.*, 32(1):4, 2007.

APPENDIX

For the ease of reading here are the definitions for the classes of dependencies reviewed in this paper. Consider $\mathbf{S}$ to be the source schema and $\mathbf{T}$ be the target schema.

- TGD
  Source-to-target tgd's.

- tgdFULL (Section 5)
  Full source-to-target tgd's.

- tLAV (Section 5)
  True-LAV source-to-target tgd's.

- STAND (Section 3)
  Source-to-target tgd's, weakly acyclic target tgd's and target egd's.

- cf TGD (Section 5.1)
  Source-to-target tgd's and target tgd's and there exists a positive integer c such that the core chase terminates in maximum c steps for any instance on the set of tgd's (*core chase property* [14]).

- SO-tgd's [16]
  $\exists f_1 \ldots \exists f_m((\forall \bar{x}_1(\alpha_1 \rightarrow \beta_1)) \wedge \ldots \wedge (\forall \bar{x}_n(\alpha_n \rightarrow \beta_n)))$
  where:
  - f_i is a function symbol, $i \in \{1, \ldots, m\}$.
  - α_i conjunction of: atoms $R(\bar{y})$, or equalities $t_1 = t_2$. Where, $R \in \mathbf{S}$, $\bar{y}$ is a vector of variables from $\bar{x}_i$, and t_1, t_2 are terms based on variables from $\bar{x}_i$ and functions $f_1, \ldots, f_m$, $1 \leq i \leq n$.
  - β_i conjunction of: atoms $S(\bar{t})$, where $S \in \mathbf{T}$ and $\bar{t}$ are terms based on $\bar{x}_i$ and $\{f_1, \ldots, f_m\}$.
  - each variable in $\bar{x}_i$ occurs in some relational atomic formula in α_i.

- st-SO tgd's [3]
  SO-tgd's where equalities over functional terms also can occur in the head of the tgd.

- plainSO-tgd's [5]
  SO-tgd's without equalities over functional terms.

- tgdSO (Section 3)
  SO-tgd's [16].

- normSO (Section 3)
  SO-tgd's, weakly acyclic target tgd's and target egd's.

- simSO (Section 3)
  plainSO-tgd's, weakly acyclic target tgd's and target egd's.

- standSO [3]
  st-SO tgd's, weakly acyclic target tgd's and target egd's.

- ordSO (Section 3)
  st-SO tgd's, richly acyclic target tgd's and target egd's.