

Towards a Theory of Schema-Mapping Optimization

Ronald Fagin

IBM Almaden

fagin@almaden.ibm.com

Phokion G. Kolaitis*

IBM Almaden

kolaitis@almaden.ibm.com

Alan Nash†

IBM Almaden

anash3@gmail.com

Lucian Popa‡

IBM Almaden

lucian@almaden.ibm.com

ABSTRACT

A schema mapping is a high-level specification that describes the relationship between two database schemas. As schema mappings constitute the essential building blocks of data exchange and data integration, an extensive investigation of the foundations of schema mappings has been carried out in recent years. Even though several different aspects of schema mappings have been explored in considerable depth, the study of schema-mapping optimization remains largely uncharted territory to date.

In this paper, we lay the foundation for the development of a theory of schema-mapping optimization. Since schema mappings are constructs that live at the logical level of information integration systems, the first step is to introduce concepts and to develop techniques for transforming schema mappings to “equivalent” ones that are more manageable from the standpoint of data exchange or of some other data interoperability task. In turn, this has to start by introducing and studying suitable notions of “equivalence” between schema mappings. To this effect, we introduce the concept of data-exchange equivalence and the concept of conjunctive-query equivalence. These two concepts of equivalence are natural relaxations of the classical notion of logical equivalence; the first captures indistinguishability for data-exchange purposes, while the second captures indistinguishability for conjunctive-query-answering purposes. Moreover, they coincide with logical equivalence on schema mappings specified by source-to-target tuple-generating dependencies (s-t tgds), but differ on richer classes of dependencies, such as second-order tuple-generating dependencies (SO tgds) and sets of s-t tgds and target tuple-generating dependencies (target tgds).

After exploring the basic properties of these three notions of equivalence between schema mappings, we focus on the following question: under what conditions is a schema mapping conjunctive-query equivalent to a schema mapping specified by a finite set of s-t tgds? We answer this question by obtaining complete characteriza-

tions for schema mappings that are specified by an SO tgd and for schema mappings that are specified by a finite set of s-t tgds and target tgds, and have terminating chase. These characterizations involve boundedness properties of the cores of universal solutions.

1. Introduction

A schema mapping is a high-level specification that describes the relationship between two database schemas. A schema mapping between two schemas is typically formalized as 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 database dependencies that specify the relationship between the source schema and the target schema. Schema mappings constitute the essential building blocks of such crucial data interoperability tasks as data integration and data exchange (see the surveys [17, 18]). For this reason, an extensive investigation of the foundations of schema mappings has been carried out in recent years. One main line in this investigation has focused on the properties of individual schema mappings and on their applications to data exchange and to query answering (see, for instance, [2, 5, 9, 10, 14, 15, 21, 24]). Another main line has focused on the study of basic operators on schema mappings, described in [3, 20], that produce new schema mappings from existing ones. Among these operators, the composition operator [11, 19, 22] and the inverse operator [8, 12]) are regarded as the most fundamental ones.

To motivate the work reported in this paper, let us for a moment reflect on the relational data model. Undoubtedly, one of the main reasons for its success is the early development of query optimization techniques that make it possible to evaluate a given query more efficiently. Query optimization is performed at both the physical level and the logical level of database systems. In particular, query optimization at the logical level amounts to transforming a given query to an equivalent one that can be executed more efficiently. Even though several different aspects of schema mappings have been explored in considerable depth, the study of schema-mapping optimization remains largely uncharted territory to date. Given that prototype industrial tools for managing schema mappings and performing data exchange have now been built [4, 16], it becomes quite imperative to make progress on the schema-mapping optimization front.

Our main aim in this paper is to lay the foundation for developing the theory of schema-mapping optimization. Since schema mappings are constructs that live at the logical level of information integration systems, the first step is to introduce concepts and to develop techniques for transforming schema mappings to “equivalent” ones that are more manageable from the standpoint of data exchange or of some other data interoperability task. This, however, raises the following questions: How do we compare schema map-

*On leave from UC Santa Cruz

†Current affiliation: Tradeworx, 54 Broad Street Suite 200, Red Bank, NJ 07701

‡Partially funded by U.S. Air Force Office for Scientific Research under contract FA9550-07-1-0223

plings? In particular, what are natural and useful notions of *equivalence* between schema mappings? Under what conditions can one schema mapping be replaced by another simpler, but “equivalent”, schema mapping?

Clearly, the classical notion of *logical equivalence* should be the starting point of any investigation of equivalence between schema mappings. It does not take long to realize, however, that logical equivalence may be too strong (and, consequently, too restrictive) a notion when applied to data exchange or query answering. We introduce and study two different and more relaxed notions of equivalence between schema mappings: *data-exchange equivalence* and *conjunctive-query equivalence*. Two schema mappings \mathcal{M} and \mathcal{M}' are *data-exchange equivalent* if, for every source instance I , the universal solutions for I under \mathcal{M} coincide with those for I under \mathcal{M}' . Intuitively, this means that the schema mappings \mathcal{M} and \mathcal{M}' are indistinguishable from each other for data-exchange purposes. Two schema mappings \mathcal{M} and \mathcal{M}' are *conjunctive-query equivalent* if, for every target conjunctive query Q and every source instance I , the certain answers of Q under \mathcal{M} coincide with those of Q under \mathcal{M}' . Thus, in this case, \mathcal{M} and \mathcal{M}' are indistinguishable from each other for conjunctive-query answering purposes. Logical equivalence always implies data-exchange equivalence; in turn, data-exchange equivalence always implies conjunctive-query equivalence. Moreover, we show that for schema mappings specified by source-to-target tuple-generating dependencies (s-t tgds), the three notions of logical equivalence, data-exchange equivalence, and conjunctive-query equivalence coincide. We show, however, that these notions are distinct for schema mappings specified by second-order tuple generating dependencies (SO tgds) and also for schema mappings specified by s-t tgds and target tuple-generating dependencies (target tgds).

After exploring the basic properties of these three notions of equivalence, we focus on the following question: under what conditions is a schema mapping conjunctive-query equivalent to a schema mapping specified by a finite set of s-t tgds? Schema mappings specified by s-t tgds are the syntactically simplest and most well-behaved schema mappings between relational schemas studied to date. Moreover, s-t tgds are the constraints of choice in the Clio schema mapping and data exchange tool [16]. Thus, the above question amounts to asking: when is a (more complex) schema mapping conjunctive-query equivalent to one of the simplest possible schema mappings? We investigate this question and obtain complete characterizations for three important classes of schema mappings between relational schemas: (i) schema mappings that are specified by a finite set of full s-t tgds and full target tgds; (ii) schema mappings that are specified by an SO tgd; and (iii) schema mappings that are specified by a finite set of s-t tgds and target tgds, and have terminating chase. The last class includes as members all schema mappings specified by a finite set of set of s-t tgds and a finite weakly acyclic set of target tgds (weakly acyclic sets are also known as “sets of constraints with stratified witness”) [7, 9].

Assume that \mathcal{M} is a schema mapping specified by a finite set of full s-t tgds and full target tgds. Using properties of Datalog, we show that \mathcal{M} is conjunctive-query equivalent to a schema mapping specified by a finite set of s-t tgds if and only if \mathcal{M} has a *bounded parallel chase*, which means that, for every source instance I , chasing I in parallel with the s-t tgds of \mathcal{M} terminates within a number of steps that depends only on \mathcal{M} (and not on I); moreover, in this case, \mathcal{M} is actually conjunctive-query equivalent to a finite set of full s-t tgds. Recall that full tgds are also known as *global-as-view* (GAV) constraints, and have been extensively studied in the context of data integration (see [18]).

The characterizations for the other two classes of schema map-

plings considered here are technically more involved; in addition, they require that we bring into the picture *boundedness* properties of the cores of universal solutions.

We show that a schema mapping \mathcal{M} specified by an SO tgd is conjunctive-query equivalent to a schema mapping specified by a finite set of s-t tgds if and only if \mathcal{M} has *bounded fact block size*. A *fact block* (or simply an *f-block*) in a target instance K is a connected component of the Gaifman graph of facts of K ; the nodes of that graph are the facts of K , and there is an edge between two facts if they have a null in common. A schema mapping \mathcal{M} has *bounded f-block size* if, for every source instance I , the number of facts in every f-block of the core of the universal solutions for I is bounded by a number that depends only on \mathcal{M} (and not in I).

Finally, let \mathcal{M} be a schema mapping that is specified by a finite set of s-t tgds and target tgds, and has terminating chase. We show that \mathcal{M} is conjunctive-query equivalent to a schema mapping specified by a finite set of s-t tgds if and only if \mathcal{M} has both *bounded core chase* and *bounded f-block size*. The *core chase*, introduced in [6, 23], is the following variant of the chase procedure: at every step, all dependencies are applied and then the core of the resulting instance is computed. A schema mapping \mathcal{M} has *bounded core chase* if, for every instance I , the core chase terminates within a number of steps that depends only on \mathcal{M} (and not on I).

