Data Correspondence, Exchange and Repair

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

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

ABSTRACT

Checking the correspondence between two or more database instances and enforcing it is a procedure widely used in practice without however having been explored from a theoretical perspective. In this paper we formally introduce the *data correspondence* setting and its associated computational problems: checking the existence of solutions, and verifying whether a candidate is a solution, for three distinct types of solutions. Data correspondence is a generalization of *data exchange* and *peer data exchange*, and a special case of *repairing inconsistent databases*. We introduce a new class of dependencies, called *semi-LAV*, that properly includes both *LAV* and *full* dependencies, while retaining tractability for peer data exchange, data correspondence, and database repairs. We also introduce the concept of Σ-*satisfying homomorphisms*. This new type of homomorphism, together with recent advances, is essential in achieving tractability, while at the same time allowing a large class of dependencies, namely the aforementioned semi-LAV ones. We also show the intractability for a series of problems in the case of weakly acyclic tuple generating dependencies. This implies that many tractability results for weakly acyclic dependencies do not carry over to data correspondence; in these new settings we need to restrict the dependencies a bit further, yielding our semi-LAV dependencies.

1. INTRODUCTION

Verifying the coherence between two or more database instances is a procedure widely used in practice, without however having been explored in a systematic fashion in database research. This paper represents a starting point for understanding this important problem, called *Data Correspondence* hereafter. By Data Correspondence we mean the constructive testing between two (or more) database instances in order to verify that they represent the same information. The challenge arises from the fact that the two databases may be structured according to different schemas. The prototypical example is to compare a decomposed, normalized instance to the initial universal instance. In this, and all other cases, the correspondence between the two instances has to be expressed in some formal way, i.e. in some logical formalism. In the decomposition example, the correspondence could for instance be expressed in the familiar algebraic notation by stating that

$$U = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n \text{ and } \pi_{R_i}(U) = R_i,\ i \in \{1, 2, \ldots, n\},$$

where U is the universal schema, and $\{R_1, R_2, \ldots, R_n\}$ is the decomposed one. Given a universal instance u and a decomposed instance $\{r_1, r_2, \ldots, r_n\}$, they correspond if $u = r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$, and $\pi_{R_i}(u) = r_i$, for $i \in \{1, 2, \ldots, n\}$. Sometimes one wants to verify such a correspondence without having the universal instance explicitly. In this case, the question is whether there exists an instance u over U, such that the above equations hold. This is known as testing the instance $\{r_1, r_2, \ldots r_n\}$ for *global consistency* (see e.g. [1]); a problem that has indeed been thoroughly investigated. Global consistency is however only a beginning, the general framework we consider in this paper uses the recently revitalized *tuple generating dependencies* [6] to describe the relationship between two instances and their schemas. We also go beyond global consistency, in that we not only want to verify a correspondence, but in case the instances do not correspond, we are interested in coercing the instances to correspond by modifying one or both of them. In the decomposition example, this could for example entail the deletion of the "dangling tuples" in the r_i relations.

For a real, concrete example from a financial brokerage firm consider employees entering their working hours into a database having schema

EmpHours (*EmpId,ProjId,TotHours*)
HourlyRate (*EmpId,Rate*)
Sponsor (*MgrId, ProjId*)
ExpensePlan (*PlanId, Rate*)

where *EmpHours* has the meaning that the employee with *EmpId* worked at the project *ProjId* for a total of *TotHours*. Relation *HourlyRate* records the hourly salary of the employee in a project. Associated with each project we have a tuple in relation *Sponsor* with the meaning that the project *ProjId* is sponsored from the funds of the manager with *MgrId*. The relation *ExpensePlan* represents the hourly rate used for different expense plans. On the other hand, the Managers have to justify their use of funds by entering data in a different database with schema:

*Contact author.

ICDT 2010, March 22–25, 2010, Lausanne, Switzerland.

Contribution (*MgrId,EmpId,TotHours,Rate*)

with the meaning that the manager with *MgrId* has paid the employee with *EmpId* for *TotHours* hours at the rate of *Rate*. In order to verify that funds have been appropriately dispersed, the company relies on the following correspondence dependencies.

$$\begin{aligned} &\mathit{EmpHours}\,(ei,pi,th), \mathit{Sponsor}\,(mi,pi), \mathit{HourlyRate}\,(ei,r) \\ &\quad\longrightarrow\quad \mathit{Contribution}\,(mi,ei,th,r), \\ &\mathit{Contribution}\,(mi,ei,th,r) \\ &\quad\longrightarrow\quad \exists pl\,:\,\mathit{HourlyRate}\,(ei,r), \mathit{ExpensePlan}\,(pl,r). \end{aligned}$$

If a non-correspondence is detected, the company would bring the two database instances up to date, by entering some missing tuples in the employee database (assuming that the managers financial reports are correct).

Related work. Notwithstanding what we said above, there are some islands of research that are closely related to, or special cases of Data Correspondence, viz. Data Exchange, Peer Data Exchange, and Repairing Inconsistent Databases.

Data Exchange is an important concept harking way back to federated and heterogeneous databases. The problem was finally put on a sound formal basis by the fundamental work of Fagin, Kolaitis, Miller and Popa in [11]. The data exchange problem can be described briefly as follows: given a "source" database schema, a "target" database schema, a database instance over the source schema, and a set of dependencies describing source to target mappings and target constraints, find a database instance over the target schema such that together with the source instance it satisfies the set of dependencies. Besides finding the target instance there are other connected problems, such as finding "good" target instances, finding compact target instances, answering queries over target instances, as well as composing and inverting schema mappings defined by dependencies. In their work Fagin et al. [11] introduced a special class of dependencies (weakly acyclic tuple generating dependencies) for which finding good target instances can be done in polynomial time. The area is currently very active, for instance a major technical breakthrough concerning compacting target instances was recently achieved by Gottlob and Nash in [16]. For an overview of the field, see [5, 18].

Peer Data Exchange is a framework introduced by Fuxman, Kolaitis, Miller and Tan in [14]. It represents a special case of peer data management and a generalization of data exchange. In this framework the source and the target databases have different roles. The source database/peer is considered to be the authoritative one, and the target is considered to have incomplete data. The relationship between the two is expressed using source to target and target dependencies, as in data exchange, but in addition there are also target to source dependencies. The peer data exchange problem is to find a superset of the target instance, such that the two together satisfy the given dependencies. Fuxman et al. study computational problems related to finding these supersets. In addition to intractability results, the authors identify a class of dependencies, albeit one having a rather technical syntactic characterization, for which peer data exchange can be executed efficiently.

Database repairs is one of the main problems associated with inconsistent databases. A database instance is considered inconsistent if it violates the integrity constraints associated with the schema (for a survey of the field, see [7, 9]). A repair of an inconsistent database instance is an instance consistent with the integrity constraints, such that the consistent instance differs in some *minimal* way from the inconsistent one. Several minimality criteria, each presenting their own computational challenges, have been considered. The *symmetric difference repair*, studied by Arenas et al. in [4], requires that the symmetric difference between the inconsistent database and the repair to be minimal, The *subset repairs*, as studied by Chomicki and Marcinkowski in [10], require the repair to be a maximal consistent subset of the inconsistent instance. Lopatenko and Bertossi, in [20], consider *cardinality repairs*, where the repair is to be a subset of maximal cardinality, and Afrati and Kolaitis, in [2], recently introduced the Pareto style *component cardinality repair*. The central computational problems related to database repairs are the *existence of a repair* for a given inconsistent database, and the *repair checking problem*, i.e. test if a given consistent database satisfies the minimality criterion. In their work, Afrati and Kolaitis [2] prove the intractability for the repair checking problem in the general case and also show that the subset repairs and symmetric difference repairs are tractable for weakly acyclic dependencies whose antecedents consist of a single atom. Such dependencies are heavily used in information integration, and are called *Local As View (LAV)*. In addition, Staworko and Chomicki prove in [22] that repair checking for subset and symmetric difference minimality is tractable for *full dependencies*, i.e. dependencies that have no existential quantifiers in the consequent. In this paper we will show that one of our correspondence problems is logspace reducible to database repairs. More importantly, we will provide a large, natural class of dependencies, called *semi-LAV*, that properly includes both the weakly acyclic LAV-dependencies and the full dependencies, and for which all computational problems related to data correspondence are tractable. Thus, as a byproduct, we also provide new, significantly extended tractability results for database repairs.

