Checking Containment of Schema Mappings (Preliminary Report)

Andrea Calì[3,1] and Riccardo Torlone[2]

Oxford-Man Institute of Quantitative Finance, University of Oxford, UK
Dip. di Informatica e Automazione, Università Roma Tre, Italy
Computing Laboratory, University of Oxford, UK

andrea.cali@oxford-man.ox.ac.uk, torlone@dia.uniroma3.it

Abstract. In data exchange, data are materialised from a source schema to a target schema, according to suitable source-to-target constraints. Constraints are also expressed on the target schema to represent the domain of interest. A schema mapping is the union of the source-to-target and of the target constraints. In this paper, we address the problem of containment of schema mappings for data exchange, which has been recently proposed in this framework as a step towards the optimization of data exchange settings. We refer to a natural notion of containment that relies on the behaviour of schema mappings with respect to conjunctive query answering, in the presence of so-called LAV TGDs as target constraints. Our contribution is a practical technique for testing the containment based on the existence of a homomorphism between special "dummy" instances, which can be easily built from schema mappings. We argue that containment of schema mappings is decidable for most practical cases, and we set the basis for further investigations in the topic. This paper extends the preliminary results of [4].

1 Introduction

In distributed database applications, restructuring information from a certain format into a desired one is a complex task. Consequently, there is a strong need for methods and tools supporting the problem of transforming data coming in different formats from multiple sources. A clean formalisation of this problem has been recently proposed under the name of *data exchange* [8]. In this approach, a first order logic specification, called *schema mapping*, describes declaratively how data structured under one schema (the *source schema*) are to be transformed into data structured under a different schema (the *target schema*): the goal is to take a given instance of the source schema and translate it into an instance of the target schema such that the schema mapping and the constraints (a.k.a. dependencies) over the target are satisfied.

For example, if we consider a source database schema with a single relation $employee(Name, Salary, Department)$ and a target database composed by the pair of relations $person(Name, Address)$ and $salary(Employee, Amount)$, then a typical schema mapping would be represented by the following *source-to-target*

dependency σ_1: $employee(X, Y, Z) \rightarrow \exists W\ person(X, W), salary(X, Y)$. This dependency, which is a TGD (tuple-generating dependency), states that for each tuple $employee(e, s, d)$ in the source there must be two tuples in the target: $person(e, a)$ and $salary(e, s)$, where a is a new, unknown value. A target dependency, stating that each employee name occurring in relation *salary* must occur in relation *person* in the first position, can also be expressed with a dependency σ_2 of the same form: $salary(X, Y) \rightarrow \exists Z\ person(X, Z)$. In [7] it has been shown that, for a certain class of target dependencies, given a source instance I over the source schema, a "most general" solution to this problem, called universal solution, can be computed by applying to I a well-known procedure called *chase*. Intuitively, this procedure generates the solution by enforcing in I the satisfaction of the dependencies. For example, if the instance of the above source schema consists of the single tuple $employee(john, 50, toys)$ then, in order to enforce the satisfaction of source-to-target dependency, the chase generates the following tuples over the target schema: $person(john, v_1)$ and $salary(john, 50)$, where v_1 denotes a labelled null value. The answer to a conjunctive query on a universal instance, without considering result tuples that have at least one null, coincides with the correct answers (called *certain answers*) to the query.

A natural question that arises in this scenario is the following: given a schema mapping M_1, is there a different schema mapping M_2 over the same source and target that has the same solution of M_1 and provides a simpler and, possibly, more efficient way to generate such solution? As an example, it is possible to see that the above schema mapping is equivalent to the one obtained by replacing σ_1 with the dependency σ_3 defined as $employee(X, Y, Z) \rightarrow salary(X, Y)$ since, intuitively, the generation of a tuple in *person* is guaranteed, during the chase, by the target dependency σ_2.

More precisely, we define a *schema mapping* as the union of the source-to-target and target dependencies. We say that a schema mapping M is *contained into* another mapping M' if for every source instance, the answers to every conjunctive query q under M are a subset of the answers to q under M'. Two schema mappings are said to be *equivalent* when they are contained into each other.