Even though much more remains to be done, the conceptual and technical contributions contained in this paper open a new line of investigation in the study of schema mappings and pave the way for the development of static-analysis techniques for schema-mapping processing and optimization.

2. Preliminaries

A *schema* or *signature* \mathbf{R} is a finite sequence (R_1, \dots, R_k) of relation symbols, each of a fixed arity. An *instance* I over \mathbf{R} is a sequence (R_1^I, \dots, R_k^I) , where each R_i^I is a relation of the same arity as R_i . We shall often use R_i to denote both the relation symbol and the relation R_i^I that interprets it. An *atom* (over \mathbf{R}) is a formula $P(x_1, \dots, x_m)$, where P is a relation symbol in \mathbf{R} and x_1, \dots, x_m are variables, not necessarily distinct. A *fact* of an instance I (over \mathbf{R}) is an expression $P^I(v_1, \dots, v_m)$, where P is a relation symbol in \mathbf{R} and v_1, \dots, v_m are values such that $(v_1, \dots, v_m) \in P^I$. We assume that all instances I considered are finite, which means that every relation R_i^I is finite, for $1 \leq i \leq k$.

Schema mappings and data exchange notions. We review several notions from [9] that will be needed in this paper. 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 constraints. We say that \mathcal{M} is *specified by* Σ . When \mathbf{S} and \mathbf{T} are clear from context, we will sometimes write Σ in place of \mathcal{M} , and talk about constraints, instead of talking about a mapping.

We assume that we have a fixed infinite set \mathbf{Const} of constants and a fixed infinite set \mathbf{Null} of nulls that is disjoint from \mathbf{Const} . We write $\text{dom}(I)$ for the *domain* of an instance I , i.e., the set of all values occurring in I . All values occurring in a source instance I are assumed to be constants, i.e., $\text{dom}(I) \subseteq \mathbf{Const}$. In contrast, target instances have values in $\mathbf{Const} \cup \mathbf{Null}$.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. If I is a source instance, then a *solution for* I under \mathcal{M} is a target instance J such that $(I, J) \models \Sigma$. The set of all solutions for I under \mathcal{M} is denoted by $\text{Sol}^{\mathcal{M}}(I)$.

Let K, K' be two instances over the target schema \mathbf{T} . A function h from $\mathbf{Const} \cup \mathbf{Null}$ to $\mathbf{Const} \cup \mathbf{Null}$ is a *homomorphism* from K to K' if for every $c \in \mathbf{Const}$, we have that $h(c) = c$, and for every relation symbol R in \mathbf{T} and every tuple $(a_1, \dots, a_n) \in R^K$,

we have that $(h(a_1), \dots, h(a_n)) \in R^{K'}$. We write $K \rightarrow K'$ to denote that there is a homomorphism from K to K' . The instances K and K' are said to be *homomorphically equivalent* if $K \rightarrow K'$ and $K' \rightarrow K$. We write $K \leftrightarrow K'$ to denote that K and K' are homomorphically equivalent. Furthermore, we write $K \cong K'$ to denote that K and K' are isomorphic.

Given a schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a source instance I , a *universal solution for I under \mathcal{M}* is a solution J for I under \mathcal{M} such that for every solution J' for I under \mathcal{M} , we have that $J \rightarrow J'$. Intuitively, universal solutions are the “most general” solutions among all solutions for I . Clearly, if both J_1 and J_2 are universal solutions for I , then $J_1 \leftrightarrow J_2$.

Cores. Let K be a target instance. A subinstance K^* of K is called a *core* of K if $K \rightarrow K^*$, but there is no proper subinstance K' of K^* such that $K \rightarrow K'$. The following facts are well known:

- Every instance K has a core (this uses the finiteness of K).
- If K_1 and K_2 are cores of K , then $K_1 \cong K_2$; hence, we talk about *the* core of K , and we write $\text{core}(K)$ to denote it.
- If K^* is the core of K , then there is a *retraction* $h : K \rightarrow K^*$, i.e., a homomorphism from K to K^* that is the identity on K^* .
- If $K \leftrightarrow K'$, then $\text{core}(K) \cong \text{core}(K')$; in particular, if \mathcal{M} is a schema mapping and I is a source instance, then all universal solutions for I have isomorphic cores. Thus, we talk (up to isomorphism) about *the core of the universal solutions for I* .

Constraints. We consider constraints of several forms.

A *tuple-generating dependency (tgd)* is a constraint φ of the form

$$\forall \mathbf{x} \forall \mathbf{y} (\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \beta(\mathbf{x}, \mathbf{z})),$$

where α and β are conjunctions of atoms and every variable in \mathbf{x} occurs in both α and β . We will generally omit writing the $\forall \mathbf{x} \forall \mathbf{y}$ part. If \mathbf{z} is empty, we say that φ is a *full tgd*.

A *source-to-target tgd (s-t tgd)* is a tgd φ such that α consists of atoms from the source schema \mathbf{S} and β consists of atoms from the target schema \mathbf{T} . These dependencies are also known as *global-and-local as view (GLAV)* constraints (see [18]).

A *target tgd* is a tgd φ such that both α and β consist of atoms from the target schema \mathbf{T} .

A *second-order tgd (SO tgd)* is a constraint φ of the form:

$$\exists \mathbf{f} ((\forall \mathbf{x}_1 (\phi_1 \rightarrow \psi_1)) \wedge \dots \wedge (\forall \mathbf{x}_n (\phi_n \rightarrow \psi_n))), \text{ where}$$

- Each member of \mathbf{f} is a function symbol.
- Each ϕ_i is a conjunction of
 - (i) atomic formulas of the form $S(y_1, \dots, y_k)$, where S is a k -ary relation symbol of the source schema \mathbf{S} and y_1, \dots, y_k are variables in \mathbf{x}_i , not necessarily distinct, and
 - (ii) equalities of the form $t = t'$ where t and t' are terms built from \mathbf{x}_i and \mathbf{f} .
- Each ψ_i is a conjunction of atomic formulas $T(t_1, \dots, t_l)$, where T is an l -ary relation symbol of the target schema \mathbf{T} and t_1, \dots, t_l are terms built from \mathbf{x}_i and \mathbf{f} .
- Each variable in \mathbf{x}_i appears in some atomic formula of ϕ_i .

We will refer to each subformula $\forall \mathbf{x}_i (\phi_i \rightarrow \psi_i)$ as an SO tgd part of the SO tgd φ .

SO tgds were introduced in [11], where it was shown that they are the “right” language for specifying the composition of two finite sets of s-t tgds. It was also shown in [11] that every finite set of SO tgds is logically equivalent to a single SO tgd. For this reason, when it comes to SO tgds, we will consider (without loss of generality) schema mappings specified by a single SO tgd.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping specified by a finite set Σ of s-t tgds and target tgds. It is well known (see [10]) that if I is a source instance and J is a solution for I , then $\text{core}(J)$ is also

a solution for I (this uses the fact that there is a retraction from J to $\text{core}(J)$). In particular, if J is a universal solution for I , then $\text{core}(J)$ is also a universal solution for I . In contrast, this is not true for schema mappings specified by an SO tgd; as a matter of fact, in Example 3.10, we will exhibit an SO tgd such that the core of a universal solution is not a solution.

Chase. The *chase procedure* is an algorithm that was originally designed to reason about database dependencies (see [1]), but it turned out to have numerous applications to data exchange and other data interoperability tasks.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping such that Σ is one of the following: (i) a finite set of s-t tgds; (ii) a finite set of s-t tgds and target tgds; or, (iii) an SO tgd. Given a source instance I , the chase procedure chases I with the dependencies in Σ and produces a universal solution for I , provided the procedure terminates on I . We put $I_0^\Sigma = \emptyset$ and we write I_s^Σ to denote the target instance obtained in s steps of the chase, for $s \geq 0$. If the chase terminates on I , then we write I^Σ for the result of the chase (clearly, $I^\Sigma = I_t^\Sigma$, where t is the last step of the chase); otherwise, the result of the chase is undefined.

We explain what *chasing I with the dependencies in Σ* means by considering a target tgd φ of the form $\forall \mathbf{x} (\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \beta(\mathbf{x}, \mathbf{z}))$. Let \mathbf{a} and \mathbf{b} be two tuples of elements in I_s^Σ . We say that φ *applies on I_s^Σ and (\mathbf{a}, \mathbf{b})* if $I_s^\Sigma \models \alpha(\mathbf{a}, \mathbf{b})$, but there is no tuple \mathbf{c} in I_s^Σ such that $I_s^\Sigma \models \beta(\mathbf{a}, \mathbf{c})$. If φ applies on I_s^Σ and (\mathbf{a}, \mathbf{b}) , then we *fire φ on I_s^Σ and (\mathbf{a}, \mathbf{b})* by adding to I_{s+1}^Σ facts with new nulls \mathbf{u} interpreting the variables \mathbf{z} so that $I_{s+1}^\Sigma \models \beta(\mathbf{a}, \mathbf{u})$. Chasing I with s-t tgds is defined in an analogous way, except that now φ *applies on I_s^Σ and (\mathbf{a}, \mathbf{b})* means that $I \models \alpha(\mathbf{a}, \mathbf{b})$, but there is no tuple \mathbf{c} in I_s^Σ such that $I_s^\Sigma \models \beta(\mathbf{a}, \mathbf{c})$. Chasing I with an SO tgd is defined in a similar way (see [11] for details).