Summary of results. In this paper we show that the aforementioned problems can be seen in our unified framework of data correspondence. A *data correspondence setting* consists of two non-overlapping schemas, named $\mathbf{R}_1$ and $\mathbf{R}_2$, to emphasize their symmetrical role, together with a set Σ of tuple generating dependencies describing the mapping from $\mathbf{R}_1$ to $\mathbf{R}_2$, and from $\mathbf{R}_2$ to $\mathbf{R}_1$. An instance of the data correspondence problem for a setting consists of two instances, one over $\mathbf{R}_1$, and the other over $\mathbf{R}_2$. Testing whether the instances satisfy Σ can be done in deterministic logarithmic space (in the size of the instances) [1]. If the instances do not correspond, we will be looking for a solution to the discrepancy. Such a solution consists of modified versions of the initial instances. We distinguish between two classes of data correspondence problems: In the *uniform* setting the role of the two instances are symmetric, and as data sources they are considered to be either *sound* but possibly *incomplete*, or *complete* but possibly *unsound*, or neither (cf. [17]). In the first case our instances consist of facts known to be true, but may miss other facts. In this case we look for modifications that are minimal supersets of the initial instances. In the second case our instances contain all true facts, but may additionally contain false facts. Then we are looking for modifications that are maximal subsets of the initial in-

stances. If our information is neither sound, nor complete, the modifications consists of instances that differ minimally from the initial ones, with respect to symmetric difference (that is, with respect to both inserted and deleted facts). The second class of problems are the *non-uniform* versions of the previous. Here one of the data sources, say $\mathbf{R}_1$, is the authoritative one, i.e. it is sound and complete, and can therefore not be modified. Only the instance over $\mathbf{R}_2$ is to be changed, based on superset, subset, or symmetric difference minimality.

We can thus see that peer data exchange corresponds to the non-uniform correspondence setting where we look for modifications that are supersets of the second instance. If we furthermore restrict the second instance to initially be empty and omit the target to source dependencies, we have the classical data exchange setting. The uniform correspondence setting, if generalized so that the schemas $\mathbf{R}_1$ and $\mathbf{R}_2$ are allowed to overlap, or equivalently, if there is only one schema, becomes repairing inconsistent databases.

Most of our results are obtained using homomorphism techniques, in particular some recently developed ones (i.e. [2,11,16]). We introduce the new concept of Σ-*satisfying homomorphisms*, where Σ is a set of dependencies. This type of homomorphism is central in our characterizations and tractability results. Previously, Staworko and Chomicki [22] gave a characterization of when an instance is a repair of another, with respect to a set Σ of full tuple generating dependencies. Using the new notion of Σ-satisfying homomorphisms, we are able to extend the characterization to any set of tuple generating dependencies. We also define a new, large class of dependencies called *semi-LAV*. This class properly includes all LAV-dependencies, and all full dependencies. Modulo a small discrepancy, our semi-LAV dependencies also properly include half of the tractable class discovered in peer data exchange [14] (that technical class actually consists of two distinct classes, and by "half" we mean the first of these). Using our Σ-satisfying homomorphisms together with the previously mentioned homomorphism techniques, we show in this paper that all here considered computational tasks related to data correspondence can be carried out in time polynomial in the input instances, whenever Σ is a set of semi-LAV dependencies. Although not "optimal" in an absolute sense, we show the goodness of the semi-LAV dependencies by deriving intractability results for the slightly larger class of general weakly acyclic dependencies, pertaining to most of the data correspondence problems.

The rest of this paper is organized as follows. Section 2 gives the basic definitions and formally describes the problems we study in this paper. Section 3 contains the characterizations of solutions to the problems, along with some polynomial algorithms for general weakly acyclic tuple generating dependencies. In Section 4 we prove the intractability for all the cases (the bulk of them) not covered by the polynomial time algorithms of Section 3. In Section 5 we introduce the new class of semi-LAV tuple generating dependencies, and give polynomial time algorithms for all data correspondence problems. Conclusions and some prospects for further work follow in Section 6. Missing proofs can be found in the full version.

[1]The original definition of LAV dependencies [19] additionally requires that all variables in the antecedent also appear in the consequent, and that there are no repeated variables in the antecendent.

2. DEFINITIONS

Basics. A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_n\}$ of relational symbols, each R_i having a fixed arity k_i. Let Const be a countably infinite set of constants, and Vars a countably infinite set of variables. An *instance* I of $\mathbf{R}$ is an interpretation that assigns to each relational symbol R_i a finite k_i-ary relation $R_i^I \subset (\mathsf{Const} \cup \mathsf{Vars})^{k_i}$. An instance I over $\mathbf{R}$ is usually identified with the set of tuples $\{R_i^I : R_i \in \mathbf{R}\}$. We denote by $|I|$ the size of I, i.e. the number of tuples in I, and with $dom(I)$ the set of constants and variables that appear in I. An instance I, such that $dom(I) \subseteq \mathsf{Const}$ is called a *ground instance*. If I and J are two instances over the same schema $\mathbf{R}$, we denote by $I \subseteq J$ the fact that $R_i^I \subseteq R_i^J$, for all $i \in \{1, \ldots, n\}$. We write $J \leq_I K$ if $J \oplus I \subseteq K \oplus I$, that is, if $(I \setminus J) \cup (J \setminus I) \subseteq (I \setminus K) \cup (K \setminus I)$. If the inclusion is proper, we write $J <_I K$. Clearly $\leq_I$ is a partial order, for each instance I.

Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be two database schemas with no relational symbols in common. An instance (I_1, I_2) is said to be an instance over $(\mathbf{R}_1, \mathbf{R}_2)$ if I_1 is an instance over $\mathbf{R}_1$ and I_2 an instance over $\mathbf{R}_2$. If (I_1, I_2) and (J_1, J_2) are instances over $(\mathbf{R}_1, \mathbf{R}_2)$, we write $(I_1, I_2) \subseteq (J_1, J_2)$ if $I_1 \subseteq J_1$ and $I_2 \subseteq J_2$, i.e. if $I_1 \cup I_2 \subseteq J_1 \cup J_2$.

Let I and J be instances over a schema $\mathbf{R}$. A *homomorphism* h from I to J is a function on $\mathsf{Vars} \cup \mathsf{Const}$ that is identity on Const, extended to tuples and relations in the natural way, such that $h(R_i^I) \subseteq R_i^J$, for all $i \in \{1, \ldots, n\}$.

The *Gaifman graph (of variables)* G^I for an instance I is an undirected graph with vertex set $dom(I) \cap \mathsf{Vars}$ and an edge between two vertices x and y if x and y appear together in a tuple of I. A *block* is a connected set of variables in G^I. Let $V \subseteq dom(I) \cap \mathsf{Vars}$. We denote with $blocks(V, G^I)$ the set of all *blocks* from the Gaifman graph G^I restricted to the variables in V. Similarly, $blocks(G^I)$ denotes the set of all blocks in G^I (without restriction). If b is a block we denote by $blocksize(b)$ the cardinality of b.

A conjunctive query $\varphi(\bar{x})$ over a schema $\mathbf{R}$ is a conjunctions of relational atoms from $\mathbf{R}$, where $\bar{x}$ denotes the free variables of the atoms of φ. A *tuple generating dependency (tgd)* is a first order sentence of the form

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

where φ (the antecedent) and ψ (the consequent) are conjunctive queries, $\bar{x}$ denotes the universally quantified variables, and $\bar{y}$ the existentially quantified ones. An *equality generating dependency (egd)* of the form $\forall \bar{x}\, \varphi(\bar{x}) \rightarrow x_1 = x_2$, is like a tgd, except that the consequent is an equality between the variables x_1 and x_2 that also are part of $\bar{x}$. For an easier representation we, as is common in the field, omit the universal quantifiers. In this paper we consider tgd's only, egd's will be dealt with in a forthcoming paper. A *full tgd* is a tgd that has no existentially quantified variables. A *LAV-tgd* (Local-As-View tgd[1]) is a tgd that has only one atom in the antecedent. We denote a finite set of tgd's by Σ, and individual tgd's by ξ.

An instance I is said to *satisfy* a set of dependencies, denoted $I \models \Sigma$, if I satisfies Σ in the standard model theoretic sense.

A *position* is a pair (R, i) where R is a relational symbol from schema $\mathbf{R}$ and i is a index between 1 and the arity of R. A variable x is said to *appear* in position (R, i) in a conjunctive query $\varphi(\bar{x})$ if x is in $\bar{x}$ and x appears on the i-th position of R in φ.

For a set Σ of tgd's over a database schema $\mathbf{R}$ the *dependency graph* of Σ is the directed graph that has as vertices all the positions associated with all relational symbols from $\mathbf{R}$. The *dependency graph* has two types of edges:

- *Regular edges.* There is a regular edge between vertices (R,i) and (S,j) if there exists a tgd $\xi \in \Sigma$ that has a variable x that appears both in position (R,i) in the antecedent of ξ, and in position (S,j) in the consequent of ξ.

- *Existential edges.* There is an existential edge between vertices (R,i) and (S,j) if there is a tgd $\xi \in \Sigma$ that has a variable x that appears in position (R,i) of the antecedent, and an existentially quantified variable y that appears in position (S,j) in the consequent.

