Chase Termination: A Constraints Rewriting Approach

Sergio Greco
DEIS, Università della Calabria
87036 Rende (Cs), Italy
greco@deis.unical.it

Francesca Spezzano
DEIS, Università della Calabria
87036 Rende (Cs), Italy
fspezzano@deis.unical.it

ABSTRACT

Several database areas such as data exchange and integration share the problem of fixing database instance violations with respect to a set of constraints. The chase algorithm solves such violations by inserting tuples and setting the value of nulls. Unfortunately, the chase algorithm may not terminate and the problem of deciding whether the chase process terminates is undecidable. Recently there has been an increasing interest in the identification of sufficient structural properties of constraints which guarantee that the chase algorithm terminates [8, 10, 14, 15].

In this paper we propose an original technique which allows to improve current conditions detecting chase termination. Our proposal consists in rewriting the original set of constraints Σ into an 'equivalent' set Σ^α and verifying the structural properties for chase termination on Σ^α. The rewriting of constraints allows to recognize larger classes of constraints for which chase termination is guaranteed. In particular, we show that if Σ satisfies chase termination conditions $\mathcal{T}$, then the rewritten set Σ^α satisfies $\mathcal{T}$ as well, but the vice versa is not true, that is there are significant classes of constraints for which Σ^α satisfies $\mathcal{T}$ and Σ does not.

1. INTRODUCTION

Several database areas such as data integration, data warehouse, data exchange, consistent query answering, query optimization, etc., share the problem of fixing database instance violations with respect to a set of constraints [4, 6, 7, 11, 12, 13]. The chase algorithm solves such violations by inserting tuples and setting the value of nulls [1, 3]. Unfortunately, the chase algorithm may not terminate and the problem of deciding whether the chase process terminates is undecidable. Recently there has been an increasing interest in the identification of sufficient structural properties of constraints which guarantee that the chase algorithm terminates [8, 10, 14, 15]. The following example shows a set of constraints for which the chase could be non-terminating.

Proceedings of the VLDB Endowment, Vol. 3, No. 1
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

EXAMPLE 1. We are given a digraph stored into the unary relation N, denoting nodes, and the binary relation E, denoting oriented edges; we also have a unary relation S, denoting special nodes. Consider the set of constraints Σ_1:

$$\forall x \; [\; N(x) \rightarrow \exists y \; E(x,y) \;]$$
$$\forall (x,y) \; [\; E(x,y) \rightarrow N(y) \;]$$

stating that i) every node must have an outgoing edge and ii) every edge must end into a node. Assume to have the database instance consisting of the tuple $N(a)$. Since the database does not satisfy the first constraint, a tuple $E(a,n_1)$ should be inserted, where n_1 is a new labeled null value. At this point the second constraint is not satisfied and the tuple $N(n_1)$ should be added to the database. Continuing with the chase process of adding tuples to make the database consistent, an infinite number of tuples $E(n_1,n_2)$, $N(n_2), E(n_2,n_3), N(n_3), \ldots$ should be inserted. □

Fagin et al. [10] introduced the class of *weakly acyclic* sets of constraints. Informally, weak acyclicity checks that the set of constraints does not present cyclic conditions for which a new null value forces (directly or indirectly) the introduction of another null in the same position. In the above example we have that the presence of a value in N_1, i.e. in the first position of the predicate N, forces the introduction of a new null value in position E_2, i.e. in the second position of the predicate E, and it is denoted as $N_1 \xrightarrow{*} E_2$; this value is then introduced in position N_1 (denoted as $E_2 \rightarrow N_1$) and next a new null value is introduced in E_2. The cycle going through the special edge $N_1 \xrightarrow{*} E_2$ means that an infinite number of nulls could be introduced.

The class of weakly acyclic sets of constraints has been generalized in several works [8, 14, 15]. Deutsch et al. proposed an extension of weak acyclicity called *stratification* [8]. The idea behind stratification is to decompose the set of constraints into independent subsets, where each subset consists of constraints that may fire each other, and to check each component separately for weak acyclicity. However, in [16] it has been shown that stratification is not able to check termination of all chase sequences (as weak acyclicity does), but it is sufficient to state that if a set of constraints is stratified, then there is at least one terminating chase sequence which can be determined from the chase graph.

Meier et al. proposed a different extension of weak acyclicity called *safety* [15]. The improvement is based on the fact that only the effective propagation of null values should be considered in the graph. Thus, a variable can propagate nulls only if all its occurrences appear in 'affected' positions, i.e. positions which may actually contain null values [5]. The following example presents a set of constraints which is both safe and stratified, but not weakly acyclic.

Example 2. Consider the below set of constraints Σ_2

$$\begin{array}{l} \forall x\ [\ N(x) \rightarrow \exists y\ E(x,y)\ ] \\ \forall (x,y)\ [\ S(y) \wedge E(x,y) \rightarrow N(y)\ ] \end{array}$$

where the second constraint states that if an edge ends into a special node y, then y must be a (normal) node as well. This set is not weakly acyclic since $N_1 \xrightarrow{*} E_2$ and $E_2 \rightarrow N_1$. However, E_2 does not propagate null values to N_1 as the variable y also appears in the relation S which does not contain null values. So, position N_1 cannot contain nulls, i.e. it is not affected. $\Box$

Stratification and safety are not comparable (i.e. there are sets of constraints which only satisfy one of the two criteria). A different extension of weak acyclicity has been introduced in [14] under the name of *Super-weak Acyclicity* (SwA). Basically, SwA takes into account the fact that variables may appear more than once in body atoms of constraints and, therefore, when different nulls are inserted in positions associated with the same variable, constraints are not fired.

Example 3. Let Σ_3 be the below set of constraints:

$$\begin{array}{l} r_1 : \forall (x,y)\ [\ E(x,x) \wedge N(y) \rightarrow \exists z\ E(x,z) \wedge S(z)\ ] \\ r_2 : \forall x\ [\ S(x) \rightarrow N(x)\ ] \end{array}$$

The chase process over the above set of constraints always terminates since the first constraint will never fire itself even transitively. Indeed, the firing of constraint r_1 introduces a null value in position E_2 (different from the value in position E_1) and the body of the constraint requires that the two values in E must coincide. $\Box$

It is worth noting that Σ_3 is super-weakly acyclic, but neither safe nor stratified (since there are database instances for which the activation of the first constraint fires the second one and the activation of the second constraint fires the first one). However, super-weak acyclicity, as well as other sufficient conditions such as stratification, is not able to detect other cases where chase terminates. The below example shows a set of constraints which is neither safe nor stratified nor super-weakly acyclic, where the chase terminates for all database instances.

Example 4. Consider the set of constraints Σ_4:

$$\begin{array}{l} \forall x\ [\ N(x) \rightarrow \exists y\ E(x,y)\ ] \\ \forall (x,y)\ [\ S(x) \wedge E(x,y) \rightarrow N(y)\ ] \end{array}$$

where the second constraint states that if there exists an edge from a (special) node x to a node y, then y must be a (normal) node as well. Assume that the database contains the tuples $S(a), N(a)$. Since the first constraint is not satisfied, the tuple $E(a, n_1)$ is inserted. This update operation fires the second constraint to insert the tuple $N(n_1)$ which in turn fires the first constraint so that the tuple $E(n_1, n_2)$ is added to the database. At this point the chase terminates since the database is consistent, i.e. the second constraint cannot be fired because n_1 is not in the relation S. $\Box$

Thus, in this paper we propose an original technique which allows to improve current conditions detecting chase termination. Our proposal consists in rewriting the original set of constraints Σ into an 'equivalent' set Σ^α and verifying the structural properties for chase termination on Σ^α. The rewriting of constraints allows to recognize larger classes of constraints for which chase termination is guaranteed. In particular, we show that if Σ satisfies the chase termination conditions $\mathcal{T}$, defined on the base of structural properties, then the rewritten set Σ^α satisfies $\mathcal{T}$ as well, but the vice versa is not true, that is there are significant classes of constraints for which Σ^α satisfies $\mathcal{T}$ and Σ does not.