In this paper, we will make use of several variants of the chase procedure that we describe next.

- The *standard chase*, where at each step we use an arbitrary order among the constraints to fire one constraint among the ones that apply. This chase was used in the study of data exchange for schema mappings specified by s-t tgds and target tgds [9].
- The *parallel chase*, where at each step we fire all constraints that apply (and this parallel firing counts as just one step of the parallel chase).
- The *core chase*, which was introduced in [6, 23]. Each step in the core chase consists of two sub-steps: (a) do one step of the parallel chase; and, (b) compute the core of the result.

Fix, for a moment, one of the above variants of the chase. We say that a set Σ of constraints has *terminating chase* if, for every source instance I , the chase of I with Σ terminates in a finite number of steps (i.e., we reach a stage at which no dependency applies). As mentioned earlier, in this case the result I^Σ of the chase is a universal solution for I . If Σ is a finite set of s-t tgds or an SO tgd, then the three variants of the chase procedure described above have terminating chase. The same holds true if Σ is a finite set of s-t tgds and full target tgds. If, however, Σ is an arbitrary finite set of s-t tgds and target tgds, then none of these chase variants may have terminating chase. It was shown in [6, 23] that the core chase terminates whenever any other variant of the chase terminates.

Since the chase does not always terminate, it is natural to ask for broad, sufficient conditions for its termination. Such a useful and extensively studied condition is that Σ is the union of a finite set of s-t tgds with a finite *weakly acyclic* set of target tgds (the latter is also known as a set of constraints with *stratified witness*) [7, 9]. Sets of full target tgds and acyclic sets of inclusion dependencies are special cases of weakly acyclic sets.

Certain answers. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping and Q a query over the target schema \mathbf{T} . If I is a source instance, then the *certain answers* $\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I)$ of Q on I under \mathcal{M} are defined as

$$\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I) = \bigcap_{J \in \text{Sol}^{\mathcal{M}}(I)} Q(J),$$

whenever $\text{Sol}^{\mathcal{M}}(I) \neq \emptyset$, and undefined otherwise. Note that if Q is a Boolean query, then the intersection corresponds to the conjunction of Boolean values; hence, in this case, $\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I) = \text{true}$ if and only if, for every solution J for I , we have that $Q(J) = \text{true}$.

On the face of it, the definition of the certain answers is not effective, as it involves computing an intersection over a potentially infinite set. Nonetheless, the certain answers of conjunctive queries can be obtained by evaluating the query on a universal solution whenever a universal solution exists (see [9]). More precisely, the following hold:

- If Q is a Boolean conjunctive query and J is a universal solution for I , then $\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I) = Q(J)$.
- If Q is a k -ary conjunctive query, for $k \geq 1$, and J is a universal solution for I , then $\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I) = Q(J) \cap \text{dom}(I)^k$.

3. Equivalences of Schema Mappings

A theory of schema-mapping optimization must be based on suitable concepts of *equivalence* between schema mappings. In this section, we introduce two such notions that are relaxations of the classical notion of logical equivalence, study their basic properties, and compare them to logical equivalence. These notions make it possible to explore schema mappings that are not necessarily logically equivalent to a given schema mapping, but are “good enough” for specific purposes, such as data exchange or query answering. We begin by recalling the notion of logical equivalence.

DEFINITION 3.1. Two schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ are *logically equivalent*, denoted by $\mathcal{M} \equiv \mathcal{M}'$, if for every source instance I and target instance J , we have that

$$(I, J) \models \Sigma \text{ if and only if } (I, J) \models \Sigma'.$$

In other words, $\mathcal{M} \equiv \mathcal{M}'$ if, for every source instance I , we have that $\text{Sol}^{\mathcal{M}}(I) = \text{Sol}^{\mathcal{M}'}(I)$.

The first relaxation of the notion of logical equivalence does not distinguish between two schema mappings with the same data exchange behavior.

DEFINITION 3.2. Two schema mappings \mathcal{M} and \mathcal{M}' are *data-exchange equivalent*, denoted by $\mathcal{M} \stackrel{\text{DE}}{\equiv} \mathcal{M}'$ if, for every source instance I , the set of universal solutions for I under \mathcal{M} coincides with the set of universal solutions for I under \mathcal{M}' .

The second relaxation of the notion of logical equivalence does not distinguish between two schema mappings with the same query-answering behavior.

DEFINITION 3.3. Let \mathcal{L} be a collection of queries. Two schema mappings \mathcal{M} and \mathcal{M}' are *\mathcal{L} -query equivalent* (or, *\mathcal{L} -equivalent*), denoted by $\mathcal{M} \equiv_{\mathcal{L}} \mathcal{M}'$, if, for every target query $Q \in \mathcal{L}$ and every source instance I , we have that

- $\text{Sol}^{\mathcal{M}}(I) = \emptyset$ if and only if $\text{Sol}^{\mathcal{M}'}(I) = \emptyset$, and
- $\text{cert}_{\mathcal{Q}}^{\mathcal{M}}(I) = \text{cert}_{\mathcal{Q}}^{\mathcal{M}'}(I)$, if $\text{Sol}^{\mathcal{M}}(I) \neq \emptyset$ and $\text{Sol}^{\mathcal{M}'}(I) \neq \emptyset$.

As an important special case, which we will explore in depth, \mathcal{M} and \mathcal{M}' are *conjunctive-query equivalent* (or *CQ-equivalent*) if $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$, where CQ is the collection of all conjunctive queries.

Clearly, query equivalence is a notion that is dependent on a particular class of queries. Next, we show that CQ-equivalence can be characterized using data-exchange concepts.

DEFINITION 3.4. If $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping, then $F_{\text{core}}^{\mathcal{M}}$ is the following partial function from source instances to target instances:

$$F_{\text{core}}^{\mathcal{M}}(I) = \begin{cases} \text{core}(J) & \text{if there is a universal solution } J \text{ for } I; \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Note that, whenever $F_{\text{core}}^{\mathcal{M}}$ is defined, it is well-defined, since the core of universal solutions is unique (up to isomorphism).

PROPOSITION 3.5. *Let \mathcal{M} and \mathcal{M}' be schema mappings such that both have the following property: for every instance I , if there is a solution for I , then there is a universal solution for I . Then the following statements are equivalent:*

1. \mathcal{M} and \mathcal{M}' are CQ-equivalent.
2. For every source instance I , we have that:
 - (a) there is a universal solution for I under \mathcal{M} if and only if there is a universal solution for I under \mathcal{M}' ; and
 - (b) if J is a universal solution for I under \mathcal{M} , and J' is a universal solution for I under \mathcal{M}' , then $J \leftrightarrow J'$ (or, equivalently, $\text{core}(J) \cong \text{core}(J')$).
3. For every source instance I , we have that $F_{\text{core}}^{\mathcal{M}}(I) = F_{\text{core}}^{\mathcal{M}'}(I)$, which means that either both $F_{\text{core}}^{\mathcal{M}}(I)$ and $F_{\text{core}}^{\mathcal{M}'}(I)$ are undefined or both are defined and equal to each other (up to isomorphism).

From now on, when studying CQ-equivalence, we will make repeated use of the characterizations of it given by Proposition 3.5. The above proposition also suggests that CQ-equivalence can be considered a weaker notion of data-exchange equivalence, since data exchange with two CQ-equivalent schema mappings always gives homomorphically equivalent universal solutions. In fact, as stated in Part 3 of the above proposition, two CQ-equivalent schema mappings give rise to the same partial function F_{core} .

It should be pointed out that the hypothesis in Proposition 3.5 that a universal solution exists whenever a solution exists is satisfied by several important and extensively studied classes of schema mappings. Specifically, it is satisfied by every schema mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ such that (a) Σ is a set of s-t tgds, or (b) Σ is an SO tgd, or (c) Σ is the union of a set of s-t tgds with a set of target tgds and egds, and has terminating chase. In particular, this hypothesis is satisfied by every schema mapping specified by $\Sigma_{st} \cup \Sigma_t$, where Σ_{st} is a set of s-t tgds and Σ_t is the union of a weakly acyclic set of target tgds with a set of target egds [9].

3.1 A hierarchy of schema-mapping equivalences

It is easy to see that data-exchange equivalence and CQ-equivalence are progressive relaxations of logical equivalence.