Different variants of this problem have been addressed by Fagin et al. in [9], where three different notions of schema mapping equivalence are proposed, and a characterisation of the problem in some special cases is presented; however, except for one case, no positive results are given. However, there is still space for positive results in cases that are interesting from the practical point of view, as we will show in the following.

In this paper, we address the problem of containment of schema mappings for data exchange. We first propose a definition of schema mapping containment which corresponds to the notion of conjunctive-query equivalence proposed in [9]. We provide some basic results that follow from general properties of first-order formulae and known characterisations of the data exchange problem. We then consider the case of LAV TGDs, i.e., TGDs that have a single atom in the left-hand side. LAV TGDs are a quite general class of TGDs that, together with some additional dependencies (which we do not consider here) are able to rep-

resent most known formalisms for ontology reasoning, as shown in [5] for *Linear Datalog*$^{\pm}$, a language whose rules are a special case of LAV TGDs, with a single atom in the left-hand side. Notice that, under LAV TGDs (and Linear Datalog$^{\pm}$ as well), due to the existentially quantified variables in the right-hand side of TGDs, the chase does not always terminate, forcing us to reason on instances of unbounded size for the target schema. Notice also that LAV TGDs generalise the well-known class of *inclusion dependencies* [1]. We propose a practical technique for testing the containment of schema mapping in this case: our technique relies on the construction of a finite set of "dummy" instances for the source schema, which depends only on the source-to-target mappings. The paper ends with a discussion on a number of challenging problems that naturally follow from these preliminary steps. To the best of our knowledge, this is the first result of this kind in this context that follows the line of previous approaches to the problem of checking the equivalence and optimisation of relational expressions representing queries [2, 3].

The rest of the paper is organised as follows. In Section 2 we provide the basic definitions and recall some useful results on the data exchange problem. Then, in Section 3, we introduce the problem of schema mapping containment and present our technique for testing the containment. Finally, in Section 4 we discuss future directions of research and draw some conclusions.

2 Preliminaries

2.1 Basics

A *(relational) schema* $\mathbf{S}$ is composed by a set of *relational predicates* $r(A_1, \ldots, A_n)$, where r is the name of the predicate and $A_1, \ldots, A_n$ are its *attributes*. A predicate having n attributes is said to be n-ary or, equivalently, to have *arity* n. An instance of a relation $r(A_1, \ldots, A_n)$ is a set of tuples, each of which associates with each A_i a value. An instance I of a schema $\mathbf{S}$ contains an instance of each relation in $\mathbf{S}$. In the following, except when explicitly stated, we assume that instances are finite. We shall consider values from two (infinite) domains: a set of constants C, and a set of *labelled nulls* N; the latter are intended to be placeholders for unknown constants, therefore they can be interpreted also as existentially quantified variables.

As usual in data exchange, we will focus on two special kind of constraints (a.k.a. dependencies): tuple generating dependencies (TGDs) and equality generating dependencies (EGDs), as it is widely accepted that they include most of the naturally-occurring constraints on relational databases. We will use the symbol $\bar{X}$ to denote a sequence (or, with slight abuse of notation, a set) of variables $X_1, \ldots, X_k$. A TGD has the form: $\forall \bar{X}(\phi(\bar{X}) \rightarrow \exists \bar{Y}(\psi(\bar{X}, \bar{Y}))$ where $\phi(\bar{X})$ and $\psi(\bar{X}, \bar{Y})$ are conjunction of atomic formulas, and allows (together with suitable EGDs) the specification of foreign key, inclusion, and multivalued dependencies, among others. An EGD has the form: $\forall \bar{X}(\phi(\bar{X}) \rightarrow (X_1 = X_2))$ where $\phi(\bar{X})$ is a conjunction of atomic formulas and X_1, X_2 are variables in $\bar{X}$: it strictly generalises key constraints and functional dependencies [1], as it can be easily seen. In

both TGDs and EGDs, the left-hand side is called *body*, and the right-hand side is called *head*. We usually omit the universal quantifiers to simplify the notation.