The safeness criterion improves over weak acyclicity as it only considers affected positions, that is positions where labeled nulls can be copied. To further improve our rewriting technique, we distinguish these positions into weakly and strongly affected. In particular, positions where a finite number of labeled nulls may be copied are called weakly affected, whereas the remaining positions are called strongly affected.

Contributions. In this paper a more in depth analysis to detect sufficient conditions for chase termination is performed:

- We first analyze the relationship among current criteria and show that super-weak acyclicity is not comparable with (c-)stratification, but it extends safety;
- We present a technique for rewriting a set of constraints into an 'equivalent' set by adorning predicate symbols and show that the target set allows to detect larger classes of source constraints for which all chase sequences terminate;
- We extend the rewriting technique to capture even larger classes of constraints for which the chase terminates by analyzing affected positions (positions which may hold nulls);
- We introduce the classes of weakly and strongly affected positions (a partition of affected positions) and show that the rewriting algorithm can be enhanced by taking into account the type of affected positions;

It is important to observe that for the sake of presentation we first introduce the basic technique (Section 3) and next two improvements (Sections 4 and 5) which analyze in depth the flow of nulls. For space limitation formal definitions on previous criteria and some proofs are reported in Appendix B and C. Complete proofs of the results reported here will appear in the full version of the paper.

2. PRELIMINARIES

We introduce the following disjunct sets of symbols: (i) an infinite set *Consts* of *constants*, (ii) an infinite set *Nulls* of *labeled nulls* and (iii) an infinite set *Vars* of variables. A *relational schema* $\mathbf{R}$ is a set of relational predicates R, each with its associated arity $ar(R)$. An *instance* of a relational predicate R of arity n is a set of ground atoms in the form $R(c_1, \ldots, c_n)$, where $c_i \in Consts \cup Nulls$. Such (ground) atoms are also called tuples or facts. We denote by D a database instance constructed on $Consts$ and by J, K the database instances constructed on $Consts \cup Nulls$. Let K be a database over a relational schema $\mathbf{R}$ and $S \subseteq \mathbf{R}$, then $K[S]$ denotes the subset of K consisting of instances whose predicates are in S (clearly $K = K[\mathbf{R}]$). Analogously, if we have a collection of databases $K_C = \{K_1, \ldots, K_n\}$ where each K_i is defined over a schema $\mathbf{R}_i$ and let $S \subseteq \cap_{i \in [1 \ldots n]} \mathbf{R}_i$, then $K_C[S] = \{K_1[S], \ldots, K_n[S]\}$.

A *position* R_i is a pair (R, i), where R is a relation predicate belonging to the schema $\mathbf{R}$ and i denotes the i-wise attribute of R. Given an instance K, $Nulls(K)$ denotes the set of labeled nulls occurring in K. An *atomic formula* (or *atom*) is of the form $R(t_1, ...t_n)$ where R is a relational predicate, $t_1, ..., t_n$ are terms belonging to the domain $Consts \cup Vars$ and $n = ar(R)$. Given a relational schema $\mathbf{R}$, a *tuple generating dependency* (TGD) over $\mathbf{R}$ is a formula of the form $r : \forall \mathbf{x}[\phi_{\mathbf{R}}(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_{\mathbf{R}}(\mathbf{x}, \mathbf{y})]$, where $\phi_{\mathbf{R}}(\mathbf{x})$ and $\psi_{\mathbf{R}}(\mathbf{x}, \mathbf{y})$ are conjunctions of atomic formulas over $\mathbf{R}$; $\phi_{\mathbf{R}}(\mathbf{x})$ is called the *body* of r, denoted as $Body(r)$, while $\psi_{\mathbf{R}}(\mathbf{x}, \mathbf{y})$ is called the *head* of r, denoted as $Head(r)$. An *equality generating dependency* (EGD) over $\mathbf{R}$ is a formula of the form $\forall \mathbf{x}[\phi_{\mathbf{R}}(\mathbf{x}) \rightarrow (x_1 = x_2)]$, where x_1 and x_2 are among the variables in $\mathbf{x}$. In the following we will omit the subscript $\mathbf{R}$ from formulas, whenever the database schema is understood and the universal quantification, since we assume that variables appearing in the body are universally quantified and variables appearing only in the head are existentially quantified. In some cases we also assume that the head and body conjunctions are sets of atoms.

Homomorphism and Universal Solutions. Let K_1 and K_2 be two instances over $\mathbf{R}$ with values in $Consts \cup Nulls$. A *homomorphism* $h : K_1 \rightarrow K_2$ is a mapping from $Consts \cup Nulls(K_1)$ to $Consts \cup Nulls(K_2)$ such that: (1) $h(c) = c$, for every $c \in Consts$, and (2) for every fact $R_i(t)$ of K_1, we have that $R_i(h(t))$ is a fact of K_2 (where, if $t = (a_1, ..., a_s)$, then $h(t) = (h(a_1), ..., h(a_s))$). K_1 is said to be *homomorphically equivalent* to K_2 if there is a homomorphism $h\colon K_1 \rightarrow K_2$ and a homomorphism h': $K_2 \rightarrow K_1$.

We say that $D \models \Sigma$ if an instance D satisfies all the constraints in Σ. For any database instance D and set of constraints Σ over a database schema $\mathbf{R}$, a *solution* for (D, Σ) is an instance J such that $D \subseteq J$ and $J \models \Sigma$. A *universal solution* J is a solution such that for every solution J' there exists a homomorphism $h : J \rightarrow J'$. The set of universal solutions for (D, Σ) will be denoted by $USol(D, \Sigma)$.

Similar to homomorphisms between instances, a homomorphism h from a conjunctive formula $\phi(\mathbf{x})$ to an instance J is a mapping from the variables $\mathbf{x}$ to $Consts \cup Nulls(J)$ such that for every atom $R(x_1, ..., x_n)$ of ϕ the fact $R(h(x_1), ..., h(x_n))$ is in J.

Chase Step. Let K be a database instance.

1. Let r be a TGD $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K such that there is no extension of h to a homomorphism h' from $\phi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y})$ to K[1]. We say that r can be applied to K with homomorphism h. Let K' be the union of K with the set of facts obtained by: (a) extending h to h' such that each variable in $\mathbf{y}$ is assigned a fresh labeled null, followed by (b) taking the image of the atoms of ψ under h'. We say that the result of applying r to K with h is K', and write $K \overset{r,h}{\rightarrow} K'$.
2. Let r be an EGD $\phi(\mathbf{x}) \rightarrow (x_1 = x_2)$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K such that $h(x_1) \neq h(x_2)$. We say that r can be applied to K with homomorphism h. More specifically, we distinguish two cases.
(a) If both $h(x_1)$ and $h(x_2)$ are in $Consts$ the result of applying r to K with h is "failure", and $K \overset{r,h}{\rightarrow} \perp$.
(b) Otherwise, let K' be K where we identify $h(x_1)$ and $h(x_2)$ as follows: if one is a constant, then the labeled null is replaced everywhere by the constant; if both are labeled nulls, then one is replaced everywhere by the other. We say that the result of applying r to K with h is K', and write $K \overset{r,h}{\rightarrow} K'$.

[1]A variant of this step is the *oblivious* one that applies to an instance K if there is a homomorphism h from $\phi(\mathbf{x})$ to K (see Appendix B for details).

Definition 1 (Chase [10]). Let Σ be a set of TGDs and EGDs, and let K be an instance.

- A *chase sequence of K with Σ* is a sequence (finite or infinite) of chase steps $K_i \overset{r,h_i}{\rightarrow} K_{i+1}$, with $i = 0, 1, ...$, $K_0 = K$ and r a dependency in Σ.
- A *finite chase of K with Σ* is a finite chase sequence $K_i \overset{r,h_i}{\rightarrow} K_{i+1}$, $0 \leq i < m$, with the requirement that either (a) $K_m = \perp$ or (b) there is no dependency r of Σ and there is no homomorphism h_m such that r can be applied to K_m with h_m. We say that K_m is the result of the finite chase. We refer to case (a) as the case of a *failing finite chase* and we refer to case (b) as the case of a *successful finite chase*. □

Observe that whenever several chase steps apply, the chase picks one nondeterministically. Therefore, there are instances and sets of constraints for which certain choices lead to terminating chase sequences, while others to non-termination.