PROPOSITION 3.6. *Let \mathcal{M} and \mathcal{M}' be two schema mappings.*

1. If $\mathcal{M} \equiv \mathcal{M}'$, then $\mathcal{M} \stackrel{\text{DE}}{\equiv} \mathcal{M}'$.
2. If $\mathcal{M} \stackrel{\text{DE}}{\equiv} \mathcal{M}'$, then $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.

We now give examples showing the converses to the two parts of Proposition 3.6 are not true: hence the three notions of equivalence are distinct. The first two examples show the distinction between data-exchange equivalence and logical equivalence.

EXAMPLE 3.7. Let \mathbf{T} be a target schema consisting of a binary relation symbol T , and let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ be two schema mappings such that $\Sigma = \emptyset$ and Σ' consists of the full target tgd $T(x, y) \rightarrow T(y, x)$. Then \mathcal{M} and \mathcal{M}' are data exchange equivalent, but not logically equivalent. They are data exchange equivalent, since, for every source instance I , they have the same set of universal solutions for I (namely, the singleton set containing the empty instance). They are not logically equivalent; for example, the target instance $J = \{T(1, 2)\}$ is a solution for every I under \mathcal{M} , but is not a solution for any I under \mathcal{M}' .

EXAMPLE 3.8. Consider the schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$, where Σ is the SO tgd

$$\exists f \exists g \forall x (S(x) \rightarrow T(x, f(x), g(f(x))))$$

and Σ' is the SO tgd

$$\exists f \exists g \forall x (S(x) \rightarrow T(x, f(x), g(x))).$$

We claim that \mathcal{M} and \mathcal{M}' are data-exchange equivalent, but not logically equivalent.

Let I be a source instance, and let $U = \{T(s, \alpha_s, \beta_s) : s \in S^I\}$, where α_s and β_s are distinct nulls for every $s \in S^I$. Then it can be seen that U is a universal solution for I under \mathcal{M} ; moreover, it is its own core. Thus, the universal solutions for I under \mathcal{M} are precisely those instances that contain U (up to a renaming of nulls) and have a homomorphism into U . The same holds for \mathcal{M}' . Consequently, \mathcal{M} and \mathcal{M}' have the same universal solutions and so they are data exchange equivalent.

We now show that \mathcal{M} and \mathcal{M}' are not logically equivalent. Let I be the source instance $\{S(1), S(2)\}$ and let J be the target instance $\{T(1, 3, 4), T(2, 3, 5)\}$. Then $(I, J) \models \Sigma'$, but $(I, J) \not\models \Sigma$; indeed, if $(I, J) \models \Sigma$, then we would have $f(1) = f(2) = 3$ and, therefore, $g(f(1)) = g(f(2))$; this is a contradiction, since it must also be the case that $g(f(1)) = 4$ and $g(f(2)) = 5$.

The next two examples show the distinction between data-exchange equivalence and CQ-equivalence.

EXAMPLE 3.9. Consider the schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$, where:

- Σ consists of the full s-t tgd $S(x) \rightarrow T(x, x)$ and the full target tgd $T(x, u) \wedge T(u, x) \rightarrow T(x, x)$;
- Σ' consists of the full s-t tgd $S(x) \rightarrow T(x, x)$ and the full target tgd $T(x, u) \wedge T(u, v) \wedge T(v, x) \rightarrow T(x, x)$.

Then \mathcal{M} and \mathcal{M}' are CQ-equivalent because they have the same chase result (and hence the same core) for every source instance I . However, \mathcal{M} and \mathcal{M}' are not data exchange equivalent; for example, the instance $U = \{T(1, 1), T(\alpha, \beta), T(\beta, \gamma), T(\gamma, \alpha)\}$, where α, β, γ are distinct nulls is a universal solution for the instance $I = \{S(1)\}$ under \mathcal{M} , but not a solution for I under \mathcal{M}' .

EXAMPLE 3.10. Consider the schema mappings $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$, where Σ is the full s-t tgd $S(x, y) \rightarrow T(x, y)$, and where Σ' is the SO tgd with parts

$$\begin{aligned} S(x, y) &\rightarrow T(x, y) \\ S(x, x) &\rightarrow T(x, f(x)) \\ S(x, x) \wedge (x = f(x)) &\rightarrow W(x) \end{aligned}$$

We now show that \mathcal{M} and \mathcal{M}' are CQ-equivalent, but not data-exchange equivalent. They are not data-exchange equivalent, because the instance $U = \{T(1, 1)\}$ is a universal solution for the instance $I = \{S(1, 1)\}$ under \mathcal{M} , but U is not a solution for I under \mathcal{M}' , since we must have $f(1) = 1$ and then the implication

in the third SO tgd part is not satisfied. Now, \mathcal{M} and \mathcal{M}' are CQ-equivalent, because for every source instance I , the target instances

$$\begin{aligned} U &= \{T(c, d) : (c, d) \in S^I\} \\ U' &= U \cup \{T(c, u_c) : (c, c) \in S^I\} \end{aligned}$$

(where u_c is a distinct null for every c) are universal solutions for I under \mathcal{M} and \mathcal{M}' respectively, and $U \leftrightarrow U'$.

There is another important piece of information that we can extract from this example. Specifically, note that \mathcal{M}' is a schema mapping specified by an SO tgd such that the core of a universal solution is not a solution. Indeed, consider again the source instance $I = \{S(1, 1)\}$. The target instance $\{T(1, 1), T(1, u)\}$, where u is a null, is a universal solution for I under \mathcal{M}' . Moreover, its core is $\{T(1, 1)\}$, which, as seen above, is not a solution for I under \mathcal{M}' .

In particular, it follows from this argument that \mathcal{M}' is not logically equivalent to *any* schema mapping specified by a finite set of s-t tgds, since such schema mappings have the property that the core of a universal solution is a solution. On the other hand, \mathcal{M}' is CQ-equivalent to a schema mapping specified by a single full s-t tgd (\mathcal{M} in this case). Thus, this example shows the potential benefit of using CQ-equivalence: we can replace a schema mapping (\mathcal{M}') by a schema mapping (\mathcal{M}) that is in a simpler language but still CQ-equivalent, whereas such simplification is not possible under logical equivalence.

The key reason that CQ-equivalence fails to imply data-exchange equivalence is the following: if U is a universal solution for I under \mathcal{M} , and $U \leftrightarrow U'$, then U' is also *universal for I under \mathcal{M}* (i.e., for every solution J for I , we have that $U' \rightarrow J$), but U' may not necessarily be a solution for I . This behavior is exhibited in both Example 3.9 and Example 3.10: the schema mappings \mathcal{M} and \mathcal{M}' have homomorphically equivalent universal solutions (which implies CQ-equivalence), but there is one universal solution under one schema mapping that is not a solution under the other schema mapping. This motivates the following definition.

DEFINITION 3.11. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that \mathcal{M} *has all the universal solutions* if whenever U is a universal solution for I under \mathcal{M} , and U' is a target instance such that $U \leftrightarrow U'$, then U' is also a solution for I under \mathcal{M} (hence, a universal solution).

PROPOSITION 3.12. *If \mathcal{M} and \mathcal{M}' have all the universal solutions, then $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$ implies $\mathcal{M} \stackrel{\text{DE}}{=} \mathcal{M}'$.*

It is clear that the following property is a sufficient condition for a schema mapping to have all the universal solutions.

DEFINITION 3.13. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that \mathcal{M} is *preserved under target homomorphisms* if, for every source instance I and all target instances J and J' such that $(I, J) \models \Sigma$ and $J \rightarrow J'$, we have that $(I, J') \models \Sigma$.

It is easy to see that schema mappings specified by s-t tgds are preserved under target homomorphisms. This condition, however, may not hold once we add target tgds, or if the schema mapping is specified by an SO tgd. The following result follows fairly easily from Proposition 3.5.

PROPOSITION 3.14. *Let \mathcal{M} and \mathcal{M}' be schema mappings such that both are preserved under target homomorphisms and both have the following property: for every instance I , if there is a solution for I , then there is a universal solution for I . Then, $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$ implies that $\mathcal{M} \equiv \mathcal{M}'$. Thus, in this case, all three notions of equivalence coincide.*

PROOF. Assume first that I has a solution under \mathcal{M} . Then by assumption, I has a universal solution U under \mathcal{M} . By Proposition 3.5, it follows that I has a universal solution U' under \mathcal{M}' , and $U \leftrightarrow U'$. By preservation under target homomorphisms, the solutions of I under \mathcal{M} are exactly the homomorphic images of U . Similarly, the solutions of I under \mathcal{M}' are exactly the homomorphic images of U' . Since $U \leftrightarrow U'$, it follows that I has the same solutions under \mathcal{M} and \mathcal{M}' . We have shown that if I has a solution under \mathcal{M} , then I has the same solutions under \mathcal{M} and \mathcal{M}' . By symmetry, the same holds when we reverse the role of \mathcal{M} and \mathcal{M}' . It follows easily that I has the same solutions under \mathcal{M} and \mathcal{M}' , so $\mathcal{M} \equiv \mathcal{M}'$. \square

From now on, our main focus will be on CQ-equivalence and on schema mappings that are not preserved under homomorphisms, in which case CQ-equivalence is a genuine relaxation of logical equivalence.