A LAV TGD is a TGD having a single atom on the left-hand side, i.e., of the form $r(\bar{X}) \rightarrow \exists \bar{Y}\, \psi(\bar{X}, \bar{Y})$. Notice that variables can be repeated in the body atom.

In the following, for space reasons, we will focus on TGDs only; significant classes of EGDs can be added without changing our results; see, for instance, the class of keys and non-conflicting TGDs in [5].

We now provide the important notion of *homomorphism*. We consider instances having constants and nulls as values.

Definition 1. *A* homomorphism *from an instance I to an instance J, both of the same schema, is a function h from constant values and nulls occurring in I to constant values and nulls occurring in J such that: (i) it is the identity on constants, and (ii) $h(I) \subseteq J$, where for a tuple $t = r(c_1, \ldots, c_n)$ we denote $h(t) = r(h(c_1), \ldots, h(c_n))$, and for a set of tuples I, $h(I) = \{h(t) \mid t \in I\}$. An* isomorphism *is a surjective and injective homomorphism.*

If there is a homomorphism from an instance I to an instance J, we write $I \overset{\mathsf{hom}}{\rightarrow} J$. We use the notation $I \overset{\mathsf{iso}}{\leftrightarrow} J$ to denote that there is an isomorphism from I to J (and therefore another one from J to I).

2.2 Schema mappings

In the relational-to-relational data exchange framework [8], a schema mapping (also called a data exchange setting) is defined as follows. In the following, we will assume that both souce-to-target TGDs and target TGDs are LAV TGDs.

Definition 2. *A schema mapping is a 4-tuple: $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where:*

(1) $\mathbf{S}$ *is a source schema,*
(2) $\mathbf{T}$ *is a target schema,*
(3) Σ_{st} *is a finite set of* s-t *(source-to-target) LAV TGDs* $\forall \bar{X}(\phi(\bar{X}) \rightarrow \exists \bar{Y}(\chi(\bar{X}, \bar{Y})))$ *where* $\phi(\bar{X})$ *is a conjunction of atomic formulas over* $\mathbf{S}$ *and* $\chi(\bar{X}, \bar{Y})$ *is a conjunction of atomic formulas over* $\mathbf{T}$*, and*
(4) Σ_t *is a finite set of LAV target TGDs over* $\mathbf{T}$*.*

Given an instance I of $\mathbf{S}$, a solution for I under M is an instance J of $\mathbf{T}$ such that (I, J) satisfies Σ_{st} and J satisfies Σ_t. A solution in general has values in $\mathsf{C} \cup \mathsf{N}$ (nulls or constants). In general, there are many possible solutions, possibly an infinite number, for I under a schema mapping M. A solution J is *universal* if there is a homomorphism from J to every other solution for I under M [7].

In [7] it was shown that a universal solution of I under M certainly exists when the dependencies in Σ_t are either EGDs or *weakly-acyclic* TGDs, a class of TGDs which admits limited forms of cycles. In this case, if a solution exists, a universal solution can be computed by applying the *chase procedure* [14] to I using $\Sigma_{st} \cup \Sigma_t$. The chase[1] is a fundamental tool that has been

[1] With *chase*, we refer interchangeably to the procedure or to its resulting instance.

widely used to investigate several database problems such as, checking implication of constraints, checking equivalence of queries, query optimisation, and computing certain answers in data integration settings (see, e.g., [12]). This procedure takes as input an instance D and generates another instance by iteratively applying *chase steps* based on the given dependencies. In particular, a TGD $\forall \bar{X}(\phi(\bar{X}) \rightarrow \exists \bar{Y}(\psi(\bar{X}, \bar{Y})))$ can be applied at step k of the chase, to the partial chase D_{k-1} obtained at the previous step, if there is a homomorphism h from $\phi(\bar{X})$ to D_{k-1}; in this case, the result of its application is $D_k = D_{k-1} \cup h'(\psi(\bar{X}, \bar{Y}))$, where h' is the extension of h to $\bar{Y}$ obtained by assigning fresh labelled nulls to the variables in $\bar{Y}$. The chase of I with respect to a set of dependencies Σ, denoted by $\text{chase}_\Sigma(I)$, is the instance obtained by applying all applicable chase steps exhaustively to I. Notice that such instance may be infinite.