In [10] it has been shown that, for any instance D and set of constraint Σ: (i) if J is the result of some successful finite chase of $\langle D, \Sigma \rangle$, then J is a universal solution; (ii) if some failing finite chase of $\langle D, \Sigma \rangle$ exists, then there is no solution.

Data Exchange. A data exchange setting $(\mathbf{S}, \mathbf{T}, \Sigma_{\mathtt{st}}, \Sigma_{\mathtt{t}})$ consists of a source schema $\mathbf{S}$, a target schema $\mathbf{T}$, a set $\Sigma_{\mathtt{st}}$ of source-to-target dependencies, and a set $\Sigma_{\mathtt{t}}$ of target dependencies. Each source-to-target dependency in $\Sigma_{\mathtt{st}}$ is a TGD $\phi_{\mathbf{S}}(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$, whereas each target dependency in $\Sigma_{\mathtt{t}}$ is either a TGD $\phi_{\mathbf{T}}(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ or an EGD $\phi_{\mathbf{T}}(\mathbf{x}) \rightarrow (x_1 = x_2)$. The data exchange problem associated with this setting is the following: given a finite source instance D over $\mathbf{S}$, find a finite target instance J over $\mathbf{T}$ such that $\langle D, J \rangle$ satisfies $\Sigma_{\mathtt{st}}$ and J satisfies $\Sigma_{\mathtt{t}}$. J is called a solution for D or, simply a solution if D is understood. The set of all universal solutions for D is denoted by $USol(D, \Sigma_{\mathtt{st}} \cup \Sigma_{\mathtt{t}})$. Note that the input to a data exchange problem is a source instance only; the data exchange setting itself (i.e. source and target schemas and dependencies) is considered fixed.

2.1 Chase termination conditions

As said in the introduction, several criteria identifying sufficient conditions for chase termination have been defined in the recent literature: Weak Acyclicity (WA) [10], Safe Conditions (SC) [15], Stratification (Str) [8], C-Stratification (CStr) [16], Inductive Restriction (IR) [15, 16] and Super-Weak Acyclicity (SwA) [14]. For space limitations the formal definitions of these criteria are reported in Appendix B.

Let Σ be a set of constraints over a database schema $\mathbf{R}$, then $pos(\Sigma)$ denotes the set of positions R_i such that R denotes a relational predicate of $\mathbf{R}$ and there is an R-atom appearing in Σ. Weak acyclicity is based on the construction of a directed graph $dep(\Sigma) = (pos(\Sigma), E)$, called the dependency graph, where E is defined as follows. For every TGD $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ in Σ, then: i) for every x in $\mathbf{x}$ occurring in position R_i in ϕ and in position S_j in ψ, add an edge

$R_i \rightarrow S_j$ (if it does not already exist); ii) for every x in $\mathbf{x}$, appearing both ϕ (in position R_i) and in ψ, and for every y in $\mathbf{y}$ appearing in position T_k in ψ, add a special edge $R_i \xrightarrow{*} T_k$ (if it does not already exist). Σ is *weakly acyclic* if $dep(\Sigma)$ has no cycle going through a special edge. Weak acyclicity condition guarantees that all chase sequences terminate.

An *affected position* denotes a position in which null values may appear, that is it can also take values from $Nulls$. A position R_i is said to be affected in there is a constraint $r : \phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ in Σ and either i) there is a variable y in $\mathbf{y}$ appearing in position R_i in ψ, or ii) there is a variable x in $\mathbf{x}$ appearing both in position R_i in ψ and only in affected positions in the body of r. The set of affected positions of Σ is denoted by $aff(\Sigma)$.

Given a set of TGDs Σ, the *propagation graph* of Σ, denoted as $prop(\Sigma) = (aff(\Sigma), E')$, is a subset of $dep(\Sigma) = (pos(\Sigma), E)$ such that E' contains the edges in E whose positions are affected[2]. Moreover, Σ is said to be safe if $prop(\Sigma)$ does not contain cycles with special edges. Safety condition guarantee that all chase sequences terminate as well.

The stratification builds the *chase graph* $G(\Sigma) = (\Sigma, E)$ where nodes are the constraints in Σ and an edge from r_i to r_j (denoted as $r_i \prec r_j$) means that fire of r_i can cause r_j to fire. Σ is stratified iff the constraints in every cycle of $G(\Sigma)$ are weakly acyclic. Stratification guarantees, as shown in [16], that, for every database D, there is a chase sequence (but not all) which terminates in polynomial time in the size of D. The following example shows such a case.

Example 5. Consider the following set of constraints Σ_5:

$$\begin{array}{ll} r_1: & R(x) \rightarrow S(x,x) \\ r_2: & S(x_1,x_2) \rightarrow \exists z\, T(x_2,z) \\ r_3: & S(x_1,x_2) \rightarrow T(x_1,x_2) \wedge T(x_2,x_1) \\ r_4: & T(x_1,x_2) \wedge T(x_1,x_3) \wedge T(x_3,x_1) \rightarrow R(x_2) \end{array}$$

taken from [16]. Σ_5 is stratified since $r_1 \prec r_2$, $r_1 \prec r_3 \prec r_4 \prec r_1$, and the set of constraints $\{r_1, r_3, r_4\}$ is weakly acyclic. Moreover, assuming that the database only contains the tuple $R(a)$, the chase firing repeatedly r_1, r_2, r_3 and r_4 never terminates, while the chase which never fires r_2 terminates successfully. □

In order to cope with this problem, the *c-stratification* criterion has been proposed in [16]. As well as weak acyclicity and safety, c-stratification guarantees that for every database D all chase sequences terminate in polynomial time in the size of D. A more refined extension of c-stratification and safety has been proposed in [15, 16] under the name of *inductive restriction.*

The super-weak acyclicity (SwA) builds a *trigger graph* $G'(\Sigma) = (\Sigma, E)$ where edges define relations among constraints. An edge $r_i \rightsquigarrow r_j$ means that a null value introduced by a constraint r_i is propagated (directly or indirectly) into the body of r_j. A set of constraints Σ is super-weakly acyclic iff the trigger graph is acyclic. With respect to other criteria, SwA also takes into account the fact that a variable may occur more than once in the same atom. SwA extends WA and guarantees the termination of all chase sequences in polynomial time in the size of the input database.

2.2 SwA versus SC and (C–)Str

We now analyze more deeply the relationship among the criteria proposed in the literature. The relationship among WA, Str, CStr, SC and IR have already been investigated in [8] and [16]. In particular, let $\mathcal{WA}$, $\mathcal{S}tr$, $\mathcal{CS}tr$, $\mathcal{IR}$, $\mathcal{SC}$ and $\mathcal{S}w\mathcal{A}$[3] denote the classes of constraints which are weakly acyclic, stratified, c-stratified, inductively restricted, safe and super-weakly acyclic, respectively, it has been shown that [8, 16]:

- $\mathcal{WA} \subsetneq \mathcal{SC}$, $\mathcal{WA} \subsetneq \mathcal{CS}tr$ and $\mathcal{CS}tr \not\parallel \mathcal{SC}$[4], i.e. the classes of constraints which are, respectively, c-stratified and safe both generalize the class of constraints which are weakly acyclic, but they are not comparable,
- $\mathcal{CS}tr \subsetneq \mathcal{IR}$ and $\mathcal{SC} \subsetneq \mathcal{IR}$, i.e. the c-stratified and safe classes are generalized by the class of constraints which are inductively restricted. Obviously $\mathcal{CS}tr \subsetneq \mathcal{S}tr$.

We now analyze the relationship between the above discussed classes and $\mathcal{S}w\mathcal{A}$. Since $\mathcal{S}w\mathcal{A}$ is defined only for oblivious chase we have to slightly reformulate, as done in [14], the definitions of $\mathcal{WA}$ and $\mathcal{SC}$. In particular, the weak acyclicity and safety criteria for oblivious chase are the same as in standard chase except that we have to add edges to the dependency and propagation graphs "for every TGD $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ and for every x in $\mathbf{x}$" (see Definition 4), relaxing the constraint that x has to occur in ψ. The comparisons done in this paper make the above assumption.