3.2 Other properties of CQ-equivalence

Our next result reveals an algorithmic difference between logical equivalence and CQ-equivalence.

THEOREM 3.15. *The following statements hold.*

- *The following problem is decidable: given two schema mappings specified by finite sets of s-t tgds and finite weakly acyclic set of target tgds, are they logically equivalent?*
- *The following problem is undecidable: given two schema mappings specified by finite sets of s-t tgds and finite weakly acyclic sets of target tgds, are they CQ-equivalent? In fact, this problem is undecidable even for schema mappings specified by copy s-t tgds and full target tgds, where a copy s-t tgd is a full s-t tgd of the form $P(\mathbf{x}) \rightarrow R(\mathbf{x})$.*

PROOF. (Sketch) The decidability result follows from well-known properties of the chase. Specifically, let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ be two schema mappings, where Σ and Σ' are each a finite set of s-t tgds and a weakly acyclic set of target tgds. To check that $\Sigma \models \phi$ for some $\phi \in \Sigma'$, it is enough to check whether the conclusion of ϕ appears in the result of chasing the premise of ϕ with Σ . (We are making use here of the fact that the chase is terminating.) This allows us to check whether $\Sigma \models \Sigma'$; the converse can be tested in a similar way.

The undecidability result is obtained via a reduction from the problem of Datalog equivalence: given two Datalog programs over the same schema, do they compute the same recursive query on every input database? This problem was shown to be undecidable in [26]. Given two Datalog programs over the same schema \mathbf{T} , we construct two schema mappings consisting of copy s-t tgds and full target tgds as follows. We first create a schema \mathbf{S} that is a replica of \mathbf{T} ; we then add copy s-t tgds from \mathbf{S} to \mathbf{T} for each relation symbol in \mathbf{S} ; finally, we view each given Datalog program as a finite set of full target tgds. The two given Datalog programs are equivalent if and only if the two schema mappings are CQ-equivalent. This is so because the result of each Datalog program on each input database I coincides with the core of the universal solutions for I under the corresponding schema mapping. \square

The last result in this section asserts that CQ-equivalence is preserved under composition of schema mappings specified by s-t tgds. This result shows a similarity between CQ-equivalence and logical equivalence (the latter is always preserved under composition, by definition of composition).

We recall the concept of the *composition* of two schema mappings, introduced in [11, 20]. Let $\mathcal{M}_{12} = (\mathbf{S}_1, \mathbf{S}_2, \Sigma_{12})$ and

$\mathcal{M}_{23} = (\mathbf{S}_2, \mathbf{S}_3, \Sigma_{23})$ be schema mappings. The *composition* $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is a schema mapping $(\mathbf{S}_1, \mathbf{S}_3, \Sigma_{13})$ such that for every instance I over \mathbf{S}_1 and every instance K over \mathbf{S}_3 , we have that $(I, K) \models \Sigma_{13}$ if and only if there is an instance J over \mathbf{S}_2 such that $(I, J) \models \Sigma_{12}$ and $(J, K) \models \Sigma_{23}$.

PROPOSITION 3.16. *Let \mathcal{M}_{12} and \mathcal{M}'_{12} be schema mappings from \mathbf{S}_1 to \mathbf{S}_2 that are specified by s-t tgds. Let \mathcal{M}_{23} and \mathcal{M}'_{23} be schema mappings from \mathbf{S}_2 to \mathbf{S}_3 that are also specified by s-t tgds. If \mathcal{M}_{12} is CQ-equivalent to \mathcal{M}'_{12} , and \mathcal{M}_{23} is CQ-equivalent to \mathcal{M}'_{23} , then the composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ is CQ-equivalent to the composition $\mathcal{M}'_{12} \circ \mathcal{M}'_{23}$.*

PROOF. We shall use Proposition 3.5 to show the CQ-equivalence of $\mathcal{M}_{12} \circ \mathcal{M}_{23}$ and $\mathcal{M}'_{12} \circ \mathcal{M}'_{23}$. First, we show that condition 2(a) in Proposition 3.5 is true. In particular, we shall show that for every source instance I , there is always a universal solution for I under $\mathcal{M}_{12} \circ \mathcal{M}_{23}$, and, similarly, there is always a universal solution for I under $\mathcal{M}'_{12} \circ \mathcal{M}'_{23}$. Indeed, let I be an arbitrary instance over \mathbf{S}_1 . Let U_1 be the result of chasing I with Σ_{12} and let V_1 be the result of chasing U_1 with Σ_{23} . By Proposition 7.2 in the full version of [8], we have that V_1 is a universal solution for $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. A similar V_2 can be constructed for $\mathcal{M}'_{12} \circ \mathcal{M}'_{23}$ by chasing I with Σ'_{12} to obtain U_2 and then by chasing U_2 with Σ'_{23} to obtain V_2 .

To show condition 2(b), it is sufficient to show that the previously constructed V_1 and V_2 are homomorphically equivalent. First, we know that U_1 and U_2 are homomorphically equivalent, by Proposition 3.5, since \mathcal{M}_{12} and \mathcal{M}'_{12} are CQ-equivalent. Let V'_2 be the result of chasing U_1 with Σ'_{23} . It follows that V_1 and V'_2 are homomorphically equivalent, by Proposition 3.5, since \mathcal{M}_{23} and \mathcal{M}'_{23} are CQ-equivalent. It remains to show that V'_2 and V_2 are homomorphically equivalent.

We know that U_1 has a homomorphism into U_2 (and vice-versa). It follows that U_1 has a homomorphism into (U_2, V_2) . By applying Lemma 3.4 in [9], since (U_2, V_2) satisfies Σ'_{23} , we can infer the existence of a homomorphism from the result of chasing U_1 with Σ'_{23} , which is (U_1, V'_2) , into (U_2, V_2) . In particular, we obtain a homomorphism from V'_2 to V_2 . A symmetrical argument exploits the fact that U_2 has a homomorphism into U_1 to obtain that V_2 has a homomorphism into V'_2 . \square

The preceding proposition has immediate consequences for the optimization of *flows* (or, sequences) of schema mappings that are specified by s-t tgds. Indeed, we can replace any schema mapping \mathcal{M} in the sequence by another schema mapping \mathcal{M}' that is specified by s-t tgds and that is CQ-equivalent to \mathcal{M} , and obtain a new sequence of mappings whose composition is CQ-equivalent to the composition of the original sequence.

It should be pointed out that Madhavan and Halevy [19] used a different notion of composition of schema mappings. Their notion asserts that a schema mapping \mathcal{M}_{13} is a *composition (relative to the class of conjunctive queries)* of two schema mappings \mathcal{M}_{12} and \mathcal{M}_{23} if \mathcal{M}_{13} has the same conjunctive-query answering behavior as the successive application of \mathcal{M}_{12} and \mathcal{M}_{23} . Thus, to be a composition in their terms, it is enough for a schema mapping \mathcal{M}_{13} to be CQ-equivalent to the “true” composition $\mathcal{M}_{12} \circ \mathcal{M}_{23}$. So, Madhavan and Halevy’s notion of composition was implicitly based on CQ-equivalence, even though they did not single out the concept of CQ-equivalence explicitly.

4. CQ-Equivalence to Source-to-Target Tgds

As mentioned earlier, for schema mappings that are not necessarily preserved under homomorphisms (e.g., schema mappings specified by an SO tgd or schema mappings specified by s-t tgds and

target tgds), the notion of CQ-equivalence is a true relaxation of the notion of logical equivalence. This distinction can be beneficial, as it may be possible to replace a given schema mapping \mathcal{M} with a “simpler” schema mapping \mathcal{M}' that is CQ-equivalent to \mathcal{M} , even though \mathcal{M} is not logically equivalent to any such “simpler” schema mapping.

In this section, we study the question: when is a schema mapping CQ-equivalent to a schema mapping specified by s-t tgds? We obtain complete characterizations for three important classes of schema mappings specified by constraints that are more complex than s-t tgds: (i) schema mappings specified by full s-t tgds and full target tgds; (ii) schema mappings specified by an SO tgd; and (iii) schema mappings specified by s-t tgds and target tgds, and having terminating chase. Although the characterization for the first class can be derived from that for the third class, we include it explicitly here for several reasons: (a) it has a very simple statement of the characterization (the CQ-equivalence holds if and only if there is a bounded parallel chase); (b) it can be proved using known concepts and results about Datalog, and (c) it paves the way for proving the other two characterizations that are technically more involved and require the introduction of new concepts and techniques.

4.1 Full s-t tgds and full target tgds

In this section, we characterize when a schema mapping specified by full s-t tgds and full target tgds is CQ-equivalent to a schema mapping specified by s-t tgds.

DEFINITION 4.1. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that \mathcal{M} has *bounded parallel chase* if there is a positive integer b such that, for every source instance I , the parallel chase of I with Σ terminates in at most b steps.