Universal solutions are particularly important also for query answering since a conjunctive query[2] q over the target scheme can be evaluated against any universal solution. More precisely, given a schema mapping $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, an instance I of $\mathbf{S}$, and a conjunctive query q over $\mathbf{T}$, the *certain answers* of q on I under M is the set of all tuples of constants occurring in every solution for I under M. Fagin et al. [7] have shown that if q is a union of conjunctive queries, the certain answers of q on I under M, denoted $\text{cert}(q, I, M)$ can be obtained by evaluating q over any universal solution J for I under M and then eliminating all the tuples with nulls (in symbols: $\text{cert}(q, I, M) = q^{\downarrow}(J)$).

3 Schema Mapping Containment and Equivalence

3.1 Definitions and preliminary results

In this section we consider two schema mappings $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ and $M' = (\mathbf{S}, \mathbf{T}, \Sigma'_{st}, \Sigma'_t)$. Also, we only consider conjunctive queries. Our notion of containment refers to the behaviour of schema mapping with respect to query answering.

Definition 3 (Containment and equivalence of schema mappings [9]). *We say that M is contained in M', in symbols $M \subseteq M'$, if for every instance I of $\mathbf{S}$ and every query q over $\mathbf{T}$ we have that the certain answers of q on I under M are contained in the certain answers of q on I under M', in symbols*

$$\forall I \forall q \ \text{cert}(q, I, M) \subseteq \text{cert}(q, I, M')$$

M and M' are equivalent, in symbols $M \equiv M'$, if both $M \subseteq M'$ and $M' \subseteq M$.

The following preliminary result on schema mapping containment states that: *(i)* we can focus on universal solutions only and *(ii)* a necessary and sufficient condition for containment relies on the existence of a homomorphism between universal solutions.

[2] We shall henceforth consider conjunctive queries (a.k.a. select-project-join queries) only, for which we refer the reader to [1].

Lemma 1. *Given two schema mappings M, M' as above, we have that $M \subseteq M'$ if and only if, for every instance I of $\mathbf{S}$, there exist two universal solutions J, J' for I under M and M' respectively, such that $J \overset{\mathsf{hom}}{\rightarrow} J'$.*

Proof. (Only if). By contradiction, assume that $M \subseteq M'$, but there is no two universal solutions J, J' for some instance I under M, M' respectively. Let $J = \text{chase}_{\Sigma}(D)$ an $J' = \text{chase}_{\Sigma'}(D)$. Now, we proceed by induction on the number of applications of the chase step on I. *Base step.* Take the Boolean conjunctive query obtained by replacing every null in I with a distinct variable. Since $M \subseteq M'$, not only $J \models q$ (i.e., q has positive answer on I) trivially, but also by hypothesis we also have $J' \models q$. *Inductive step.* Assume that at the k-th application, the partial chase, denoted $\text{chase}_{\Sigma}^{(k)}(D)$ maps onto J' via some homomorphism. Consider $\text{chase}_{\Sigma}^{(k+1)}(D)$ and turn it into a query as above. Similarly, we get $\text{chase}_{\Sigma}^{(k+1)}(D) \overset{\mathsf{hom}}{\rightarrow} J'$. With this inductive argument we show $J \overset{\mathsf{hom}}{\rightarrow} J'$. *(If).* Let J, J' be universal solutions for an instance I under M, M' respectively, with $J \overset{\mathsf{hom}}{\rightarrow} J'$. Given a query q, if for a tuple t if holds $t \in \text{cert}(q, I, M)$, this amounts to say $t \in q^{\downarrow}(J)$, or equivalently there is a homomorphism from q to J that sends the head variables of q to constants. Since there is also another homomorphism from J to J', we have also $t \in q^{\downarrow}(J)$ and therefore $t \in \text{cert}(q, I, M')$.