Concerning super-weak acyclicity and c-stratification we have that, as stated by the following proposition, they are not comparable.

Proposition 1. $\mathcal{S}w\mathcal{A} \not\parallel \mathcal{CS}tr$. □

Corollary 1. $\mathcal{S}w\mathcal{A} \not\parallel \mathcal{IR}$. □

Since the set of constraints of Example 3 is not safe, $\mathcal{S}w\mathcal{A} \not\subseteq \mathcal{SC}$, that is $\mathcal{SC}$ is not more general that $\mathcal{S}w\mathcal{A}$. Therefore, the question is: "does $\mathcal{SC} \subseteq \mathcal{S}w\mathcal{A}$"?

Theorem 1. $\mathcal{SC} \subsetneq \mathcal{S}w\mathcal{A}$. □

The previous results state that Super-weak Acyclicity is not comparable with C-Stratification and generalizes Safety.

3. CONSTRAINTS REWRITING

In this section we present a technique for checking chase termination based on rewriting the original set of TGDs Σ into an 'equivalent' set Σ^α and verifying the structural properties for chase termination on Σ^α. The technique performs a deep analysis of constraints by considering pattern analysis through the introduction of adornments associated with predicates. The adornments here considered are similar to those used in binding propagation in deductive databases (e.g. magic-set) for the optimization of bound queries [2]. Before presenting our rewriting technique we introduce some definitions concerning constraints equivalence. In particular, the equivalence between two sets of constraints Σ_1 and Σ_2 defined, respectively, over two schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, is given with respect to two sets of relations $R, S \subseteq \mathbf{R}_1 \cap \mathbf{R}_2$ called, respectively, input and output relations.

[2]Since $aff(\Sigma) \subseteq pos(\Sigma)$

[3]We are using the calligraphic style to distinguish the class of constraints recognized by the criterion.

[4]The notation $A \not\parallel B$ is a shorthand for $A \not\subseteq B$ and $A \not\supseteq B$.

Definition 2 (Sets of constraints equivalence). Given two sets of constraints Σ_1 and Σ_2 over the two database schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively and two sets of relations $R, S \subseteq \mathbf{R}_1 \cap \mathbf{R}_2$, we say that $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ if for all database D over R, $USol(D, \Sigma_1)[S] \subseteq USol(D, \Sigma_2)[S]$. Moreover, we say that $\langle \mathbf{R}_1, \Sigma_1 \rangle$ and $\langle \mathbf{R}_2, \Sigma_2 \rangle$ are equivalent with respect to R/S and write $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ if both $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ and $\langle \mathbf{R}_2, \Sigma_2 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_1, \Sigma_1 \rangle$. □

When $R = S = \mathbf{R}_1 \cap \mathbf{R}_2$ we simply write $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq \langle \mathbf{R}_2, \Sigma_2 \rangle$ and $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv \langle \mathbf{R}_2, \Sigma_2 \rangle$.

Example 6. Consider the database schema $\mathbf{R}_1 = \{E(A, B)\}$ consisting of the binary relation E and the database schema $\mathbf{R}_2 = \{E(A, B), Q(C)\}$ consisting of the binary relation E and the unary relation Q. Assume to have the following sets of TGDs

$$\Sigma_1 = \{E(x, y) \rightarrow E(y, x)\} \text{ and}$$
$$\Sigma_2 = \{E(x, y) \rightarrow Q(x),\quad Q(x) \wedge E(x, y) \rightarrow E(y, x)\}$$

defined over $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively.

Clearly, $USol(D, \Sigma_1)[E] = USol(D, \Sigma_2)[E]$ for all databases D over $\mathbf{R}_1 \cap \mathbf{R}_2 = \{E\}$ and, therefore, $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv \langle \mathbf{R}_2, \Sigma_2 \rangle$. □

Adornments

An adornment α of a predicate p with arity m is a string of length m over the alphabet $\{b, f\}$. A predicate symbol p^α is said to be adorned, whereas an adorned atom is of the form $p^{\alpha_1 \cdots \alpha_m}(x_1, ..., x_m)$; if $\alpha_i = b$ we say that the variable x_i is bounded, otherwise ($\alpha_i = f$) we say that x_i is free. Intuitively, bounded terms can take values from finite domains; consequently, constant terms are always adorned with the symbol b. If each body variable of a TGD is associated with a unique adornment we say that the adornment of the body is coherent. Before introducing how constraints are adorned, let us introduce some further definitions and notations.

We assume that TGDs are in standard form, that is existentially quantified variables appear within the scope of universally quantified ones; variables appearing in constraints with empty body are replaced by Skolem constants.

Given a TGD $r : \phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{z}, \mathbf{y})$ with $\mathbf{z} \subseteq \mathbf{x}$ and let α be a coherent adornment for the body atoms, then $HeadAdn(r, \phi^\alpha(\mathbf{x}))$ denotes the *adornment of the head* of r (with respect to the adorned body $\phi^\alpha(\mathbf{x})$) obtained by adorning head atoms as follows: i) every universally quantified variable has the same adornment of the body occurrences, ii) constants are adorned as b; iii) existentially quantified variables are adorned as f.

Rewriting algorithm

Given a set of TGDs Σ over a schema $\mathbf{R}$ the corresponding rewriting set $Adn(\Sigma)$ consists of the union of four sets of TGDs: the base set $Base(\Sigma)$, the derived set $Derived(\Sigma)$, the input set $In(\Sigma)$ and the output set $Out(\Sigma)$.

The rewriting is performed by means of the function Adn reported in Figure 1. It starts by adorning, for each TGD, body predicates with strings of b symbols and adorning heads according to the body adornments by using the function $HeadAdn$ (base set); then, each new adorned predicate symbol is used to generate new adorned constraints until all adorned predicate symbols are used (derived set); at the end, TGDs mapping source relations into relations adorned with strings of b symbols (input set) and TGDs mapping relations having the same predicate and different adornments into a unique relation (output set) are added.

```
Function $Adn(\Sigma)$;
Input: Set of TGDs $\Sigma$ over a schema $\mathbf{R}$;
Output: The set of (adorned) TGDs $Base \cup Derived \cup In \cup Out$;
begin
  $Base = Derived = In = Out = New\_Pred = \emptyset$;
  // Let $Body^b(r)$ be the conjunction obtained by adorning
  // atoms in $Body(r)$ with strings of $b$ symbols
  $Used\_Pred = \bigcup_{r \in \Sigma} Body^b(r)$;
  for each $r \in \Sigma$ do begin
    $Base = Base \cup \{Body^b(r) \rightarrow HeadAdn(r, Body^b(r))\}$;
    $New\_Pred = New\_Pred \cup \{p^\alpha | p^\alpha(t) \in HeadAdn(r, Body^b(r))\}$;
  end for;
  while ($New\_Pred \neq \emptyset$) do begin
    Select nondeterministically $p^{\alpha_1 \cdot\cdot \alpha_n} \in New\_Pred$;
    $New\_Pred = New\_Pred - \{p^{\alpha_1 \cdot\cdot \alpha_n}\}$;
    $Used\_Pred = Used\_Pred \cup \{p^{\alpha_1 \cdot\cdot \alpha_n}\}$;
    for each $r \in (Base \cup Derived)$ do
      for each $p^\beta(x_1, ..., x_n) \in Body(r)$ do begin
        $B' = Body(r) - \{p^\beta(x_1, ..., x_n)\} \cup \{p^{\gamma_1 \cdot\cdot \gamma_n}(x_1, ..., x_n)\}$;
        $\gamma_i = b$ if $x_i \in Consts$;  $\gamma_i = \alpha_i$ if $x_i \in Vars$; $(i \in [..n])$;
        if $B'$ is coherent then
          $Derived = Derived \cup \{B' \rightarrow HeadAdn(Adn^{-1}(r), B')\}$;
          $New\_Pred = New\_Pred \cup \{p^\omega | p^\omega$ appears in
                $HeadAdn(Adn^{-1}(r), B') \wedge p^\omega \notin Used\_Pred\}$;
        else
          $Derived = Derived \cup \{B' \rightarrow Adn^{-1}(r)\}$;
        end if;
      end for;
  end while;
  Delete from $Derived$ constraints with unadorned heads;
  for each $p(A_1, ..., A_n) \in \mathbf{R}$ do
    $In = In \cup \{p(x_1, ..., x_n) \rightarrow p^{b \ldots b}(x_1, ..., x_n)\}$;
  for each $p(A_1, ..., A_n) \in \mathbf{R}$ do
    for each $p^\alpha(z_1, ..., z_n)$ appearing in $(Base \cup Derived)$ do
      $Out = Out \cup \{p^\alpha(x_1, ..., x_n) \rightarrow \hat{p}(x_1, ..., x_n)\}$;
  return $Base \cup Derived \cup In \cup Out$;
end.
```