It is clear that if \mathcal{M} is specified by a finite set of s-t tgds, then \mathcal{M} has bounded parallel chase. The next theorem tells us that bounded parallel chase is just what is needed for a schema mapping specified by a finite set of full s-t tgds and full target tgds to be CQ-equivalent to a schema mapping specified by s-t tgds.

THEOREM 4.2. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping specified by a finite set of full s-t tgds and full target tgds. Then the following statements are equivalent:

1. \mathcal{M} has bounded parallel chase.
2. There exists a schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ specified by a finite set of full s-t tgds such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.
3. There exists a schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ specified by a finite set of s-t tgds such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.
4. There exists a schema mapping $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ specified by an SO tgd such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.

PROOF. (*Sketch*) We will use the following connection between full tgds and Datalog. Suppose that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is a schema mapping specified by a finite set of full s-t tgds and full target tgds. Then the following facts can be established easily:

- There is a Datalog program $\pi_{\mathcal{M}}$ over the source schema \mathbf{S} such that, for every source instance I , the recursive (IDB) predicates of $\pi_{\mathcal{M}}$ compute the canonical universal solution for I under \mathcal{M} . Moreover, this canonical universal solution for I is equal to its own core, since it consists entirely of constants.
- \mathcal{M} has bounded parallel chase if and only if the associated Datalog program $\pi_{\mathcal{M}}$ is *bounded*, which means that there is a positive integer b such that, for every input database I , the bottom-up evaluation of $\pi_{\mathcal{M}}$ on I converges to the least fixed-point of $\pi_{\mathcal{M}}$ within at most b iterations.

We now show that the first two statements are equivalent. If \mathcal{M} has bounded parallel chase, then the Datalog program $\pi_{\mathcal{M}}$ is bounded. This implies that the IDB predicates of $\pi_{\mathcal{M}}$ are definable by a finite union of conjunctive queries over the source. Each member of this union gives rise to a full s-t tgd, and we can take \mathcal{M}' to be the set of these s-t tgds. Conversely, assume that \mathcal{M} is CQ-equivalent to a schema mapping \mathcal{M}' specified by a finite set of full s-t tgds. It follows that every relation in the canonical universal solution is defined by both an infinite union of conjunctive queries over the source (obtained by “unfolding” the Datalog program $\pi_{\mathcal{M}}$) and by a finite union of conjunctive queries over the source (obtained by the s-t tgds in \mathcal{M}'). We now recall one of the main results in [25] to the effect that a union of conjunctive queries is contained in another union of conjunctive queries if and only if every conjunctive query in the first union is contained in some conjunctive query in the second union. By applying this result twice, we can conclude that the Datalog program $\pi_{\mathcal{M}}$ is bounded, because there is a finite subset of conjunctive queries in the infinite union such that every conjunctive query in the infinite union is contained in some member of this finite subset. Consequently, \mathcal{M} has bounded parallel chase.

It is obvious that the second statement implies the third, and that the third implies the fourth. We complete the proof by showing that the fourth statement implies the second. Suppose that \mathcal{M} is CQ-equivalent to a schema mapping \mathcal{M}' specified by an SO tgd Σ' . We will show that we can “drop” from Σ' all function terms, and transform it to a schema mapping \mathcal{M}^* specified by a finite set of full s-t tgds while preserving CQ-equivalence. First, for every source instance I , we have that $I^{\Sigma} = \text{core}(I^{\Sigma'})$, where I^{Σ} and $I^{\Sigma'}$ denote the result of the parallel chase of I with Σ and with Σ' , respectively. This is so because $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$ and all values in I^{Σ} are constants. Moreover, because we care only about CQ-equivalence, we can assume that Σ' has no SO tgd parts with equalities in the premises (this follows from the definition of the chase with SO tgds in [11]). Hence we can assume that the only function terms are in the conclusions of the SO tgd parts of Σ' . Let ψ be one of the SO tgd parts in Σ' that has some function terms in its conclusion. Then, for every source instance I , if we chase I with Σ' , all tuples generated by the SO tgd part ψ will contain nulls. So, they cannot be in $\text{core}(I^{\Sigma'})$. Let Σ'' be the SO tgd obtained from Σ' by removing the SO tgd part ψ . Then the result $I^{\Sigma''}$ of the chase of I with Σ'' still contains $\text{core}(I^{\Sigma'})$, but also is a subset of $I^{\Sigma'}$. Hence $I^{\Sigma'} \leftrightarrow I^{\Sigma''}$, and so $\Sigma' \equiv_{\text{CQ}} \Sigma''$. We can continue this process until all SO tgd parts with function terms have been removed from Σ' . The result is a finite set of full s-t tgds that is CQ-equivalent to \mathcal{M} . \square

In [13], it was shown that testing a Datalog program for boundedness is an undecidable problem. By combining this result with the reduction in the second part of Theorem 3.15, we obtain the following result.

PROPOSITION 4.3. *The following problem is undecidable: given a schema mapping \mathcal{M} specified by a finite set of full s-t tgds and full target tgds, is there a schema mapping \mathcal{M}' specified by s-t tgds such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$?*

4.2 SO tgds

In this section, we characterize when a schema mapping specified by an SO tgd is CQ-equivalent to a schema mapping specified by s-t tgds. We introduce next a boundedness property, called *bounded f-block size*, which will turn out to be precisely the necessary and sufficient condition for such equivalence.

DEFINITION 4.4. (1) If K is a target instance, then the *Gaifman graph of facts of K* is a graph whose nodes are the facts of K , and

with an edge between two facts if they have a null in common. A *fact block* (or simply an *f-block*) of K is a connected component of the Gaifman graph of facts of K .

(2) Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that \mathcal{M} has *bounded f-block size* if there is a positive integer b such that for every source instance I , the core of the universal solutions for I under Σ (if one exists) has f-block size at most b .

The following simple proposition gives a sufficient condition for bounded f-block size.

PROPOSITION 4.5. *If \mathcal{M} is a mapping specified by a finite set of s-t tgds, then \mathcal{M} has bounded f-block size.*

PROOF. Take the bound to be the maximum number of atoms in a conclusion of the s-t tgds. \square

It is important to note that the notion of *bounded f-block size* is different from the notion of *bounded block size* in [10]. If K is an instance, then the *Gaifman graph of the nulls of K* is an undirected graph in which: (1) the nodes are all the nulls of K , and (2) there exists an edge between two nulls whenever the nulls are adjacent in K . A *block* of nulls is the set of nulls in a connected component of the Gaifman graph of the nulls. For s-t tgds where the conclusions have at most e existential quantifiers, the maximal block size of a core of a universal solution is bounded by e . The next example shows a difference between bounded block size and bounded f-block size.

EXAMPLE 4.6. The SO tgd $\exists f \forall x \forall y (S(x, y) \rightarrow T(x, f(y)))$ has bounded block size (of size 1), but unbounded f-block size.

In contrast, it can be easily seen that bounded f-block size always implies bounded block size.

We now give another notion of boundedness, called *bounded support*, which we shall use in proving the characterization theorem in this section. Later, we shall make explicit use of bounded support and bounded f-block size in a characterization involving schema mappings specified by s-t tgds and target tgds.

In the following, we shall write $\|I\|$ to denote the number of facts in an instance I .

DEFINITION 4.7. We say that a schema mapping \mathcal{M} has *bounded support* if there exists a positive integer r such that for every instance I and instance J , if $J \rightarrow F_{\text{core}}^{\mathcal{M}}(I)$, then there is $I' \subseteq I$ such that $\|I'\| \leq r\|J\|$ and $J \rightarrow F_{\text{core}}^{\mathcal{M}}(I')$.

Intuitively, a schema mapping has bounded support (with bound r) if whenever an instance J “appears” in the core of the universal solutions for I , then it “appears” due to a “small” subinstance I' of I (where “small” means that I' is at most r times bigger than J). The following proposition, which we shall use in proving our next characterization theorem, gives two important cases where a schema mapping has bounded support.

PROPOSITION 4.8. *If \mathcal{M} is a schema mapping specified by a finite set of s-t tgds or an SO tgd, then \mathcal{M} has bounded support.*

PROOF. Since every finite set of s-t tgds is equivalent to an SO tgd, we need only prove the proposition for SO tgds. Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is an SO tgd. Set r to be the maximum number of atoms in the premise of a constraint (or an SO tgd part) in Σ . We show that r is the bound that witnesses that Σ has bounded support. Pick instances I and J . Consider the instance U obtained by doing the parallel chase of I with Σ . Such U is homomorphically equivalent to $F_{\text{core}}^{\mathcal{M}}(I)$ and therefore, if there is a

homomorphism $J \rightarrow F_{\text{core}}^{\mathcal{M}}(I)$, then there is also a homomorphism $h : J \rightarrow U$. Now consider I' consisting of the facts in I on which constraints in Σ fired to obtain the facts in $h(J)$. By construction, it is clear that $\|I'\| \leq r\|J\|$ and $J \rightarrow U'$ where U' is the result of the parallel chase of I' with Σ . Since U' is homomorphically equivalent to $F_{\text{core}}^{\mathcal{M}}(I')$ we have $J \rightarrow F_{\text{core}}^{\mathcal{M}}(I')$ as desired. \square

We shall also make use of the following simple proposition, whose proof is immediate from the definitions, and which asserts that the two boundedness notions introduced in this section are preserved under CQ-equivalence.