A set of Σ of tgd's is said to be *weakly acyclic* if the dependency graph of Σ does not have any cycles containing an existential edge [11].

The *chase* of an instance I w.r.t. a set of tgd's Σ is an algebraic proof procedure that repeatedly forces $I \vDash \exists\bar{y}\phi(\bar{a},\bar{y})$ by adding tuples to I, whenever $I \vDash \varphi(\bar{a})$, for some tgd $\varphi(\bar{x}) \rightarrow \exists\bar{y}\ \phi(\bar{x},\bar{y})$ in Σ, and vector $\bar{a}$ of elements from $dom(I)$. For a more detailed description of the chase, see [3, 6, 11, 21]. The chase might not terminate (in the finite), but [11] has shown that if Σ is a finite set of weakly acyclic tgd's it terminates, and does so in time polynomial in $|I|$. We denote the (finite or infinite) result by $chase_\Sigma(I)$. The crucial property, from our point of view, is that $chase_\Sigma(I) \vDash \Sigma$.

Data Correspondence. Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be two schemas with no relation symbols in common. A *(data) correspondence mapping* for $(\mathbf{R}_1,\mathbf{R}_2)$ is either a tgd of the form

$$\phi_1(\bar{x}) \rightarrow \exists\bar{y}\phi_2(\bar{x},\bar{y}),$$

or a tgd of the form

$$\phi_2(\bar{x}) \rightarrow \exists\bar{y}\phi_1(\bar{x},\bar{y}),$$

where all the atoms of ϕ_1 are over $\mathbf{R}_1$, and the atoms of ϕ_2 are over $\mathbf{R}_2$. We consider finite sets $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ of such correspondence mappings, where Σ_{12} contains all the tgd's of the first form, and Σ_{21} the tgd's of the second form. Note that tgd's Σ_{12} are used to specify data exchange mappings [11], where the source schema is $\mathbf{R}_1$, and the target schema is $\mathbf{R}_2$.

A *data correspondence setting* is as a triple $(\mathbf{R}_1,\mathbf{R}_2,\Sigma)$, where $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ is a set of correspondence mappings for $(\mathbf{R}_1,\mathbf{R}_2)$ as above. We say that a ground instance (I_1,I_2) of $(\mathbf{R}_1,\mathbf{R}_2)$ *satisfies* the setting if $(I_1,I_2) \vDash \Sigma$.

Let $(\mathbf{R}_1,\mathbf{R}_2,\Sigma)$ be a correspondence setting, and (I_1,I_2) a ground instance over $(\mathbf{R}_1,\mathbf{R}_2)$. Then $\langle (I_1,I_2),(\mathbf{R}_1,\mathbf{R}_2,\Sigma)\rangle$, denotes an *instance* of the *uniform data correspondence problem*. For such an instance, usually denoted simply (I_1,I_2,Σ), there are three types of solutions, namely: subset solutions, superset solutions, and $\oplus$-solutions.

Definition 1. *Let (I_1,I_2,Σ) be an instance of the uniform data correspondence problem, and (K_1,K_2) a non-empty ground instance over $(\mathbf{R}_1,\mathbf{R}_2)$. If (K_1,K_2) satisfies Σ, we say that (K_1,K_2) is a*

- *subset solution if $(K_1,K_2) \subseteq (I_1,I_2)$ and (K_1,K_2) is maximal among the subsets of (I_1,I_2) satisfying Σ.*

- *superset solution if $(K_1,K_2) \supseteq (I_1,I_2)$ and (K_1,K_2) is minimal among the supersets of (I_1,I_2) satisfying Σ.*

- *$\oplus$-solution if (K_1,K_2) is $\leq_{(I_1,I_2)}$-minimal among the instances satisfying Σ.* ∎

Note that in all cases an instance can have one, or several solutions. Subset and $\oplus$-solutions might not exist, whereas superset solutions always do.

The *non-uniform data correspondence problem* is similar to the uniform one, except that the instance I_1 is kept fixed when looking for solutions. This has, among other things, the consequence that not even superset solutions are guaranteed to exist.

Definition 2. *Let $(I_1,I_2,\Sigma_{12},\Sigma_{21})$ be an instance of the non-uniform data correspondence problem, and K_2 a non-empty ground instance over $\mathbf{R}_2$. If (I_1,K_2) satisfies Σ, we say that K_2 is a*

- *subset solution if $K_2 \subseteq I_2$ and K_2 is maximal among the subsets of I_2, such that (I_1,K_2) satisfies Σ.*

- *superset solution if $K_2 \supseteq I_2$ and K_2 is minimal among the supersets of I_2, such that (I_1,K_2) satisfies Σ.*

- *$\oplus$-solution if K_2 is $\leq_{I_2}$-minimal among the instances over $\mathbf{R}_2$, such that (I_1,K_2) satisfies Σ.* ∎

These above definitions give rise to classes of decision problems that we study in this paper. The first class is the existence of solutions to the correspondence problem, and the second class is to check whether a given instance is a solution. We note that Fuxman, Kolaitis, Miller and Tan [14] have investigated, under the name of *Peer Data Exchange*, the existence of superset solutions to the non-uniform correspondence problem. However, they do not require a solution to be minimal, which in fact leaves out an additional computational hurdle similar to computing the core of a solution. We shall return to this issue in Section 4.

In addition, it turns out that our correspondence problem can be seen as a special case of the problem of *repairing inconsistent databases*, in particular the problem of *repair checking*, which recently has been illuminated by Afrati and Kolaitis [2]. The definition is as follows:

Definition 3. ([2]) *Let I be a ground instance over a schema $\mathbf{R}$, and Σ a set of constraints. Then a ground instance K over $\mathbf{R}$ that satisfies Σ is a*

- *subset repair of I w.r.t. Σ, if $K \subseteq I$ and K is maximal among the subsets of I that satisfies Σ.*

- *superset repair of I w.r.t. Σ, if $K \supseteq I$ and K is minimal among the supersets of I that satisfies Σ.*

- *$\oplus$-repair of I w.r.t. Σ, if K is $\leq_I$-minimal among the instances satisfying Σ.* ∎

Afrati and Kolaitis actually did not consider the second problem, but they did in addition also investigate cardinality based repairs (i.e. a subset of maximal cardinality satisfying the dependencies). Cardinality based repairs however turned out to be intractable even for the simplest kind of dependencies, so they are not included in the present paper.

We now formally define the decision problems studied in this paper.

EXISTENCE-OF-SOLUTION($\Sigma, *$, uniform/non-uniform)

Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$. The input is an instance (I_1, I_2), and the question is whether the uniform/non-uniform correspondence problem $(I_1, I_2, \Sigma_{12}, \Sigma_{21})$ has a $*$-solution, where $*$ is one of subset/superset/$\oplus$.

SOLUTION-CHECKING($\Sigma, *$, uniform/non-uniform)

Input: instances (I_1, I_2) and (K_1, K_2). Question: is the instance (K_1, K_2) a $*$-solution to the uniform/non-uniform correspondence problem $(I_1, I_2, \Sigma_{12}, \Sigma_{21})$.

EXISTENCE-OF-REPAIR($\Sigma, *$)

The input is an instance I, and the question is whether I has a $*$-repair w.r.t. Σ.

REPAIR-CHECKING($\Sigma, *$)

The input is instances I and K, and the question is whether the instance K is a $*$-repair of I w.r.t. Σ.

3. BASIC CHARACTERIZATIONS

Our first contribution is a general characterization of repairs, applicable to any set of tgd's. For this we need the following lemma and definition.

LEMMA 1. ([13]) *Let I and J be instances, and Σ a set of tgd's over a schema $\mathbf{R}$. If $I \subseteq J$ there exists a homomorphism h, such that $h(chase_\Sigma(I)) \subseteq chase_\Sigma(J)$.* ∎

Before we introduce the next concept let us consider the following set of dependencies between $\mathbf{R}_1$ and $\mathbf{R}_2$ in a uniform correspondence setting (cf. [12]):

$$\begin{aligned}\Sigma_{12} &= \{Emp\,(e) \rightarrow \exists m\; EmpMgr\,(e, m)\},\\ \Sigma_{21} &= \{EmpMgr\,(e, m) \rightarrow Manager;(m),\\ &\qquad EmpMgr(e, e) \rightarrow SelfMgr\,(e, e)\}.\end{aligned}$$

Consider the instance I, where $Emp^I = \{(ron),\ (phokion)\}$, and $EmpMgr^I = Manager^I = SelfMgr^I = \varnothing$. Let the dependencies be $\Sigma = \Sigma_{12} \cup \Sigma_{21}$, and denote $chase_\Sigma(I)$ by K. Then $Emp^K = \{(ron),\ (phokion)\}$, $Manager^K = \{(x),\ (y)\}$, $EmpMgr^K = \{(ron, x),\ (phokion, y)\}$, and $SelfMgr^K = \varnothing$. Then take instance J, with $Emp^J = \{(ron),\ (phokion)\}$, $Manager^J = \{(ron),\ (thomas)\}$, $SelfMgr^J = \varnothing$, $EmpMgr^J = \{(ron, ron),\ (phokion, ron)\}$. It is easy to see that the homomorphism h that maps both null values x and y to *ron* is a homomorphism from $K = chase_\Sigma(I)$ into J. On the other hand, neither $h(K)$ nor J satisfies Σ. These observations lead to the introduction of a new type of homomorphism, namely Σ-satisfying homomorphisms.