The next result easily follows from the lemma above and the results on the generation of universal solutions for data exchange [7]. We shall make use of the notion of chase on the union of source-to-target and target TGDs; in this case, we implicitly assume that the schema is the union of the source and the target schema.

Lemma 2 (straightforward from [9]). *Given two schema mappings M, M' as above, we have $M \subseteq M'$ if and only for every instance I of $\mathbf{S}$, $\text{chase}_{\Sigma_{st} \cup \Sigma_t}(I) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'_{st} \cup \Sigma'_t}(I)$.*

Proof. It follows by Lemma 1 and the fact that $J = \text{chase}_{\Sigma_{st} \cup \Sigma_t}(I)$ is a universal solution for I under M and $J' = \text{chase}_{\Sigma'_{st} \cup \Sigma'_t}(I)$ is a universal solution for I under M' [7].

Example 1. Let us consider the following schema mapping $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$:

$$\begin{aligned} M = \{ \quad & \mathbf{S} = \{r_1(A, B)\}, \quad \mathbf{T} = \{r_2(C, D), R_3(M, F)\} \\ & \Sigma_{st} = \{r_1(X, Y) \rightarrow \exists Z\, r_2(Y, Z), r_1(X, Y) \rightarrow \exists Z\, r_3(Z, Y)\} \\ & \Sigma_t = \{r_2(X, Y) \rightarrow \exists Z\, r_3(Z, X)\} \quad \} \end{aligned}$$

and the generic database instance $I = \{r_1(a_i, b_i)\}$ $(1 \leq i \leq n)$ of $\mathbf{S}$. The universal solution generated by applying the chase process to I using the given dependencies is: $J = \{r_2(b_i, v_i), r_3(v'_i, b_i), r_3(v''_i, b_i)\}$ $(1 \leq i \leq n)$, where the v_i are labelled nulls. Let us now consider the following schema mapping defined over the same source and target schemas:

$$\begin{aligned} M' = \{ \quad & \mathbf{S} = \{r_1(A, B)\}, \quad \mathbf{T} = \{r_2(C, D), r_3(M, F)\} \\ & \Sigma'_{st} = \{r_1(X, Y) \rightarrow \exists Z\, r_2(Y, Z)\} \\ & \Sigma'_t = \{r_2(X, Y) \rightarrow \exists Z\, r_3(Z, X)\} \quad \} \end{aligned}$$

We have that the universal solution generated by applying the chase process to I using $\Sigma'_{st} \cup \Sigma'_t$ is: $J' = \{r_2(b_i, v_i), r_3(v'_i, b_i)\}$ $(1 \leq i \leq n)$. It is easy to see that J' and J are homomorphically equivalent, i.e., $J \overset{\mathsf{hom}}{\rightarrow} J'$ and $J' \overset{\mathsf{hom}}{\rightarrow} J$. This is actually true for every instance of $\mathbf{S}$ and so $M \equiv M'$. Notice that M' is "more compact" than M and so the chase using the dependencies in M' requires a lower number of steps (we do not discuss on optimality criteria related to containment in this paper).

Finally, consider a schema mapping M'' as follows:

$$\begin{array}{rl} M'' = \{ & \mathbf{S} = \{r_1(A, B)\}, \quad \mathbf{T} = \{r_2(C, D), r_3(E, F)\} \\ & \Sigma''_{st} = \{r_1(X, Y) \rightarrow \exists Z\, r_2(Y, Z)\} \\ & \Sigma''_t = \{r_2(X, Y) \rightarrow \exists Z\, r_3(Y, X)\} \quad \} \end{array}$$

By applying the chase process to I using these dependencies we obtain the target instance: $J'' = \{r_2(b_i, v_i), r_3(v_i, b_i)\}$ $(1 \leq i \leq n)$. Now, we have $J' \overset{\mathsf{hom}}{\rightarrow} J''$ $(v'_i \mapsto v_i)$ but there is no homomorphism from J'' to J'. Again, this is valid in general and so $M'' \subseteq M'$ but $M' \not\subseteq M''$.