PROPOSITION 4.9. *Bounded f-block size and bounded support are preserved under CQ-equivalence.*

We are now ready to give our characterization of when a schema mapping specified by an SO tgd is CQ-equivalent to a schema mapping specified by s-t tgds.

THEOREM 4.10. *A schema mapping \mathcal{M} specified by an SO tgd is CQ-equivalent to a schema mapping specified by a finite set of s-t tgds if and only if \mathcal{M} has bounded f-block size.*

PROOF. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$. Assume first that \mathcal{M} is CQ-equivalent to a schema mapping \mathcal{M}' specified by a finite set of s-t tgds. By Proposition 4.5, \mathcal{M}' has bounded f-block size. So by Proposition 4.9, it follows that \mathcal{M} also has bounded f-block size.

Assume now that \mathcal{M} has bounded f-block size. Thus, there is a positive integer b such that every f-block in the core of a universal solution under \mathcal{M} has at most b facts. Assume there are at most r atoms in the premise of an SO tgd part of Σ . Say that an instance I is *small* if it has at most br facts.

We now define an s-t tgd $\tau_{I,B}$ whenever I is a source instance and B is a target instance. Fix a one-to-one function that maps every value (constant or null) onto a variable, and denote by v_c the variable corresponding to the value c . Let α be the conjunction of the source atoms $P(v_{c_1}, \dots, v_{c_k})$ such that $P(c_1, \dots, c_k)$ is a fact in I . Similarly, let β be the conjunction of the target atoms $Q(v_{c_1}, \dots, v_{c_m})$ such that $Q(c_1, \dots, c_m)$ is a fact in B . Let $\tau_{I,B}$ be the s-t tgd $\alpha \rightarrow \exists \bar{y} \beta$, where \bar{y} consists of the variables in β but not α .

Let S be a finite set that contains an isomorphic copy of all small source instances. Let Σ' be the set of all s-t tgds $\tau_{I,B}$ where $I \in S$, and B is an f-block in the core of the universal solution for I under \mathcal{M} . It is clear that Σ' is finite. Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$.

Let I be a source instance. Let I^Σ be the result of chasing I with the SO tgd Σ (this chase is described in [11]). Let $I^{\Sigma'}$ be the result of parallel-chasing I with Σ' . We need only show that $I^\Sigma \leftrightarrow I^{\Sigma'}$.

We begin by showing that $I^\Sigma \rightarrow I^{\Sigma'}$. It is enough to show that $\text{core}(I^\Sigma) \rightarrow I^{\Sigma'}$, since $I^\Sigma \leftrightarrow \text{core}(I^\Sigma)$. Let B be an arbitrary f-block of $\text{core}(I^\Sigma)$. Since Σ has bounded f-block with bound b , it follows that $\|B\| \leq b$. By Proposition 4.8 (and the fact, shown in the proof of Proposition 4.8, that the bound r in Definition 4.7 is our value of r , namely, the maximal number of atoms in the premise of an SO tgd part of Σ), it follows that there is a small instance $I' \subseteq I$ such that $B \rightarrow \text{core}((I')^\Sigma)$.

Since B is an f-block in a core, it follows that there is no homomorphism from B into a proper subinstance of B . Therefore, since $B \rightarrow \text{core}((I')^\Sigma)$, the homomorphism that maps B into $\text{core}((I')^\Sigma)$ simply renames the nulls in a one-to-one manner (and of course, being a homomorphism, maps each constant onto itself). Thus, (up to a renaming of nulls), B is a subinstance of an f-block B' of $\text{core}((I')^\Sigma)$. Therefore, $\tau_{I',B}$ is a logical consequence of $\tau_{I',B'}$, and so of Σ' . Hence, $B \rightarrow (I')^{\Sigma'}$. But also $(I')^{\Sigma'} \rightarrow I^{\Sigma'}$,

since $I' \subseteq I$. Since $B \rightarrow (I')^{\Sigma'}$ and $(I')^{\Sigma'} \rightarrow I^{\Sigma'}$, it follows that $B \rightarrow I^{\Sigma'}$. Let h_B be a homomorphism that maps B to $I^{\Sigma'}$. Since each h_B maps each constant onto itself, and since no two distinct f-blocks share a null, the function $h = \cup h_B$ is well-defined, and is a homomorphism from $\text{core}(I^{\Sigma})$ to $I^{\Sigma'}$. So $\text{core}(I^{\Sigma}) \rightarrow I^{\Sigma'}$, which, as we noted, is sufficient to show $I^{\Sigma} \rightarrow I^{\Sigma'}$.

We now show $I^{\Sigma'} \rightarrow I^{\Sigma}$. Define Σ'' to consist of the formulas $\tau_{I',B'}$ where as before $I' \in S$, but where now B' is $(I')^{\Sigma}$. It is straightforward to see that Σ'' is logically equivalent to Σ' .

We now show that Σ logically implies Σ'' . Let $\tau_{I',B'}$ be an arbitrary member of Σ'' . Since the result of chasing I' with Σ is B' , this tells us that the result of chasing the premise of $\tau_{I',B'}$ with Σ gives the conclusion of $\tau_{I',B'}$. So by well-known properties of the chase, it follows that Σ logically implies $\tau_{I',B'}$. Since $\tau_{I',B'}$ is an arbitrary member of Σ'' , this tells us that Σ logically implies Σ'' . So Σ logically implies Σ' , since Σ' and Σ'' are logically equivalent.

Since Σ logically implies Σ' , it follows that $I^{\Sigma'} \rightarrow I^{\Sigma}$ (this is because, intuitively, in doing the chase of I with Σ we can replace Σ by $\Sigma \cup \Sigma'$ and do the chase with Σ' first). This was to be shown. \square

We do not know whether it is decidable if a schema mapping specified by an SO tgd has bounded f-block size. However, if we know the f-block size b , then the proof of Theorem 4.10 gives a procedure for constructing a CQ-equivalent schema mapping specified by a finite set of s-t tgds.

4.3 s-t tgds and target tgds with terminating chase

In this section, we characterize when a schema mapping specified by s-t tgds and target tgds (and having terminating chase) is CQ-equivalent to a schema mapping specified by s-t tgds.

We need one more notion of boundedness.

DEFINITION 4.11. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping. We say that \mathcal{M} has *bounded core chase* if there is a positive integer b such that for every source instance I , the core chase of I with Σ terminates in at most b steps.

Note that in the full case, the core chase is simply the parallel chase, so bounded core chase is the same as bounded parallel chase. Intuitively, a schema mapping has bounded core chase if it has “no recursion”. Clearly, a sufficient condition for bounded core chase is that the schema mapping be specified only by s-t tgds. The next theorem gives a necessary condition for a schema mapping specified by s-t tgds and target tgds to have a bounded core chase. Although we are giving this theorem as a tool in proving our characterization theorems, it is interesting in its own right, since it provides a sufficient condition for a schema mapping specified by s-t tgds and target tgds to be CQ-equivalent to a schema mapping specified by an SO tgd.

THEOREM 4.12. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping where Σ is the union of a finite set of s-t tgds and a finite set of target tgds. If \mathcal{M} has bounded core chase then \mathcal{M} is CQ-equivalent to a schema mapping specified by an SO tgd.

PROOF. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping with $\Sigma = \Sigma_{st} \cup \Sigma_t$, where Σ_{st} is a finite set of s-t tgds and Σ_t is a finite set of target tgds (not necessarily weakly-acyclic). Assume that \mathcal{M} has bounded core chase with bound $b > 1$ (if $b = 1$ then the target constraints never fire, and $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}_{st}$, where $\mathcal{M}_{st} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$). Then it is easy to verify that \mathcal{M} is CQ-equivalent to the schema mapping specified by the composition

$$\Sigma_{st} \circ \Sigma_{t_1} \circ \Sigma_{t_2} \circ \dots \circ \Sigma_{t_{b-1}}$$

on schemas $S, T_1, \dots, T_b = T$ where T_i is a copy of T and where Σ_{t_i} is the union of copy constraints between T_i and T_{i+1} and the constraints Σ_t modified by replacing the relation symbols in the premises by the corresponding relation symbols in T_i and the relation symbols in the conclusions by the corresponding relation symbols in T_{i+1} . Set Σ' to be the SO tgd that expresses the composition above (it was shown in [11] that the composition of an arbitrary finite number of schema mappings, each specified by a finite set of s-t tgds, is a schema mapping specified by an SO tgd). Let $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$. Then \mathcal{M} is CQ-equivalent to \mathcal{M}' . \square

It is interesting to note that in contrast to Proposition 4.9, which states that bounded f-block size and bounded support are preserved under CQ-equivalence, we have the following negative proposition.