DEFINITION 4. *Given an instance K and a set Σ of tgd's over a schema $\mathbf{R}$, as well as a ground instance I over $\mathbf{R}$, a Σ-satisfying homomorphism from K into I is a homomorphism h from K into I, such that $h(K) \subseteq J \subseteq I$, and $J \models \Sigma$, for some $J \subseteq I$.* ∎

We can now state necessary and sufficient conditions for when an instance K, such that $K \models \Sigma$, is a $\oplus$-repair for a given instance I w.r.t. the set Σ of tgd's. The first condition assures that $K \setminus I$ does not contain superfluous tuples, and the second condition guarantees that no more tuples from I could be added to K.

THEOREM 1. *Let Σ be a set of tgd's, I a ground instance, and K a ground instance such that $K \models \Sigma$. Then K is a $\oplus$-repair if and only if the following conditions are satisfied:*

1. *For all Σ-satisfying homomorphisms h, such that $h(chase_\Sigma(I \cap K)) \subseteq K$, and all instances $J \subseteq K$, such that $J \models \Sigma$ and $h(chase_\Sigma(I \cap K)) \subseteq J$, it holds that $J = K$.*

2. *There does not exist a tuple $t \in I \setminus K$, and Σ-satisfying homomorphism h with an instance $J \subseteq K \cup I$, $J \models \Sigma$, such that $h(chase_\Sigma((I \cap K) \cup \{t\})) \subseteq J \subseteq K \cup I$.*

Proof: Suppose K is a $\oplus$-repair. As $K \models \Sigma$, we have $chase_\Sigma(K) = K$. By Lemma 1 there then exists a homomorphism h such that $h(chase_\Sigma(I \cap K)) \subseteq K$. Towards a contradiction, suppose that there is an instance J, such that $h(chase_\Sigma(I \cap K)) \subseteq J \subset K$ and $J \models \Sigma$. Then $J \setminus I \subset K \setminus I$, and since $I \cap K \subseteq chase_\Sigma(I \cap K)$ and $I \cap K$ is ground, we have that $I \cap K \subseteq J$, meaning that $I \setminus J \subseteq I \setminus K$, and consequently that $J <_I K$. But this contradicts the assumption that K is a $\oplus$-repair. Therefore it must be that $J = K$, meaning that K satisfies condition 1.

Suppose then that condition 2 is violated, and tuple t, Σ-satisfying homomorphism h, and instance J exist. Since for such a J, we would have $I \setminus J \subset I \setminus K$, it would follow that $J <_I K$, contradicting the assumption that K is a $\oplus$-repair.

For the only if direction, let K be an instance such that $K \models \Sigma$, and K satisfies conditions 1 and 2. We need to show that K is $\oplus$-minimal. If this is not the case, there must be an instance J, such that $J \models \Sigma$ and $J <_I K$. Thus $J \setminus I \subseteq K \setminus I$, and $I \setminus J \subseteq I \setminus K$, and at least one of the inclusions is proper. Suppose first that it were the case that $I \setminus J = I \setminus K$. Then we would have $J \subseteq K$. Thus, if $J \setminus I$ were to be a proper subset of $K \setminus I$, it would necessarily be that $J \subset K$. Since then $I \cap K = I \cap J$, we will also have $chase_\Sigma(I \cap K) = chase_\Sigma(I \cap J)$. By Lemma 1 we have a homomorphism h, such that $h(chase_\Sigma(I \cap J)) \subseteq chase_\Sigma(J) = J$. Then it would hold that $h(chase_\Sigma(I \cap K)) \subseteq J \subset K$, contradicting condition 1.

If it were the case that $I \setminus J \subset I \setminus K$, we would find a tuple $t \in (J \cap I) \setminus (K \cap I)$, that is in $I \setminus K$, such that by Lemma 1 we would have a homomorphism h, such that $h(chase_\Sigma((I \cap K) \cup \{t\})) \subseteq J \subseteq K \cup I$, a contradiction to the assumption that K satisfies condition 2. ∎

As we mentioned before, a correspondence solution is a special case of a repair, so the above characterization can also be used for the correspondence problem, by the following lemma.

LEMMA 2. *Let $(I_1, I_2, \Sigma_1, \Sigma_2)$ be an instance of the uniform data correspondence problem, and (K_1, K_2) a ground instance, such that $(K_1, K_2) \models \Sigma_{12} \cup \Sigma_{21}$. Then (K_1, K_2) is a subset (superset, $\oplus$) solution to the uniform correspondence problem if and only if (K_1, K_2) is a subset (superset, $\oplus$, respectively) repair of (I_1, I_2) w.r.t. $\Sigma_{12} \cup \Sigma_{21}$.* ∎

In case of solutions to the non-uniform correspondence problem, we have a characterization similar to Theorem 1.

THEOREM 2. *Let $(I_1, I_2, \Sigma_{12}, \Sigma_{21})$, with $\Sigma = \Sigma_{12} \cup \Sigma_{21}$, be an instance of the non-uniform correspondence problem, and K_2 an instance such that $(I_1, K_2) \models \Sigma$. Then K_2 is a $\oplus$-solution for $(I_1, I_2, \Sigma_{12}, \Sigma_{21})$ if and only if the following conditions are satisfied:*

1. *For all Σ-satisfying homomorphism h, such that we have $h(chase_\Sigma(I_1, I_2 \cap K_2)) \subseteq (I_1, K_2)$, and all instances $J_2 \subseteq K_2$ such that we have $(I_1, J_2) \models \Sigma$ and $h(chase_\Sigma(I_1, I_2 \cap K_2)) \subseteq (I_1, J_2)$, it holds that $J_2 = K_2$.*
2. *There does not exist a tuple $t \in I_2 \setminus K_2$ and a Σ-satisfying homomorphism h with a "witnessing" instance $J_2 \subseteq (I_2 \cup K_2)$, such that we have $(I_1, J_2) \models \Sigma$, and $h(chase_\Sigma(I_1, I_2 \cap K_2 \cup \{t\})) \subseteq (I_1, J_2)$.*

Proof: Follows the lines of the proof of Theorem 1. ∎

The next theorem is due to Fagin, Kolaitis and Popa [11], and was further highlighted by Gottlob and Nash [16]. Note that any set Σ_{12} of tgd's is non-recursive, and hence weakly acyclic.

THEOREM 3. ([11])

1. *Let I be a ground instance, and Σ a weakly acyclic set of tgd's. Then there is a polynomial p, such that $|chase_\Sigma(I)| \leq p(|I|)$.*
2. *Let (I_1, I_2) be a ground instance. Then $chase_{\Sigma_{12}}(I_1, I_2)$ has bounded blocksize.*
3. *Let I and K be instances such that $blocksize(G^I) \leq c$, for some constant c. Then it can be tested whether there exists a homomorphism from I to K in time $O(|K|^c)$.* ∎

We will also use the following result, obtained by Afrati and Kolaitis.

THEOREM 4. ([2]) *Let Σ be a set of weakly acyclic tgd's and I an instance, such that $I \models \Sigma$. Then there exists a constant c, depending only on Σ, such that for each tuple $t \in I$ there is an instance $K_t \subseteq I$, such that $t \in K_t, K_t \models \Sigma$, and $|K_t| \leq c$.* ∎

Armed with these results we will now derive polynomial time algorithms that determine the existence of database repairs and of uniform correspondence solutions for weakly acyclic tgd's.

THEOREM 5. *Let Σ be a set of weakly acyclic tgd's. Then the problem*

EXISTENCE-OF-REPAIR(Σ, subset)

can be solved in polynomial time.