Figure 1: Constraint Rewriting Function

In the definition of the function Adn we have also used the function $Adn^{-1}(\cdot)$ which takes in input an adorned first order formula consisting of a conjunction of atoms or a constraint or a set of constraints and gives in output the same formula without adornments. Clearly, for any set of constraints Σ, $Adn^{-1}(Adn(\Sigma)) = \Sigma$.

For any input database schema $\mathbf{R}$ and set of constraints Σ over $\mathbf{R}$, we shall denote with (i) $\hat{\mathbf{R}} = \{\hat{p}(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in \mathbf{R}\}$ the output schema derived from $\mathbf{R}$, (ii) $Map(\mathbf{R}) = \mathbf{R} \cup \hat{\mathbf{R}}$ the union of the input and output schemas, and (iii) $Adn(\mathbf{R}, \Sigma) = \mathbf{R} \cup \{p^\alpha(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in \mathbf{R} \wedge p^\alpha \textit{ appears in } Adn(\Sigma)\} \cup \hat{\mathbf{R}}$ the schema obtained by adding to $\mathbf{R}$ the schemas of the relations introduced in the rewriting of constraints. Moreover, we shall also denote with $Map(\Sigma) = \Sigma \cup \{p(x_1, ..., x_n) \rightarrow \hat{p}(x_1, ..., x_n) | p(A_1, ..., A_n) \in \mathbf{R}\}$ the set of constraints containing, in addition to Σ, a set of TGDs mapping tuples over the input schema to tuples over the output schema.

Example 7. Consider the constraints Σ_4 of Example 4. Initially, $Adn(\Sigma_4)$ contains two constraints derived by adorning the body variables as bound ($Base(\Sigma_4)$)

$$r_1 : \; N^b(x) \rightarrow \exists y\; E^{bf}(x, y)$$
$$r_2 : \; S^b(x) \wedge E^{bb}(x, y) \rightarrow N^b(y)$$

In the second step two new constraints are generated ($Derived(\Sigma_4)$). Due to the new predicate E^{bf}, the following constraint, derived from constraint r_2, has been introduced:

$$r_3: \ S^b(x) \wedge E^{bf}(x,y) \rightarrow N^f(y)$$

At this point the new predicate symbol N^f has been generated and, thus, a new constraint derived from r_1 is added:

$$r_4: \ N^f(x) \rightarrow \exists y \ E^{ff}(x,y)$$

From the new predicate E^{ff} no new constraint is generated since the variable x in the body of the second constraint is bounded as it also appears in the predicate S^b.

Moreover, $Adn(\Sigma)$ also contains TGDs mapping input tuples into "bounded predicates" ($In(\Sigma_4)$):

$$\begin{array}{ll} r_5: & N(x) \rightarrow N^b(x) \\ r_6: & S(x) \rightarrow S^b(x) \\ r_7: & E(x,y) \rightarrow E^{bb}(x,y) \end{array}$$

and TGDs mapping tuples of adorned relations into "output" relations ($Out(\Sigma_4)$):

$$\begin{array}{ll} r_8: & N^b(x) \rightarrow \hat{N}(x) \\ r_9: & N^f(x) \rightarrow \hat{N}(x) \\ r_{10}: & S^b(x) \rightarrow \hat{S}(x) \\ r_{11}: & E^{bb}(x,y) \rightarrow \hat{E}(x,y) \\ r_{12}: & E^{bf}(x,y) \rightarrow \hat{E}(x,y) \\ r_{13}: & E^{ff}(x,y) \rightarrow \hat{E}(x,y) \end{array}$$ □

It is important to observe that the set of constraints Σ_4 is neither stratified nor super-weak acyclic, while $Adn(\Sigma_4)$ is weakly acyclic. In fact, $dep(Adn(\Sigma_4))$, without considering edges in $In(\Sigma_4)$ and $Out(\Sigma_4)$, which do not affect chase termination, contains only the following edges: $N_1^b \rightarrow E_1^{bf}$, $N_1^b \overset{*}{\rightarrow} E_2^{bf}$, $E_2^{bb} \rightarrow N_1^b$, $E_2^{bf} \rightarrow N_1^f$, $N1^f \rightarrow E_1^{ff}$, $N_1^f \overset{*}{\rightarrow} E_2^{ff}$.

FACT 1. *Let Σ be a set of standard TGDs. A position R_i belongs to $aff(\Sigma)$ iff there is some predicate $R_i^{\alpha_1 \dots \alpha_m}$ in $Adn(\Sigma)$ such that $\alpha_i = f$.* □

THEOREM 2. *For every set of TGDs Σ over a database schema $\mathbf{R}$, $\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn(\mathbf{R}, \Sigma), Adn(\Sigma)\rangle$.* □

The previous theorem states that for every database D over a schema $\mathbf{R}$ and for each universal solution J derived by applying the source TGDs Σ to D there is a universal solution K derived by applying the rewritten constraints $Adn(\Sigma)$ to D such that $J[\hat{\mathbf{R}}] = K[\hat{\mathbf{R}}]$ and vice versa. In particular, since $p^\alpha(t) \in K - D$ implies that there is a $p(t) \in J$ and $p(t) \in J$ implies that there is a $p^\alpha(t) \in K$, we have that each relation in J is partitioned into relations of $K - D$. The tuples in D appear twice in K since they are also copied into 'bounded' relations. Note that if the set of constraints $Adn(\Sigma)$ satisfies some chase termination criterion, the chase terminates considering both the source set Σ and the rewritten set $Adn(\Sigma)$. Clearly, the set $Adn(\Sigma)$ is only used to check chase termination conditions (at compile-time), whereas we use the source set Σ to compute (at run-time) universal solutions.

EXAMPLE 7. (CONT.) Consider again the constraints Σ_4 of Example 4 and the source database $D = \{S(a), N(a)\}$. The set of constraints $Map(\Sigma_4)$ is equal to $\Sigma_4 \cup \{S(x) \rightarrow \hat{S}(x), N(x) \rightarrow \hat{N}(x), E(x,y) \rightarrow \hat{E}(x,y)\}$. The application of the chase to $\langle D, Map(\Sigma_4)\rangle$ produces the database $J = \{S(a), N(a), E(a,n_1), N(n_1), E(n_1,n_2), \hat{S}(a), \hat{N}(a), \hat{E}(a,n_1), \hat{N}(n_1), \hat{E}(n_1,n_2)\}$. The application of the chase to $\langle D, Adn(\Sigma)\rangle$ produces the database $K = \{S(a), N(a), S^b(a), N^b(a), N^f(n_1), E^{bf}(a,n_1), E^{ff}(n_1,n_2), \hat{S}(a), \hat{N}(a), \hat{E}(a,n_1), \hat{N}(n_1), \hat{E}(n_1,n_2)$. Clearly, the two solutions are equivalent (with respect to $\{S,N,E\}/\{\hat{S},\hat{N},\hat{E}\}$). □

In the following we shall denote with $Adn\text{-}\mathcal{WA}$ (resp. $Adn\text{-}\mathcal{SC}$, $Adn\text{-}\mathcal{S}tr$, $Adn\text{-}\mathcal{CS}tr$, $Adn\text{-}\mathcal{S}w\mathcal{A}$) the class of TGDs Σ such that $Adn(\Sigma)$ is weakly acyclic (resp., safe, stratified, c-stratified, super-weakly acyclic).