PROPOSITION 4.13. *Bounded core chase is not preserved under CQ-equivalence.*

PROOF. Let Σ consist of the s-t tgd $D(x) \rightarrow \exists u(F(x, u) \wedge G(u, u))$ and the target tgds $F(x, u) \wedge F(y, v) \rightarrow \exists w(G(u, w) \wedge G(v, w) \wedge G(w, w))$ and $F(x, u) \wedge G(u, w) \rightarrow F(x, v)$. Let Σ' consist of the s-t tgd $D(x) \rightarrow \exists u F(x, u)$ and the target tgds $F(x, u) \rightarrow G(u, u)$ and $F(x, u) \wedge F(y, v) \rightarrow F(x, v)$. It will be shown in the full version of the paper that (1) Σ does not have bounded core chase, (2) Σ' has bounded core chase. and (3) Σ and Σ' are CQ-equivalent \square

We now state and prove our final characterization theorem. It gives several exact criteria as to when a schema mapping specified by a finite set of s-t tgds and target tgds (and having terminating chase) is CQ-equivalent to a schema mapping specified by a finite set of s-t tgds only.

THEOREM 4.14. Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be a schema mapping such that Σ is the union of a finite set of s-t tgds with a finite set of target tgds and such that the core chase with Σ is terminating. Then the following statements are equivalent:

1. There exists $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ where Σ' is a finite set of s-t tgds such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.
2. There exists $\mathcal{M}' = (\mathbf{S}, \mathbf{T}, \Sigma')$ where Σ' is an SO tgd with bounded f-block size such that $\mathcal{M} \equiv_{\text{CQ}} \mathcal{M}'$.
3. Σ has bounded support and bounded f-block size.
4. Σ has bounded core chase and bounded f-block size.

PROOF. We first show that (1) \Rightarrow (2). Assume that (1) holds. Since each finite set of s-t tgds can be expressed by an SO tgd, there is an SO tgd logically equivalent to Σ' . It has bounded f-block size by Proposition 4.5.

We now show that (2) \Rightarrow (3). Assume that (2) holds. By Proposition 4.8, Σ' has bounded support. So (3) follows by Propositions 4.9. We now show that (4) \Rightarrow (1). Assume that (4) holds. Since Σ has bounded core chase, it follows from Theorem 4.12 that Σ is CQ-equivalent to an SO tgd Σ'' . By Proposition 4.9, Σ'' has bounded f-block size. By Theorem 4.10, Σ'' is CQ-equivalent to some set Σ' of s-t tgds.

We now show that (3) \Rightarrow (4). Assume that (3) holds. Assume that Σ has bounded support with bound r , and that Σ has bounded f-block size with bound b .

We write I^{Σ} for the result of the core chase of I with Σ and I_m^{Σ} for the result of doing the core chase of I with Σ for m steps. Notice that because Σ has terminating core chase, I^{Σ} is defined and $I^{\Sigma} \cong F_{\text{core}}^{\mathcal{M}}(I)$ for every I .

We will compute m , which depends only on Σ , and we will show that $I^{\Sigma} \rightarrow I_m^{\Sigma}$. Since also $I_m^{\Sigma} \rightarrow I^{\Sigma}$ and both I^{Σ} and I_m^{Σ} are cores,

it follows that they are isomorphic. Since $I^\Sigma \models \Sigma$, we also have $I_m^\Sigma \models \Sigma$ and therefore the core chase of I with Σ ends in m steps, as desired.

For every f-block B , define K_B to be the set of source instances I minimal under set inclusion such that $B \rightarrow F_{\text{core}}^M(I)$. Let K be $\bigcup_{B \in S} K_B$ where S is the set of f-blocks B such that there exists a source instance I satisfying $B \rightarrow F_{\text{core}}^M(I)$.

Then $I \in K$ implies $\|I\| \leq rb$, since if $I \in K$ and $B \rightarrow F_{\text{core}}^M(I)$ for some $B \in S$, then $\|I\| \leq r\|B\| \leq rb$ by minimality of I and because Σ has bounded support and bounded f-block size with bounds r and b respectively. It follows that K has finitely many instances up to non-constant-preserving isomorphisms. Therefore, there exists m such that the core chase of every instance in K terminates in at most m steps.

Now pick some instance I . Set S_I to be the set of f-blocks B in I^Σ . For every $B \in S_I$, pick $I_B \in K_B$ such that $I_B \subseteq I$. There must be such I_B by definition of K_B . Then there is a homomorphism $h_B : B \rightarrow I_m^\Sigma$ because

$$B \rightarrow (I_B)^\Sigma = (I_B)_m^\Sigma \rightarrow I_m^\Sigma.$$

Let h be $\bigcup_{B \in S_I} h_B$. Since $I^\Sigma = \bigcup_{B \in S_I} B$ and every $B \in S_I$ is a f-block, h is a homomorphism $I^\Sigma \rightarrow I_m^\Sigma$ as desired. \square

Note that statement (1) of Theorem 4.14 is undecidable (even when the tgds in Σ are full) by Proposition 4.3. Hence, each of the statements (1)–(4) of Theorem 4.14 are undecidable.

5. Conclusions and Open Problems

In this paper, we have taken the first steps towards developing a theory of schema-mapping optimization. To this effect, we defined and studied the notions of data-exchange equivalence and CQ-equivalence of schema mappings, which are more relaxed notions of “equivalence” than logical equivalence. Intuitively, two schema mappings are data-exchange equivalent if they are indistinguishable for data-exchange purposes, and two schema mappings are CQ-equivalent if they are indistinguishable for the purpose of answering conjunctive queries. Furthermore, we have shown that CQ-equivalence can be viewed as simply a slightly relaxed version of data-exchange equivalence.

The long-term goal is to develop techniques that enable us to optimize schema mappings by converting them into “simpler” schema mappings that are equivalent in one of these senses. In this paper, we have given characterizations for when some important, well-studied classes of schema mappings are CQ-equivalent to a schema mapping specified only by s-t tgds.

Much work remains to be done. One obvious open problem is how to give similar characterizations for data-exchange equivalence rather than CQ-equivalence. Further, in view of our undecidability results, it is a worthy goal to find useful heuristics for optimizing schema mappings that arise in practice by converting them into simpler schema mappings.

6. References

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] M. Arenas and L. Libkin. XML Data Exchange: Consistency and Query Answering. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 13–24, 2005.
- [3] P. A. Bernstein. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.
- [4] P. A. Bernstein and S. Melnik. Model Management 2.0: Manipulating Richer Mappings. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1–12, 2007.
- [5] A. Cali, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data Integration under Integrity Constraints. *Inf. Syst.*, 29(2):147–163, 2004.
- [6] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *ACM Symposium on Principles of Database Systems (PODS)*, 2008.
- [7] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *International Conference on Database Theory (ICDT)*, pages 225–241, 2003.
- [8] R. Fagin. Inverting Schema Mappings. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 50–59, 2006. Full version to appear, *ACM Transactions on Database Systems (TODS)*.
- [9] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science (TCS)*, 336(1):89–124, 2005.
- [10] R. Fagin, P. G. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. *ACM Transactions on Database Systems (TODS)*, 30(1):174–210, 2005.
- [11] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing Schema Mappings: Second-order Dependencies to the Rescue. *ACM Transactions on Database Systems (TODS)*, 30(4):994–1055, 2005.
- [12] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Quasi-Inverses of Schema Mappings. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 123–132, 2007.
- [13] H. Gaifman, H. G. Mairson, Y. Sagiv, and M. Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, 1993.
- [14] G. Gottlob. Computing Cores for Data Exchange: New Algorithms and Practical Solutions. In *ACM Symposium on Principles of Database Systems (PODS)*, 2005.
- [15] G. Gottlob and A. Nash. Data exchange: Computing cores in polynomial time. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 40–49, 2006.
- [16] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 805–810, 2005.
- [17] P. G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 61–75, 2005.
- [18] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
- [19] J. Madhavan and A. Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, pages 572–583, 2003.
- [20] S. Melnik. *Generic Model Management: Concepts and Algorithms*, volume 2967 of *Lecture Notes in Computer Science*. Springer, 2004.
- [21] S. Melnik, A. Adya, and P. A. Bernstein. Compiling Mappings to Bridge Applications and Databases. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 461–472, 2007.
- [22] A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. *ACM Transactions on Database Systems (TODS)*, 32(1):4, 2007.
- [23] A. Nash, A. Deutsch, and J. Remmel. Data exchange, data integration, and chase. Technical Report CS2006-0859, UC San Diego, 2006.
- [24] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *International Conference on Very Large Data Bases (VLDB)*, pages 598–609, 2002.
- [25] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.
- [26] O. Shmueli. Equivalence of DATALOG queries is undecidable. *J. Log. Program.*, 15(3):231–241, 1993.