Proof: Let I, Σ be an instance of the problem, and c the constant mentioned in Theorem 4. Consider the following algorithm:

REPAIR-SEARCH(I, Σ, c)

```
1  for i ← 1 to c
2        do for K ⊆ I, |K| = i
3                 do if K ⊨ Σ return TRUE
4  return FALSE
```

The algorithm is obviously sound. The completeness follows directly from Theorem 4. The inner loop on line 2 is executed at most $\binom{|I|}{i}$ times, and the outer loop on line 1 at most c times. Thus line 3 is executed at most $|I|^c$ times. Each execution of line 3 is done deterministically in space logarithmic in at most c. Consequently, the algorithm runs in time polynomial in the size of I. ∎

Using Lemma 2 we obtain

COROLLARY 1. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of weakly acyclic tgd's. Then the problem*

EXISTENCE-OF-SOLUTION$(\Sigma, \text{subset, uniform})$

can be solved in polynomial time. ∎

We note that EXISTENCE-OF-REPAIR$(\Sigma, \text{superset})$ as well as EXISTENCE-OF-SOLUTION$(\Sigma, \text{superset, uniform})$ are non-problems, since superset solutions always exist (recall that no egd's are present). We now turn our attention to the solution checking problem for data correspondence. It the next section we shall see that the uniform version of this problem is coNP-complete for weakly acyclic tgd's and all types of solutions. However, using homomorphism techniques we are able show that the non-uniform version of the problem is polynomial for subset and superset solutions for any set of tgd's. Somewhat surprisingly then, as shown in the next section, it turns out that the non-uniform version is coNP-hard for $\oplus$-solutions even for weakly acyclic tgd's. Before that, we show the polynomial results.

THEOREM 6. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of tgd's. Then the problem*

SOLUTION-CHECKING$(\Sigma, \text{subset, non-uniform})$

can be solved in polynomial time.

Proof: Let (I_1, I_2) be an instance, Then instance $K_2 \subseteq I_2$ is a subset solution for $(I_1, I_2, \Sigma_{12}, \Sigma_{21})$, if $(I_1, K_2) \models \Sigma$, and there does not exist an instance J_2, such that $K_2 \subset J_2 \subseteq I_2$ and $(I_1, J_2) \models \Sigma$. The test $(I_1, K_2) \models \Sigma$ can be done deterministically in space logarithmic in the size of (I_1, K_2). It is easy to see that a J_2, as mentioned above, does exist if and only if there is a tuple $t \in I_2 \setminus K_2$ and a homomorphism h, such that $h(chase_{\Sigma_{21}}(K_2 \cup \{t\})) \subseteq I_1$. It follows from Theorem 3 that the existence of such a homomorphism h can be determined in time polynomial in the size of I_1. The chase with Σ_{21} and the homomorphism test is repeated for each tuple in $I_2 \setminus K_2$. The whole process thus runs in time polynomial in the size of (I_1, I_2). ∎

THEOREM 7. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of tgd's. Then the problem*

SOLUTION-CHECKING$(\Sigma, \text{superset, non-uniform})$

can be solved in polynomial time.

Proof: Let the instance be (I_1, I_2) and K_2 the candidate solution. First test if $(I_1, K_2) \models \Sigma$. If so, look for a tuple $t \in K_2 \setminus I_2$, and a homomorphism h from $chase_{\Sigma_{12}}(I_1)$ to

$K_2 \setminus \{t\}$. By Theorem 3 this process can be carried out in time polynomial in the size of (I_1, K_2). Evidently, if such a tuple t and homomorphism h is found, K_2 is not a superset solution, since then $(I_1, K_2 \setminus \{t\}) \models \Sigma$, meaning that K_2 is not minimal. If no tuple t and homomorphism h is found, K_2 is obviously minimal. ∎

As we will see in the next section the remaining problems are intractable for weakly acyclic tgd's.

4. HARD CASES

We just saw that for the non-uniform correspondence problem, checking whether an instance is a subset solution or a superset solution is polynomial. If we allow a solution to be in part a subset, and in part a superset, the problem becomes coNP-complete.

THEOREM 8. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of tgd's. Then the problem*

SOLUTION-CHECKING$(\Sigma, \oplus, \text{non-uniform})$

is in coNP, and is coNP-hard even for weakly acyclic tgd's.

Proof: Let the instance be (I_1, I_2) and K_2 the candidate solution. To see that the problem is in coNP, consider the complementary problem, this is, does there exist an instance J_2, such that $(I_1, J_2) \models \Sigma$, and $J_2 <_{I_2} K_2$? This problem is clearly in NP, since we can guess J_2, and then check in logarithmic space whether $(I_2, J_2) \models \Sigma$.

For the lower bound we will, in the spirit of [2], reduce the POSITIVE 1-IN-3-SAT problem to the $\oplus$-repair checking problem. The POSITIVE 1-IN-3-SAT problem asks whether a set of disjunctive clauses, each having three positive variables, is satisfiable with a truth assignment that makes exactly one variable in each clause true. This problem is known to be NP-complete [15].

We consider the schema $(\mathbf{R}_1, \mathbf{R}_2)$, where $\mathbf{R}_1 = \{P, E, V, F\}$, and $\mathbf{R}_2 = \{T, S, D\}$. Let Σ_{12} consist of the following dependencies

$$P(x,y,z) \rightarrow \exists u,v,w : T(x,u), T(y,v)\ T(z,w), S(u,v,w) \quad (1)$$

$$F(u,v,w,u',v',w') \rightarrow D(u,v,w,u',v',w') \quad (2)$$

and Σ_{21} consist of

$$T(x,u), T(x,u') \rightarrow E(u,u') \quad (3)$$

$$S(u,v,w), S(u',v',w'), D(u,v,w,u',v',w') \rightarrow V(u,v,w). \quad (4)$$

Note that $\Sigma_{12} \cup \Sigma_{21}$ is a weakly acyclic set.

Given an instance $\mathscr{P}$ of the POSITIVE 1-IN-3-SAT problem we construct I_1 as

$$\begin{aligned} P^{I_1} &= \{(x,y,z) : x \vee y \vee z \text{ is a clause in } \mathscr{P}\} \\ E^{I_1} &= \{(0,1),\ (1,0)\} \\ V^{I_1} &= \{(0,0,1),\ (0,1,0)\ (1,0,0)\} \\ F^{I_1} &= \{(u,v,w)\ (u',v',w') : u,v,w,u',v',w' \in \{1,0\}, \\ &\qquad (u,v,w) \neq (u',v'.w')\} \end{aligned}$$

and I_2 as

$$\begin{aligned} T^{I_2} &= \{(x,0) : x \text{ variable in } \mathscr{P}\} \\ S^{I_2} &= \{(0,0,1),\ (0,1,0),\ (1,0,0)\} \\ D^{I_2} &= F^{I_1}. \end{aligned}$$

Finally let K_2 be

$$\begin{aligned} T^{K_2} &= \{(x,1) : x \text{ is a variable in } \mathscr{P}\} \\ S^{K_2} &= \{(1,1,1)\} \\ D^{K_2} &= D^{I_2}. \end{aligned}$$

It is straightforward to verify that $(I_1, K_2) \models \Sigma$. The reduction clearly being polynomial, it remains only to prove that there exists a truth assignment for $\mathscr{P}$ making exactly one variable in each clause true if and only if K_2 is not a $\oplus$-repair.

Suppose that there exists such a POSITIVE 1-IN-3 truth assignment making $\mathscr{P}$ true. Let this truth assignment be represented by a mapping ν from the variables of $\mathscr{P}$ to $\{0,1\}$. Consider then an instance J_2 with

$$\begin{aligned} T^{J_2} &= \{(x,\nu(x)) : x \text{ is a variable in } \mathscr{P}\} \\ S^{J_2} &= \{(0,0,1),\ (0,1,0),\ (1,0,0)\} \\ D^{J_2} &= D^{K_2}. \end{aligned}$$

It is easy to verify that $(I_1, J_2) \models \Sigma$, and that $J_2 <_{I_2} K_2$. Thus K_2 is not a $\oplus$-repair.

For the other direction, suppose that there does not exist a POSITIVE 1-IN-3 truth assignment making $\mathscr{P}$ true. Let us try to construct an instance J_2, such that $(I_1, J_2) \models \Sigma$ and $J_2 <_{I_2} K_2$, by manipulating K_2 to become $\oplus$-closer to I_2.

First we note that dependencies (1) and (3) force the interpretation of T to contain exactly a truth assignment for each variable, and that dependency (2) blocks us from deleting tuples from D^{K_2}.

We could make K_2 closer to I_2 by replacing some tuples $(x,1)$ in T^{K_2} with $(x,0)$. But then, in order to satisfy dependency (1), we would have to replace the tuple $(1,1,1) \in S^{K_2}$ with at least one tuple containing at least one '0'. Suppose first, that there were only one new tuple, and this tuple would contain only one '0'. This would make the resulting instance $\leq_{I_2}$-incomparable with K_2. To avoid this, we could leave tuple $(1,1,1)$ in the interpretation of S. But then the interpretation of S would contain at least two tuples. So, in any case, dependency (4) will be triggered, forcing the interpretation of S to contain only tuples from V^{I_1}. We would then have constructed a POSITIVE 1-IN-3 truth assignment for $\mathscr{P}$ in the interpretation of T. We have now exhausted all possibilities to improve the solution K_2, and can therefore conclude that K_2 indeed is a $\oplus$-solution. ∎

Afrati and Kolaitis derived the following complexity theoretic characterization for repair checking with weakly acyclic tgd's.