THEOREM 3. *$\mathcal{T} \subsetneq Adn\text{-}\mathcal{T}$ for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{S}w\mathcal{A}\}$.*

The above theorem states that the rewriting technique allows to recognize (by using classical criteria) larger classes of constraints for which chase termination is guaranteed. From Theorem 2 we have that if a set of constraints $\Sigma \in Adn\text{-}\mathcal{WA}$ (resp. $Adn\text{-}\mathcal{SC}$, $Adn\text{-}\mathcal{CS}tr$, $Adn\text{-}\mathcal{S}w\mathcal{A}$) all chase sequences terminate, whereas if $\Sigma \in Adn\text{-}\mathcal{S}tr$ there is at least one terminating chase sequence. Notice that we are considering sets of TGDs, while (c-)stratification also considers EGDs.

It is worth noting that the size of the adorned program increases and in the worst case it is exponential in the size of the source set. Moreover, in practical cases the number of adorned predicates is not much larger than the number of source predicates and the check is a compile time operation.

4. CONSTRAINTS REDUCTION

We now present an extension of our rewriting technique which takes into account the relationship among database values and null values. Consider, for instance, the set of constraints Σ_5' consisting of the constraints r_1, r_2 and r_4 in Example 5. Both Σ_5' and $Adn(\Sigma_5')$ are neither c-stratified nor super-weakly acyclic, however it easy to check that the chase process terminates for all database instances.

In order to improve our rewriting technique, we will use different types of adornments for free variables. Thus, instead of simply using f to denote that a position may contain null values, we will use adornments of the form f_i. Any time a new adorned constraint is added to the set of constraints for each existentially quantified variable a new symbol f_i is introduced, where i is a fresh subscript[5].

Head adornment. Given a (possibly adorned) TGD $r: \phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{z}, \mathbf{y})$ with $\mathbf{z} \subseteq \mathbf{x}$ and let $\phi^\alpha(\mathbf{x})$ be a coherent adornment for the body atoms, then $FHeadAdn(r, \phi^\alpha(\mathbf{x}))$ denotes the adornment of the head of r obtained as follows: i) every universally quantified variable has the same adornment of the body occurrences, ii) every constant is adorned as b; iii) every existentially quantified variable y is adorned with a new symbol f_i. In the following we shall also use substitutions over free symbols. In particular, a substitution θ is a set of pairs f_i/f_j such that $i \neq j$; obviously, the same symbol cannot be used in both left and right sides of substitutions, i.e. a symbol f_j used to replace a symbol f_i cannot be substituted in θ by a symbol f_k.

We shall denote with $Adn^{-s}(\cdot)$ the function which receives in input an adorned first order formula F, where free symbols are subscripted, and gives in output the first order formula derived from F by deleting subscripts, i.e. every symbol f_i is replaced by f. The function can be trivially extended to sets of first order formulae.

The rewriting of constraints is performed by the function Adn^+ reported in Appendix A which differs from the previous version in two main points: (i) the adornment of the head is done by applying the function $FHeadAdn$ and (ii)

[5]Basically, every free variable is adorned with the skolemized function used to replace the existentially quantified variable where each variable of the function is replaced by the corresponding adornment.

a new adorned constraint r^α is added only if there is no adorned constraint r^β having a 'similar' structure, i.e. no substitution θ, defined over the free symbols, exists such that $Body(r^\alpha)\theta = Body(r^\beta)$.

Observe that the function Adn^+ gives as output a set of constraints where subscripts are deleted from adornments. Analogously to the previous rewriting, given a set of TGDs Σ, the rewritten set is denoted by $Adn^+(\Sigma)$ and we have $Adn^+(\Sigma) = Adn^{-s}(Base^+(\Sigma) \cup Derived^+(\Sigma) \cup In^+(\Sigma) \cup Out^+(\Sigma))$, where, $Base^+(\Sigma)$ (resp. $Derived^+(\Sigma), In^+(\Sigma)$, $Out^+(\Sigma)$) is the set of base (resp. derived, input, output) constraints derived by the algorithm before deleting subscripts.

Example 8. Consider the set of constraints Σ'_5 containing the constraints r_1, r_2, r_4 of Example 5. Σ'_5 is neither super-weakly acyclic nor c-stratified. $Base^+(\Sigma'_5)$ consists of the three constraints:

$$\begin{array}{l} r'_1: \ R^b(x) \to S^{bb}(x,x) \\ r'_2: \ S^{bb}(x_1,x_2) \to \exists z\, T^{bf_1}(x_2,z) \\ r'_3: \ T^{bb}(x_1,x_2) \wedge T^{bb}(x_1,x_3) \wedge T^{bb}(x_3,x_1) \to R^b(x_2) \end{array}$$

whereas $Derived^+(\Sigma'_5)$ consists of the three constraints:

$$\begin{array}{l} r'_4: \ T^{bf_1}(x_1,x_2) \wedge T^{bb}(x_1,x_3) \wedge T^{bb}(x_3,x_1) \to R^{f_1}(x_2) \\ r'_5: \ R^{f_1}(x) \to S^{f_1f_1}(x,x) \\ r'_6: \ S^{f_1f_1}(x_1,x_2) \to \exists z\, T^{f_1f_2}(x_2,z) \end{array}$$

The constraint

$$r'': \ T^{f_1f_2}(x_1,x_2) \wedge T^{f_1f_2}(x_1,x_3) \wedge T^{f_1f_2}(x_3,x_1) \to R^{f_2}(x_2)$$

is not added since the body adornment is not coherent, while $Adn(\Sigma'_5)$ also contains the adorned constraint $Adn^{-s}(r'')$. Thus, the set of adorned constraints $Adn^+(\Sigma'_5) = Adn^{-s}(\{r'_1, ..., r'_6\} \cup In^+(\Sigma'_5) \cup Out^+(\Sigma'_5))$ is weakly acyclic, whereas $Adn(\Sigma'_5) = Adn^{-s}(\{r'_1, .., r'_6, r''\} \cup In(\Sigma'_5) \cup Out(\Sigma'_5))$ is neither stratified nor super-weakly acyclic. Moreover, if we consider the set Σ_5 of Example 5, also containing constraint r_3, the function Adn^+ generates (before deleting subscripts), in addition to $r'_1, ..., r'_6$, the constraints:

$$\begin{array}{l} r'_7: \ S^{bb}(x_1,x_2) \to T^{bb}(x_1,x_2), T^{bb}(x_2,x_1) \\ r'_8: \ S^{f_1f_1}(x_1,x_2) \to T^{f_1f_1}(x_1,x_2), T^{f_1f_1}(x_2,x_1) \\ r'_9: \ T^{f_1f_1}(x_1,x_2) \wedge T^{f_1f_1}(x_1,x_3) \wedge T^{f_1f_1}(x_3,x_1) \to R^{f_1}(x_2) \end{array}$$

The resulting set $Adn^+(\Sigma_5)$ (obtained by deleting subscripts) is neither c-stratified nor super-weakly acyclic. □

Lemma 1. *For every set of TGDs Σ the function Adn^+ always terminates.* □

Therefore, the set of adorned predicates in $Adn^+(\Sigma)$ is finite. The following proposition and theorem state that Adn^+ gives in output a subset of constraints generated by the function Adn and that those constraints are 'equivalent' to the original set of constraints.

Proposition 2. *For any set of TGDs Σ over a database schema $\mathbf{R}$, $Adn^+(\Sigma) \subseteq Adn(\Sigma)$.* □

Theorem 4. *For every set of TGDs Σ over a database schema $\mathbf{R}$, $\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn^+(\mathbf{R}, \Sigma), Adn^+(\Sigma)\rangle$.*

Analogously to the previous rewriting, Adn^+-$\mathcal{WA}$ (resp. Adn^+-$\mathcal{SC}$, Adn^+-$\mathcal{S}tr$, Adn^+-$\mathcal{CS}tr$, Adn^+-$\mathcal{S}w\mathcal{A}$) denotes the class of TGDs Σ such that $Adn^+(\Sigma)$ is weakly acyclic (resp., safe, stratified, c-stratified, super-weakly acyclic).