Clearly, the test of Lemma 2 is not usable in practice since it refers to an unbounded number of source instances, and moreover of unbounded size in general.

3.2 A practical test for containment

We now present a technique for deciding containment between two schema mappings. We are able to show that given a schema mapping $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, in order to test containment, it suffices to test it on a *finite* set of instances for $\mathbf{S}$, depending only on M, that we call *dummy instances* for M, denoted D_M.

Definition 4. *The set of* dummy databases D_M *for a schema mapping* $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ *is constructed as follows. For every relational symbol* r *of arity* n *appearing in the body of some TGD in* Σ_{st}*, the corresponding instances* D_r *are obtained by considering all possible tuples of the form* $r(z_1, \ldots, z_n)$*, where the* z_i *are freshly invented constants in* C*, possibly with repetitions.*

Example 2. Consider again $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ of Example 1. Then, the dummy set D_M contains four single-tuple instances: $\{r_1(z_1, z_2)\}$, $\{r_1(z_1, z_1)\}$, $\{r(z_2, z_2)\}$ and $\{r(z_2, z_1)\}$. We do not consider $\{r(z_2, z_2)\}$ or $\{r(z_2, z_1)\}$ (though it would not harm the decision algorithm that we are going to introduce in the following) because they can be obtained from the first two instances by null renaming.

We now show the main property of the dummy instances, showing that they are a useful tool for checking containment of schema mappings. Before stating this paper's main theorem, we need some intermediate results.

Lemma 3. *Consider two instances* D_1, D_2 *constructed with fresh and non-fresh constants, i.e., with values in* $\mathsf{C} \cup \mathsf{N}$*, and a set* Σ *of LAV TGDs. We then have*

$$\mathrm{chase}_\Sigma(D_1 \cup D_2) \overset{\mathsf{iso}}{\leftrightarrow} \mathrm{chase}_\Sigma(D_1) \cup \mathrm{chase}_\Sigma(D_2)$$

Without loss of generality, we assume that the two sets of nulls generated in $\mathrm{chase}_\Sigma(D_1)$ *and* $\mathrm{chase}_\Sigma(D_2)$ *have empty intersection.*

Proof. (sketch). The result is proved by observing that every tuple t in D_1 generates a fragment of the chase $\text{chase}_\Sigma(\{t\})$ which is totally independent of any other tuple in D_1 (the same holds, of course, for tuples in D_2). This because the application of a LAV TGD depends, by definition, *solely* on the single tuple to which it is applied, independently of the others.

From the above result we immediately get the following.

Lemma 4. *Consider two instances D_1 and D_2, and two sets of LAV TGDs Σ_1 and Σ_2. We have that (1) and (2) iff (3), where (1), (2), (3) are as follows. (1) $\text{chase}_{\Sigma_1}(D_1) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma_2}(D_1)$; (2) $\text{chase}_{\Sigma_1}(D_2) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma_2}(D_2)$; (3) $\text{chase}_{\Sigma_1}(D_1 \cup D_2) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma_2}(D_1 \cup D_2)$.*

We now come to our first main result.

Theorem 1. *Consider two schema mappings $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ and $M' = (\mathbf{S}, \mathbf{T}, \Sigma'_{st}, \Sigma_t)$. We have that $M \subseteq M'$ if and only if, for every instance D in the set of dummy instances D_M for M, we have $\text{chase}_\Sigma(D) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(D)$, where $\Sigma = \Sigma_{st} \cup \Sigma_t$ and $\Sigma' = \Sigma'_{st} \cup \Sigma'_t$.*