THEOREM 9. ([2]) *There is a weakly acyclic set Σ of tgd's such that the*

REPAIR-CHECKING$(\Sigma, *)$

problem is coNP-complete, both for $*$ being subset and $\oplus$ ∎

To this we can now add the missing piece.

THEOREM 10. *There is a weakly acyclic set Σ of tgd's such that the*

$$\boxed{\text{REPAIR-CHECKING}(\Sigma, \text{superset})}$$

problem is coNP-complete. ∎

The next theorem shows that the result of Afrati and Kolaitis [2] can actually be sharpened to also hold for subset solution checking for the correspondence problem.

THEOREM 11. *There are weakly acyclic sets $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ of tgd's such that the problem*

$$\boxed{\text{SOLUTION-CHECKING}(\Sigma, *, \text{uniform})}$$

is coNP-complete, for $*$ being any of subset/superset/$\oplus$. ∎

Let us now consider the existence of solutions for the non-uniform correspondence settings. For this, as will be proved, all the three cases *subset/superset/*$\oplus$ are intractable for general weakly acyclic set of tuple generating dependencies.

THEOREM 12. *There are weakly acyclic sets $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ of tgd's such that the problem*

$$\boxed{\text{EXISTENCE-OF-SOLUTION}(\Sigma, *, \text{non-uniform})}$$

is NP-complete, for $*$ being any of subset/superset/$\oplus$. ∎

5. PTIME FOR SEMI-LAV TGD'S

In order to overcome the intractability barriers from the previous section, it is clear that the set Σ of dependencies has to be restricted. We next introduce a large class of tgd's for which all of the considered problems can be solved in polynomial time. The proofs rely on recently developed homomorphism techniques (i.e. [2, 11, 16]), and on the Σ-satisfying homomorphisms introduced in Section 3.

First, we introduce the new class of tgd's. For this, we need the concept of the rank of a node in the dependency graph of a set Σ of tgd's.

DEFINITION 5. ([11]) *Let (R, i) be a vertex in the dependency graph of a set Σ of tgd's. Then $rank(R, i)$ is the maximum number of existential edges along any path in the graph ending in (R, i).* ∎

LEMMA 3. ([11]) *If Σ is a weakly acyclic set of tgd's, then for each node (R, i) of the dependency graph of Σ, $rank(R, i)$ is finite and is bounded by some constant c depending only on Σ.* ∎

DEFINITION 6. *Let Σ be a set of tgd's over a schema $\mathbf{R}$. For all relational symbols $R \in \mathbf{R}$ that occur in Σ, we say that a position (R, i) is* unsafe *if $rank(R, i) > 0$. Any relational symbol R that contains an unsafe position is said to be unsafe. A set Σ of weakly acyclic dependencies is said to be* semi-LAV *if all unsafe relational symbols occur in the antecedents of LAV-dependencies only.* ∎

For an example, consider the following set of semi-LAV dependencies that is neither LAV nor full. The application is a police database concerning traffic contraventions. The schema is

Driver(*LicenceId, Name*)
Contravention(*LicenceId, Municipality, Date, Type*)
Ticket(*LicenceId, Date, Penalty, DemeritPoints*)
DriverHist(*LicenceId, Type*)
PenaltyUpgrade(*Type1, Type2, Type3*)
Notification(*InsuranceId, LicenceId, DemeritPoints*)

The relation *Driver* contains information about drivers that have been convicted of traffic offenses, and relation *Contravention* stores data regarding the contraventions. The attribute *Type* can take values 'minor', 'medium', and 'major.' The relation *Ticket* contains the ticket information, and *DriverHist* contains information regarding the history of the driver's contraventions. The relation *PenaltyUpgrade* represents a constraint between different contraventions, for example if for some driver we have both a 'minor' and a 'medium' contravention it implies that the driver is also assigned a 'major' contravention, since she is a repeat offender. Finally, the relation *Notification* represents the notifications sent to the insurance companies for fee increases. The set Σ of dependencies is

$$\begin{aligned} &Driver(l, n), Contravention(l, m, d, t) \\ &\qquad\qquad \rightarrow \quad \exists p, dm \ : \ Ticket(l, d, p, dm), \\ &\qquad\qquad\qquad\qquad\quad DriverHist(l, t) \end{aligned}$$

$$\begin{aligned} &DriverHist(l, \text{minor}), \\ &DriverHist(l, \text{medium}), \\ &PenaltyUpgrade(\text{minor,medium,major}) \\ &\qquad\qquad \rightarrow \quad DriverHist(l, \text{major}) \end{aligned}$$

$$Ticket(l, d, p, dm) \quad \rightarrow \quad \exists i \ : \ Notification(i, l, dm)$$

In this set we have $rank(Ticket, 3) = rank(Ticket, 4) = 1$, and $rank(Notification, 1) = 2$. The rest of the positions have a rank of 0. Therefore *Ticket* and *Notification* are the only unsafe relation symbols, and since *Ticket* appears in the antecedent only of a LAV dependency and *Notification* does not appear in any antecedent it follows that Σ is a set of semi-LAV tgd's.

The class of semi-LAV tgd's possesses several useful properties, which we show next.

LEMMA 4. *Let Σ be a set of semi-LAV tgd's, let I be a ground instance, and K be the result of an arbitrary chase sequence using Σ starting from I. Then for any variable x that occurs in K, the number of tuples in K that contain x is bounded by a constant that depends only on Σ.*

Proof: Let J be the first instance in the chase sequence from I to K where the variable x appears, and denote with J_x the subset of tuples of J where x appears. If r is the maximum number of atoms in the consequent of any dependency in Σ, it is clear that $|J_x| \leq r$. Furthermore, as x can appear only in tuples of unsafe relations (called *unsafe tuples*), it means that the tuples in J_x can only be matched with the antecedents of some LAV-dependencies, each such antecedent consisting of a single atom. Consequently, the set of tuples K_x in K that contain x can be generated by chasing J_x using only the LAV-dependencies in Σ. As Σ is weakly acyclic, it follows from Theorem 3 that there is a polynomial p such that $|K_x| \leq p(|J_x|)$. Thus $|K_x| \leq p(r)$, which is a constant depending only on Σ. ∎

LEMMA 5. *Let Σ be a set of weakly acyclic tgd's, I a ground instance, and K the result of some arbitrary chase sequence using Σ starting from I. Then there exists a partitioning $\{V_0, V_1, \ldots, V_m\}$ of the variables in K such that variable $x \in V_i$ iff x was generated during the chase process using only variables from $\{V_0, \ldots, V_{i-1}\}$.*

Proof: Let $N_0, N_1, \ldots, N_m$ be a partition of the nodes in the dependency graph of Σ based on their rank. Since Σ is weakly acyclic, such a partitioning exists, and for any dependency in Σ the rank of the positions in the consequent is larger than or equal to the rank of any position in the antecedent [11]. We show the existence of the variable partitioning constructively based on the chase steps: Initially let V_i to be the empty set, for all $i \in \{0, 1, \ldots, m\}$. During the chase process when a new variable x is created in a position (R, i) we add that variable to set V_1 if there were no variables in the tuples that triggered the dependency. We add the variable x to the set V_i if the set of variables belonging to the tuples that triggered the dependency that generated x is a subset of $\{V_0 \cup V_1 \cup \ldots \cup V_{i-1}\}$ and there exists a variable y in that set such that $y \in V_{i-1}$. Because the rank of each position is at most m, it means that variable x cannot be generated by variables belonging to V_m or higher. This proves the existence of partitioning $\{V_0, V_1, \ldots, V_m\}$ of the variables. Note that $V_0 = \varnothing$. ∎

THEOREM 13. *Let Σ be a set of semi-LAV tgd's, let I be a ground instance, and K be the result of some arbitrary chase sequence using Σ starting from I. Then there is a constant c, depending only on Σ, such that $blocksize(G^K) \leq c$.*

Proof: Let $\{V_0, V_1, \ldots, V_m\}$ be the partitioning of the variables in K, as described in Lemma 5. We prove by an induction on i that there are constants c_i, such that for any block $b \in blocks(V_0 \cup \cdots \cup V_i, G^K)$, we have $blocksize(b) \leq c_i$, meaning that $blocksize(G^K) = max\{blocksize(b) : b \in blocks(V_0 \cup \cdots \cup V_m, G^K)\} \leq c_m$.

Basis: Clearly $V_0 = \varnothing$, and $blocks(V_0, G^K) = \varnothing$. We can therefore set $c_0 = 0$.