Corollary 2. *Adn-$\mathcal{T} \subsetneq Adn^+$-$\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{S}w\mathcal{A}\}$.* □

5. REFINING AFFECTED POSITIONS

We present now a further improvement to our technique. In particular, we present a new definition of affected positions which restricts the set of positions which should be considered for checking termination conditions and show that it can be used to enhance the rewriting of constraints.

5.1 Weakly and strongly affected positions

The safe condition improves over weak acyclicity as it only considers positions where labeled nulls can be copied and analyzes the data flow among these positions by constructing the propagation graph. Our idea is to statically determine the set of positions where an infinite number of labeled nulls may be copied and distinguish these positions from the remaining ones in the adornment of constraints.

Example 9. Consider the following set of TGDs Σ_9:

$$\begin{array}{l} r_1: N(z) \to F(z,x) \wedge S(x) \wedge E(x,y) \\ r_2: S(x) \wedge E(x,y) \to \exists z\ E(y,z) \end{array}$$

As $aff(\Sigma_9)=\{F_2, S_1, E_1, E_2\}$ and $prop(\Sigma_9)$ contains the edges $E_2 \to E_1$, $E_2 \xrightarrow{*} E_2$ and presents a cycle with special edges. However, although Σ_9 is not safe (nor super-weakly acyclic), all chase sequences are terminating since the number of nulls which may be assigned to affected positions is limited. □

Thus, we now analyze in depth the role of affected positions and introduce the concepts of weak and strong affection. In particular, positions which can take values from $Consts$ and a finite subset of $Nulls$ are called *weakly affected*, whereas the remaining positions are called *strongly affected*. Formally, let Σ be a set of TGDs and EGDs, we denote with $AFF^-(\Sigma)$ the set of *weakly affected* positions in Σ, that is the set of positions which can take value (through the execution of the chase algorithm) from $Consts$ and a finite subset of $Nulls$. The positions in the complementary set $AFF^+(\Sigma) = aff(\Sigma) - AFF^-(\Sigma)$, taking values from $Consts \cup Nulls$, will be called *strongly affected*. Computing the set of position in $AFF^-(\Sigma)$ and $AFF^+(\Sigma)$ is undecidable, but it is possible to statically determine a subset of the position in $AFF^-(\Sigma)$ which are not associated with an infinite number of labeled nulls.

Definition 3. For any set of TGDs Σ, let $aff(\Sigma)$ be the set of affected positions in Σ and *prop*$(\Sigma) = (aff(\Sigma), E)$ the propagation graph, then the set $aff^+(\Sigma)$ is defined as follows. A position R_j in *prop*(Σ) is in $aff^+(\Sigma)$ if:

- it appears in a cycle with special edges, or
- there is a path in *prop*(Σ) from some $S_i \in aff^+(\Sigma)$ to R_j.

The set $aff^-(\Sigma) = aff(\Sigma) - aff^+(\Sigma)$. □

Clearly, $aff^-(\Sigma) \subsetneq AFF^-(\Sigma)$ and $aff^+(\Sigma) \supsetneq AFF^+(\Sigma)$. Considering, for instance, the affected positions from Example 9, we have $F_2, S_1 \in aff^-(\Sigma_9)$ (i.e. they are weakly affected), while $E_1, E_2 \in aff^+(\Sigma_9)$ (i.e. they could be strongly affected). We now present an extension of the rewriting technique presented in the previous section taking into account the type of affected positions.

$FHeadAdn^*(r, \phi^\alpha(\mathbf{x}))$ denotes the adornment of $Head(r)$ derived from $FHeadAdn(r, \phi^\alpha(\mathbf{x}))$ by replacing, for any symbol f_i, associated with some position $R_j \in aff^-(\Sigma)$, all its occurrences with with b_i. The symbol b_i means that the corresponding position takes values from a finite number of null symbols. Since we are now considering two different types of adornments with subscripts, the function Adn^{-s} eliminates subscripts from adornments and, therefore, replaces every adornment f_i (resp. b_i) with f (resp. b). Analogously to the notation previously used, we shall denote with Adn^* the function derived from Adn^+ by replacing the function $FHeadAdn$ with $FHeadAdn^*$ and using substitutions which could also contain pairs b_h/b_k other than pairs f_i/f_j. We shall also denote with $Base^*(\Sigma)$, $Derived^*(\Sigma)$, $In^*(\Sigma)$ and $Out^*(\Sigma)$ the sets of base, derived, input and output constraints computed by Adn^* before deleting subscripts and we have that $Adn^*(\Sigma) = Adn^{-s}(Base^*(\Sigma) \cup Derived^*(\Sigma) \cup In^*(\Sigma) \cup Out^*(\Sigma))$.

As shown in the following example, the rewriting technique benefits from the knowledge that some positions are weakly affected.

Example 10. The application of the function Adn^* to the set of TGDs Σ_9 of Example 9 generates, before deleting subscripts, since attributes S_1 and F_2 are weakly affected, the constraints:

$$\begin{array}{l} r_1: \ N^b(z) \rightarrow \exists x, y \ F^{bb_1}(z,x) \wedge S^{b_1}(x) \wedge E^{b_1 f_2}(x,y) \\ r_2: \ S^b(x) \wedge E^{bb}(x,y) \rightarrow \exists z \ E^{bf_3}(y,z) \\ r_3: \ S^{b_1}(x) \wedge E^{b_1 f_2}(x,y) \rightarrow \exists z \ E^{f_2 f_4}(y,z) \\ r_4: \ S^b(x), E^{bf_3}(x,y) \rightarrow E^{f_3 f_5}(y,z) \end{array}$$

The application of the function Adn^{-s} collapses r_3 and r_4 into a unique constraint. As a result we have that $Adn^*(\Sigma_9)$ is weakly acyclic. □

Theorem 5. *Let Σ be a set of TGDs over a database schema $\boldsymbol{R}$. Then $\langle Map(\boldsymbol{R}), Map(\Sigma)\rangle \equiv_{\boldsymbol{R}/\hat{\boldsymbol{R}}} \langle Adn^*(\boldsymbol{R},\Sigma), Adn^*(\Sigma)\rangle$.*

As for the previous rewriting techniques, Adn^*-$\mathcal{WA}$ (resp. Adn^*-$\mathcal{SC}$, Adn^*-$\mathcal{S}tr$, Adn^*-$\mathcal{CS}tr$, Adn^*-$\mathcal{S}w\mathcal{A}$) denotes the class of TGDs Σ such that $Adn^*(\Sigma)$ is weakly acyclic (resp., safe, stratified, c-stratified, super-weakly acyclic).

Theorem 6. *Adn^+-$\mathcal{T} \subsetneq Adn^*$-$\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{S}w\mathcal{A}\}$.* □

5.2 Adding EGDs

In some cases it is interesting to check if there is a terminating chase sequence and understand how constraints should be fired to compute finite sequences. We propose a new criterion which in the presence of equality conditions allows to identify classes of chase-terminating constraints not captured by previous conditions (even if constraints are adorned).

Proposition 3. *Given a set of constraints Σ, a relation schema $r(A_1, \ldots, A_n)$ and a functional dependency $f : A_{i_1}...A_{i_k} \rightarrow A_j$ with $k \geq 0$, if $A_{i_1}...A_{i_k}$ are in $AFF^-(\Sigma)$, then A_j is in $AFF^-(\Sigma)$ as well.* □

Thus, in the following, we introduce a new class of positions $Aff^-(\Sigma)$ which could be statically determined to be weakly affected. More specifically, $Aff^-(\Sigma)$ may be obtained by initially setting $Aff^-(\Sigma) = aff^-(\Sigma)$ and iteratively adding to $Aff^-(\Sigma)$ nodes A_j such that there is a functional dependency $A_i, ..., A_k \rightarrow A_j$ with $A_i, ...A_k \in Aff^-(\Sigma)$, while $Aff^+(\Sigma) = aff(\Sigma) - Aff^-(\Sigma)$. Clearly, $aff^-(\Sigma) \subsetneq Aff^-(\Sigma) \subsetneq AFF^-(\Sigma)$. As a result, the class of safe constraints can be extended by considering, in the construction of the propagation graph, positions in $Aff^+(\Sigma)$ instead of general affected ones. For these classes of constraints there is at least a terminating chase sequence, that is the sequence giving preference to the application of EGDs.