Proof. (sketch).
(Only if). It follows from Definition 3, since $\text{chase}_\Sigma(I)$ to $\text{chase}_{\Sigma'}(I)$ are both solutions for M, M' respectively (in particular, they are universal solutions).
(If). Consider a generic source instance I. From Lemma 3 it is easy to see that $\text{chase}_\Sigma(D) \overset{\mathsf{iso}}{\leftrightarrow} \bigcup_{t \in I} \text{chase}_\Sigma\{t\}$. Then, for each $t \in I$ that generates at least one tuple in the chase procedure, there is a dummy instance $D \in D_M$ such that $D \overset{\mathsf{iso}}{\leftrightarrow} \{t\}$. By hypothesis, for each $D \in D_M$ we have $\text{chase}_\Sigma(D) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(D)$, and therefore for every $t \in I$ we immediately have $\text{chase}_\Sigma(\{t\}) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(\{t\})$. By Lemma 3, it then holds $\text{chase}_\Sigma(D) \overset{\mathsf{iso}}{\leftrightarrow} \bigcup_{t \in I} \text{chase}_\Sigma\{t\} \overset{\mathsf{hom}}{\rightarrow} \bigcup_{t \in I} \text{chase}_{\Sigma'}\{t\} \overset{\mathsf{iso}}{\leftrightarrow} \text{chase}_{\Sigma'}(D)$, and by Lemma 2 we get $M \subseteq M'$.

The above result provides an insight into the containment problem, but since in general the chase of a dummy instance $D \in D_M$ can be infinite, it is not obvious if checking $\text{chase}_\Sigma(D) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(D)$ is decidable. The following result provides an answer to the question.

Theorem 2. *Given an instance D and two sets of LAV TGDs Σ_1 and Σ_2, such that $\text{chase}_{\Sigma_1}(D)$ is finite, checking whether $\text{chase}_{\Sigma_1}(D) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma_2}(D)$ is decidable.*

Proof. (sketch). The proof is a direct consequence of the results in [13]. If chase_{Σ_2} is finite, the result is trivial. Let us consider the case where chase_{Σ_2} is infinite. Preliminarly, we have to introduce the notion of *level* of a tuple in the chase of an instance I. The notion of level is inductively defined as follows. All tuples in I have level 0; if a tuple t_2 is added in a chase step applied on a tuple t_1 at level ℓ, then t_2 has level $\ell+1$. To check whether $\text{chase}_{\Sigma_1}(D) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma_2}(D)$, it suffices to

check whether there exists a homomorphism from $\text{chase}_{\Sigma_1}(D)$ (which is finite) to the (finite) segment of $\text{chase}_{\Sigma_2}(D)$ constitute by its first $|\Sigma_2| \cdot (W+1)^W$ levels, where W is the maximum arity of predicates appearing in Σ_2.

The following result is a direct consequence of Theorems 1 and 2.

Theorem 3. *Consider two schema mappings M and M' as above such that, for every instance D in the set of dummy instance D_M for M, $\text{chase}_{\Sigma_1}(D)$ is finite. We have that checking whether $M \subseteq M'$ is decidable.*

Example 3. Let us consider again the schema mappings $M = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ and $M' = (\mathbf{S}, \mathbf{T}, \Sigma'_{st}, \Sigma'_t)$ of Example 1. As shown in Example 2, the set of dummy instances is $D_M = \{D_1, D_2\}$ where $D_1 = \{r_1(z_1, z_2)\}$ and $D_2 = \{r_1(z_1, z_1)\}$. Let us now apply the chase procedure to D_1 and D_2 using the dependencies in Σ and Σ'. We obtain the instances: $\text{chase}_{\Sigma}(D_1) = \{r_2(z_2, \alpha_1), r_3(\alpha_2, z_2), r_3(\alpha_3, z_2)\}$, and $\text{chase}_{\Sigma'}(D_1) = \{r_2(z_2, \beta_1), r_3(\beta_2, z_2)\}$. As for D_2, we immediately have $\text{chase}_{\Sigma}(D_2) \overset{\mathsf{iso}}{\leftrightarrow} \text{chase}_{\Sigma}(D_1)$ and $\text{chase}_{\Sigma'}(D_2) \overset{\mathsf{iso}}{\leftrightarrow} \text{chase}_{\Sigma'}(D_1)$. We immediately notice $\text{chase}_{\Sigma}(D_1) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(D_1)$, $\text{chase}_{\Sigma}(D_2) \overset{\mathsf{hom}}{\rightarrow} \text{chase}_{\Sigma'}(D_2)$, therefore $M \subseteq M'$. With an analogous test we also get $M' \subseteq M$.