Inductive step: Let b be a block in $blocks(V_0 \cup \cdots \cup V_i, G^K)$. If all the variables in b are in $V_0 \cup \cdots \cup V_{i-1}$, the block b is also in $blocks(V_0 \cup \cdots \cup V_{i-1}, G^K)$ and by the inductive hypothesis we have $blocksize(b) \leq c_{i-1}$. Suppose then that x is a variable in b from V_i. This means that x was generated from variables in $V_0 \cup \cdots \cup V_{i-1}$ by an existential dependency ξ. There are three cases to consider:

Case 1: ξ is a dependency with possibly several atoms in the antecedent. Then all these atom are over safe relations, not containing variables. Thus x, along with the other existential variables in the consequent of ξ will form their own block in $blocks(V_0 \cup \cdots \cup V_i, G^K)$, with blocksize at most s, where s is the maximum number of special edges incident from any node in the dependency graph of Σ.

Case 2: ξ has a single unsafe atom in the antecedent, but none of the (universally quantified) variables in the antecedent occur in the consequent. Then ξ does not propagate any variables from the antecedent to the consequent, and, as in case 1, the variable x will belong to a block in $blocks(V_0 \cup \cdots \cup V_i, G^K)$, with blocksize at most s.

Case 3: ξ has a single unsafe atom in the antecedent. Then ξ was triggered by a single tuple containing only variables in $V_0 \cup \cdots \cup V_{i-1}$. Thus each of these variables belong to a block $b' \in blocks(V_0 \cup \cdots \cup V_{i-1}, G^K)$, and by the inductive hypothesis $blocksize(b') \leq c_{i-1}$. Therefore b' consists of at most c_{i-1} variables, say $Y = \{y_1, \ldots, y_{c_{i-1}}\}$. Let K_Y denote the subset of tuples of K, that contain variables in Y, and K_{y_ℓ} the tuples that contain variable y_ℓ. By Lemma 4 there are constants $d_1, \ldots, d_{c_{i-1}}$, such that $|K_{y_\ell}| \leq d_\ell$, for $\ell \in \{1, \ldots, c_{i-1}\}$. Consequently $|K_Y| \leq \sum_{\ell=1}^{c_{i-1}} d_\ell$. Each tuple in K_Y can generate at most $D \cdot s$ new variables x in V_i, where D is the number of dependencies in Σ. Each such generated variable x increases the blocksize of b' by at most one. Consequently $blocksize(b) \leq blocksize(b') + |K_Y| \cdot D \cdot s \leq c_{i-1} + \sum_{\ell=1}^{c_{i-1}} d_\ell \cdot D \cdot s$. ∎

We are now in a position to show a crucial result.

THEOREM 14. *Let Σ be a set of semi-LAV tgd's, and I and K be two ground instances such that $K \subseteq I$. Then the problem of deciding if there exists a Σ-satisfying homomorphism from $chase_\Sigma(K)$ into I is polynomial.*

Proof: We know that the result of chasing a ground instance with a set of semi-LAV tgd's has the size of each block bounded by a constant that depends only on Σ. Also, we know from the definition of semi-LAV tgd's that unsafe relational symbols (that is relational symbols that may contain variables generated during the chase process) in Σ appear only in antecedents of LAV tgd's.

The following algorithm takes as input a set Σ of semi-LAV tgd's, a ground instance I, and the constant c from Theorem 4. The algorithm returns TRUE if there is a Σ-satisfying homomorphism from from $chase_\Sigma(K)$ into I, and returns FALSE otherwise. For a block b of the variables of J, we denote by J_b the set of all tuples where variables of b occur. Also, for an instance I, we denote with I_u the subset of unsafe tuples in I.

HOMOM-SEARCH(K, I, Σ, c)

```
1  J ← chase_Σ(K), result ← TRUE
2  for each block b ∈ blocks(G^J)
3        do result_b ← FALSE
4        do for each homom. h_b from b to I
5            for each tuple t ∈ h_b(J_b)
6            for i ← 1 to c
7            for A_{b,t} : {t} ⊆ A_{b,t} ⊆ I_u, |A_{b,t}| = i
8                 do if (h_b(b) ∪ A_{b,t}) ⊨ Σ
9                       then result_b ← TRUE
10       do result ← result & result_b
11 return result
```

Suppose that the algorithm returns TRUE. Let b be a block in G^J. Based on an enumeration of $dom(K \cup I)$, let h_b be the first homomorphism the algorithm discovers, and for each $t \in J_b$, let $A_{b,t}$ the smallest corresponding set, such that $(h_b(J_b) \cup A_{b,t}) \models \Sigma$. Let

$$J^\Sigma = \bigcup_{b \in blocks(G^J)} (h_b(J_b) \cup \{A_{b,t} : t \in h_b(J_b)\}).$$

We claim that $h = \bigcup\{h_b : b \in blocks(G^J)\}$ is a Σ-satisfying homomorphism form J to I. Clearly $h(J) \subseteq J^\Sigma \subseteq I$. We thus have to show that $J^\Sigma \models \Sigma$, that is, that $J^\Sigma \models \xi$ for each $\xi \in \Sigma$. There are two cases to consider:

Case 1: ξ is a LAV-tgd whose antecedent consists of a single atom over an unsafe predicate. Let $t \in J^\Sigma$ be an

unsafe tuple triggering ξ. Then there is a block b in G^J, such that $t \in h_b(J_b)$, or $t \in A_{b,t}$. Then $(h_b(b) \cup A_{b,t}) \models \xi$. Since LAV-tgd's are closed under union [8], and all sets $A_{b,t}$ contain unsafe tuples only, it follows that $J^\Sigma \models \xi$.

Case 2: The antecedent of ξ consists of possibly several atoms, and all these atoms are over safe predicates. Since any possible set of triggering tuples is already present in J, and J was obtained by chasing K with Σ, it follows that $J^\Sigma \models \xi$.

To show that the algorithm is complete, let $J = chase_\Sigma(K)$, and suppose that there exists a homomorphism h, and an an instance $L \models \Sigma$, such that $h(J) \subseteq L \subseteq I$. Let h be the smallest such homomorphism, based on the enumeration of $dom(K \cup I)$. We need to show that the algorithm returns TRUE. The homomorphism h can obviously be expressed as a union of homomorphisms h_b, each from a block of $b \in blocks(G^J)$. Thus the algorithm will certainly discover h. Since $h(J) \subseteq L$, and $L \models \Sigma$, Theorem 4 guarantees that for each b and tuple $t \in I_u$, the corresponding set $A_{b,t} \subseteq L$ will be discovered by the algorithm. Consequently, the algorithm will return TRUE.

It remains to analyze the time complexity of the algorithm. We know from Theorem 3 that the size of J is polynomial in the size of K and that J therefore has a polynomial number of blocks. The same theorem also tells us that for each block of $b \in blocks(G^J)$ there are at most $|I|^c$ homomorphisms from b into I. Similarly to the algorithm in Theorem 5, line 8 is repeated at most $O(|I|^c)$ times. Finally, the test if $(h_b(b) \cup A_{b,t}) \models \Sigma$ can be done in space logarithmic in $|I|$. In conclusion, the algorithm runs in time polynomial in $|K| + |I|$. ∎

The preceding theorems allow us to derive polynomial time algorithms for all problems, not covered in Section 3, for the case when Σ is a set of semi-LAV tgd's.

THEOREM 15. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of semi-LAV tgd's. Then the problem*

EXISTENCE-OF-SOLUTION(Σ, subset, non-uniform)

can be solved in polynomial time.

Proof: Let (I_1, I_2) be an instance of the problem. In order to determine if there exists an instance $K_2 \subseteq I_2$, such that $(I_1, K_2) \models \Sigma$, it suffices to determine if there exists a Σ-satisfying homomorphism from $chase_\Sigma(I_1, \varnothing)$ to (I_1, I_2). Since Σ is a set of semi-LAV tgd's, it follows from Theorems 14 and 3 that the latter condition can be tested in polynomial time. ∎

THEOREM 16. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of semi-LAV tgd's. Then the problem*

EXISTENCE-OF-SOLUTION(Σ, superset, non-uniform)

is polynomial.

Proof: Let (I_1, I_2) be an instance of the problem. First compute $(K_1, K_2) = chase_\Sigma(I_1, I_2)$. Then, search for a homomorphism h, such that $h(K_1) \subseteq I_1$, and $(h(K_1), h(K_2)) \models \Sigma$. The crucial point is that (K_1, K_2) has bounded blocksize, since Σ is semi-LAV. ∎

THEOREM 17. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of semi-LAV dependencies. Then the problem*

EXISTENCE-OF-SOLUTION($\Sigma, \oplus$, non-uniform)

is polynomial.

Proof: Chase $(I_1, \varnothing)$ with Σ and search for a homomorphism h, such that $h(chase_\Sigma(I_1, \varnothing))) \models \Sigma$. Theorems 3 and 13 guarantee that this can be done in polynomial time. ∎

THEOREM 18. *Let Σ be a set of semi-LAV dependencies. Then the problem*

REPAIR-CHECKING($\Sigma, *$)

is polynomial, for $*$ being any of subset, superset, and $\oplus$.

Proof: Let us first consider the $\oplus$-case. We want to check if an instance K is a $\oplus$-repair of I w.r.t. Σ. First test if $K \models \Sigma$. If so, K is indeed a $\oplus$-repair if and only if both conditions in Theorem 1 are fulfilled. Since Σ is semi-LAV, the required homomorphism tests can be done in polynomial time, as per Theorem 14. For the subset and superset cases, it suffices to test condition 2, and condition 1, respectively, of Theorem 1. ∎

From the previous theorem and Lemma 2 we can conclude:

THEOREM 19. *Let $\Sigma = \Sigma_{12} \cup \Sigma_{21}$ be a set of semi-LAV dependencies. The problem*

SOLUTION-CHECKING($\Sigma, *$, uniform)

can be solved in polynomial time, for $*$ being any of subset, superset, and $\oplus$. ∎

Here is the final piece.

THEOREM 20. *Let Σ be a set of semi-LAV dependencies. Then the problem*

SOLUTION-CHECKING($\Sigma, *$, non-uniform)

is polynomial, for $*$ being any of subset, superset, and $\oplus$.

Proof: The subset and superset cases follows directly from Theorems 6 and 7 respectively. In the $\oplus$-case we need to check if an instance K_2 is a $\oplus$-solution for the non-uniform correspondence problem of (I_1, I_2) w.r.t. $\Sigma = \Sigma_{12} \cup \Sigma_{21}$. First test if $(I_1, K_2) \models \Sigma$. If so, K_2 is indeed a $\oplus$-solution for the non-uniform correspondence problem if and only if both conditions in Theorem 2 are fulfilled. Since Σ is semi-LAV, the required homomorphism tests can be done in polynomial time, as per Theorem 14. ∎

6. SUMMARY

For an overview of the complexities, the results are summarized in the three tables below. In Table 1 the complexities for the solution existence problems are presented, by considering different types of dependencies. For database repair and uniform correspondence cases the superset and symmetric difference solutions always exist. The semi-LAV

Table 1: The complexity of existence for correspondence solution/database repair

EXISTENCE Σ	db-repair weakly acyc.	unif. corr. sol. weakly acyc.	non-unif. corr. sol. weakly acyc.	non-unif. corr. sol. semi-LAV
subset	P *Thrm. 5*	P *Cor. 1*	NPC *Thrm. 12*	P *Thrm. 15*
superset	-	-	NPC *Thrm. 12*	P *Thrm. 16*
$\oplus$	-	-	NPC *Thrm. 12*	P *Thrm. 17*

Table 3: The complexity of solution check

SOLUTION CHECK Σ	uniform weakly acyc.	uniform semi-LAV	non-uniform weakly acyc.	non-uniform semi-LAV
subset	coNPC *Thrm. 11*	P *Thrm. 19*	P *Thrm. 6*	P *Thrm. 6*
superset	coNPC *Thrm. 11*	P *Thrm. 19*	P *Thrm. 7*	P *Thrm. 7*
$\oplus$	coNPC *Thrm. 11*	P *Thrm. 19*	coNPC *Thrm. 8*	P *Thrm. 20*

Table 2: The complexity of database repair

REPAIR CHECK	Σ weakly acyc.	Σ semi-LAV
subset	coNPC [2]	P *Thrm. 18*
superset	coNPC *Cor. 10*	P *Thrm. 18*
$\oplus$	coNPC [2]	P *Thrm. 18*

dependencies make the problems tractable for the non-uniform correspondence case even though the existence check is NP-complete for general weakly acyclic tgd's.

Table 2 summarizes the complexities of the database repair check problems. Afrati and Kolaitis proved in [2] the coNP-completeness for general weakly acyclic tgd's for subset and symmetric difference repair check. In this paper we showed that the intractability also holds for superset repair check. When replacing the weakly acyclic tgd's with semi-LAV dependencies the repair check, for all cases, becomes polynomial.

Finally, Table 3 tallies the complexities for the solution checking problems for both the uniform and non-uniform settings. For the uniform correspondence problems the semi-LAV dependencies enable tractability for all cases (subset, superset and symmetric difference). When the non-uniform correspondence setting is considered, the semi-LAV dependencies add tractability only for the symmetric difference case, the subset and superset cases were proved tractable already for general weakly acyclic tgd's.

Evidently solution checking and solution existence are computationally related. For instance, if solution checking is decidable, it follows that establishing the existence of a solution is recursively enumerable. Likewise, since our solutions are of polynomial size in a finite search space, if solution checking is doable in polynomial time, then solution existence can be determined in non-deterministic polynomial time. However, the main hurdle lies in verifying the desired properties. Indeed, in some cases the problem does jump from P to NP, for example Theorems 6 and 7 vs. Theorems 12. In other cases, for example in Theorem 6 and 7 vs. Theorem 15 and 16, both problems are in P. On the other hand, there are two exceptions, where solution checking is intractable, but determining the existence of a solution is polynomial. These are Theorem 9 [2], vs. Theorem 5, and Theorem 11 vs. Corollary 1.

For the 9 vs. 5 case, we note that the existence version is easier, since we don't have to verify any maximality. If a subset satisfying the dependencies exist, surely a maximal one also exists.

For the 7 vs. 12 case, we note that the checking problem is easier, since we only have to test if even one tuple can be deleted from the candidate. If so, surely the candidate is not minimal. On the other hand, for Theorem 12 we can construct an instance such that a consistent subset of the candidate can exist if and only if the intractable reductee has a solution. Note however, that opposed to Theorems 6 and 7, where we can inspect single tuples, in Theorem 8 this is no longer true: we have to simultaneously deal with insertable and deletable tuples. Consequently the combinatorial blow up.

7. CONCLUSIONS

In this paper we introduced the data correspondence setting and carried out a systematic investigation of the associated computational problems. One of the main results is the introduction of the new class of semi-LAV dependencies that are an extension of full and LAV dependencies. The class of semi-LAV dependencies has the desirable property that all the considered problems become tractable. The class is optimal in the sense that relaxing the condition results in intractability, which can be seen from the many proofs of Section 4.

In the current work we only considered tuple generating dependencies. Since equality generating dependencies easily lead to intractability [7, 9, 10] they are left for future work. We also omitted the cardinality based repairs as these types of repairs are intractable even in the case of full and LAV tuple generating dependencies.

Compared to the Peer Data Exchange problem [14] the superset solution existence/check for the non-uniform correspondence problem adds an extra minimality condition to the solution. Even if the classes of semi-LAV dependencies and the tractable class in [14] are not comparable, the semi-LAV dependencies introduce tractability also when there are target dependencies, something that could not be handled by [14].

Query answering in correspondence settings was not explicitly considered here. On one hand, a conjunctive query is expressible as a tgd, but it remains to study the problem of deciding when a tuple belongs to all solutions/repairs. The general problem is known to be Π_2^p-complete, but a more fine grained analysis is still lacking.

Perhaps the most challenging problem is to allow the input instances to contain null values. This is a natural requirement, since database instances should be closed under data exchange [13]. The presence of null values in the input instances however destroys the the bounded blocksize of the chase, thus presenting a problem not easy to cope with.

Acknowledgements

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

8. REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, pages 31–41, 2009.

[3] Alfred V. Aho, Catriel Beeri, and Jeffrey D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, 1979.

[4] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.

[5] Pablo Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[6] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[7] Leopoldo E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.

[8] Balder ten Cate and Phokion G. Kolaitis. Structural characterizations of schema-mapping languages. In *ICDT*, pages 63–72, 2009.

[9] Jan Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, pages 1–17, 2007.

[10] Jan Chomicki and Jerzy Marcinkowski. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*, pages 119–150, 2005.

[11] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[12] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.

[13] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.

[14] Ariel Fuxman, Phokion G. Kolaitis, Renée J. Miller, and Wang Chiew Tan. Peer data exchange. *ACM Trans. Database Syst.*, 31(4):1454–1498, 2006.

[15] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[16] Georg Gottlob and Alan Nash. Data exchange: computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

[17] Gösta Grahne and Alberto O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *ICDT*, pages 332–347, 1999.

[18] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[19] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[20] Andrei Lopatenko and Leopoldo E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, pages 179–193, 2007.

[21] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.

[22] Slawomir Staworko and Jan Chomicki. Consistent query answers in the presence of universal constraints. *CoRR*, abs/0809.1551, 2008.