4 Conclusions and Open Problems

In this paper we have investigated the problem of containment and equivalence of schema mappings for data exchange. As a first contribution, we have provided a technique for testing a containment $M \subseteq M'$ between two schema mappings, that is based on the existence of a homomorphism between the chase of "dummy" instances. Our technique is applicable in the case where both the source-to-target TGDs and the target TGDs are LAV TGDs, and the chase of all dummy instances under the TGDs in M is finite. With respect to the results of [9], which is a paper closely related to our work, our notion of containment corresponds to the notion of *containment with respect to conjunctive query answering* in [9]. We think this is a natural definition if we are querying the target schema with conjunctive queries, and it can be extended in case of different query languages (see below). In [9] it is also shown that the problem is undecidable for full TGDs on the target schema (full TGDs are TGDs, not necessarily of the LAV kind, that do not have existentially quantified variables in the head). Our opinion is that it is important to consider restricted classes of TGDs on the target schema that are able to capture interesting real-world cases. In particular, the class of LAV TGDs that we have considered generalises inclusion dependencies (see [1]) and also the Datalog$^{\pm}$ family [5]. In particular, the latter is able to represent most ontology formalisms that are used in the Semantic Web. In [5] it is also shown how relevant classes of EGDs, which generalise common database constraints such as key and functional dependencies, can be combined with TGDs so that TGDs and EGDs do not interact, and all the results for TGDs alone apply also in the presence of EGDs; we refer the reader to the paper for the details.

We believe that this preliminary study opens a number of very challenging issues, which will be subject of future research.

(1) The identification of more general conditions under which the problem is decidable. In particular, we intend to remove the condition of finiteness of the chase of dummy instances w.r.t. the left-hand-side TGDs. We conjecture that the containment problem is also decidable when the above condition is removed [10].
(2) The definition of a notion of "minimal" schema mapping, based on the efficient generation of the schema exchange solution. This notion may rely on the notion of *core* solution [11]; however, it will be important to characterise what makes a mapping better than another;
(3) A method for "reducing" a schema mapping into a one that is minimal according to the above definition, and equivalent with respect to conjunctive query answering;
(4) A definition of containment with respect to more expressive classes of queries than conjunctive queries; this would also require a novel definition of certain answers (and possibly of solutions).

Acknowledgements. Andrea Calì was partially supported by the EPSRC project EP/E010865/1 *Schema Mappings and Automated Services for Data Integration and Exchange.*

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. A. V. Aho, Ye. Sagiv, and J. D. Ullman. Equivalences Among Relational Expressions. *SIAM J. Comput.*. 8(2): 218-246, 1979.
3. A. V. Aho, Y. Sagiv, J. D. Ullman. Efficient Optimization of a Class of Relational Expressions. *ACM Trans. Database Syst.*. 4(4): 435-454, 1979.
4. A. Calì and R. Torlone. On the containment of schema mappings. In *SEBD*, pages 255–262, 2008.
5. A. Calì and G. Gottlob and T. Lukasiewicz. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. In *PODS*, 2009, to appear.
6. A. K. Chandra and P. M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *STOC*, pages 77–90, 1977.
7. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and Query Answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
8. R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.
9. R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a theory of schema-mapping optimization In *PODS*, pages 33–42 , 2008.
10. Georg Gottlob and Bruno Marnette. Personal communication. 2009.
11. G. Gottlob and A. Nash. Data Exchange: Computing Cores in Polynomial Time. In *PODS*, pages 40–49, 2006.
12. G. Grahne and A. Mendelzon. Tableau Techniques for Querying Information Sources through Global Schemas. In *ICDT*, pages 332–347, 1999.
13. D.S. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28:167–189, 1984.
14. D. Maier, A. O. Mendelzon, Y. Sagiv. Testing Implications of Data Dependencies. *ACM Trans. Database Syst.*, 4(4): 455–469, 1979.