6. CONCLUSIONS

This paper has presented a technique for checking chase termination based on constraints rewriting. We have shown that significant classes of constraints for which the chase execution terminates can be captured by classical criteria after that constraints have been adorned. The rewriting technique is orthogonal to termination conditions and improves current chase termination criteria. Further investigation should be devoted to improve the rewriting technique by also considering EGDs and identify larger sets of weakly affected positions by analyzing other special classes of constraints, e.g. embedded multi-valued dependencies [9].

Acknowledgments. We thank Irina Trubitsyna for useful suggestions and the reviewers for their thorough reviews and helpful comments.

7. REFERENCES

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

[2] C. Beeri and R. Ramakrishnan. On the power of magic. *J. Log. Program.*, 10(1/2/3&4):255–299, 1991.

[3] C. Beeri and M. Y. Vardi. Formal systems for tuple and equality generating dependencies. *SIAM J. Comput.*, 13(1):76–98, 1984.

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

[5] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Description Logics*, 2008.

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

[7] G. DeGiacomo, D. Lembo, M. Lenzerini, and R. Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS*, pages 133–142, 2007.

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

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

[10] 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.

[11] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.

[12] P. G. Kolaitis, J. Panttaja, and W. C. Tan. The complexity of data exchange. In *PODS*, pages 30–39, 2006.

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

[14] B. Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, pages 13–22, 2009.

[15] M. Meier, M. Schmidt, and G. Lausen. On chase termination beyond stratification. *PVLDB*, 2(1):970–981, 2009.

[16] M. Meier, M. Schmidt, and G. Lausen. On chase termination beyond stratification. *CoRR*, abs/0906.4228, 2009.

APPENDIX

A. FUNCTION ADN^+

```
Function $Adn^+(\Sigma)$;
Input Set of TGDs $\Sigma$ over a schema $\mathbf{R}$;
Output Set of (adorned) TGDs $Base^+ \cup Derived^+ \cup In^+ \cup Out^+$;
begin
  $Base^+ = Derived^+ = In^+ = Out^+ = New\_Pred = \emptyset$;
   // Let $Body^b(r)$ be the conjunction obtained by adorning
   // atoms in $Body(r)$ with strings of $b$ symbols
   $Used\_Pred = \bigcup_{r \in \Sigma} Body^b(r)$
  for each $r \in \Sigma$ do begin
    $Base = Base \cup \{Body^b(r) \rightarrow FHeadAdn(r, Body^\alpha(r))\}$;
    $New\_Pred = New\_Pred \cup \{p^\alpha | p^\alpha(t) \in FHeadAdn(r, Body(r))\}$;
  end for;
  while $(New\_Pred \neq \emptyset)$ do begin
    Select nondeterministically $p^{\alpha_1 .. \alpha_n} \in New\_Pred$
    $New\_Pred = New\_Pred - \{p^{\alpha_1 .. \alpha_n}\}$;
    $Used\_Pred = Used\_Pred \cup \{p^{\alpha_1 .. \alpha_n}\}$;
    for each $r \in (Base \cup Derived^+)$ do
      for each $p^\beta(x_1, ..., x_n) \in Body(r)$ do begin
        $B' = Body(r) - \{p^\beta(x_1, ..., x_n)\} \cup \{p^{\gamma_1 .. \gamma_n}(x_1, ..., x_n)\}$;
        // $\gamma_i = b$ if $x_i \in Consts$;  $\gamma_i = \alpha_i$ if $x_i \in Vars$; $(i \in [1..n])$
        if $B'$ is coherent and $\nexists$ (subst. $\theta$ and $r^\beta \in Derived^+$)
          s.t. $B'\theta = Body(r^\beta)$ then
            Let $H' = FHeadAdn(Adn^{-1}(r), B')$;
            $Derived^+ = Derived^+ \cup \{B' \rightarrow H'\}$;
            $New\_Pred = New\_Pred \cup \{p^\omega | p^\omega(t) \in H'$
                                                  $\wedge\ p^\omega \notin Used\_Pred)$;
        else
            $Derived = Derived \cup \{B' \rightarrow Adn^{-1}(r)\}$;
        end if;
      end for;
  end while;
  Delete from $Derived$ constraints with unadorned heads;
  for each $p(A_1, ..., A_n) \in \mathbf{R}$ do
    $In^+ = In^+ \cup \{p(x_1, ..., x_n) \rightarrow p^{b...b}(x_1, ..., x_n)\}$;
  for each $p(A_1, ..., A_n) \in \mathbf{R}$ do
    for each $p^\alpha(z_1, .., z_n)$ appearing in $Base^+ \cup Derived^+$ do
      $Out^+ = Out^+ \cup \{p^\alpha(x_1, ..., x_n) \rightarrow \hat{p}(x_1, ..., x_n)\}$;
    return $Adn^{-s}(Base^+ \cup Derived^+ \cup In^+ \cup Out^+)$;
end.
```

Figure 2: Constraint Rewriting Function (version 2)

B. CHASE TERMINATION CONDITIONS

Oblivious Chase. Intuitively, an oblivious chase step always applies when the body of a constraint can be mapped to an instance, even if the constraint is satisfied.

More formally, an oblivious chase step for a TGD $r : \phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})$ applies to an instance K if there is a homomorphism h from $\phi(\mathbf{x})$ to K. In such a case the oblivious chase step $K \overset{*,r,h}{\rightarrow} K'$ is defined as follows. Let h' be a homomorphism s.t. (a) h' agrees with h on $\mathbf{x}$, (b) for every existentially quantified variable $y_j \in \mathbf{y}$ we choose a fresh labeled null n_i and define $h'(y_j) = n_i$. We set K' to be $K \cup \{h'(\psi(\mathbf{x}, \mathbf{y}))\}$. An oblivious chase step for an EGD is a chase step for an EGD except that we also add an $*$ on the arrow (like in the case of TGDs) that indicates the step.

B.1 Weak Acyclicity

The first and basic criterion concerning the identification of sufficient conditions, determined by the structure of TGDs, guaranteeing chase termination, is known as weak acyclicity and was given in [10]. The criterion is based on the structural properties of a graph $G(\Sigma)$, called dependency graph, derived from the input set of TGDs Σ.

Definition 4 (Weakly acyclic set of TGDs [10]). Let Σ be a set of TGDs over a fixed schema. Construct a directed graph $dep(\Sigma) = (pos(\Sigma), E)$, called the dependency graph, as follows: (1) there is a node R_i for every pair (R, A) with R a relation symbol of the schema and A an attribute of R in position i; (2) add edges as follows: for every TGD $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})$ in Σ and for every x in $\mathbf{x}$ that occurs both in ϕ (in position R_i) and in ψ:

1. for every occurrence of x in ψ in position S_j, add an edge $R_i \rightarrow S_j$ (if it does not already exist).
2. for every existentially quantified variable y and for every occurrence of y in ψ in position T_k, add a special edge $R_i \xrightarrow{*} T_k$ (if it does not already exist).

Then, Σ is *weakly acyclic* if $dep(\Sigma)$ has no cycle going through a special edge. ☐

Example 11. Consider the sets of constraints of Examples 1 and 2. The dependency graphs contain in both cases a cycle with special edges and, therefore, both Σ_1 and Σ_2 are not weakly acyclic. ☐

Clearly, the problem of checking whether a set of TGDs is weakly acyclic is polynomial in the size of $|\Sigma|$. In [10] it has been shown that if Σ is the union of a weakly acyclic set of TGDs with a set of EGDs, and D is a database instance, then there exists a polynomial in the size of D that bounds the length of every chase sequence of D with Σ.

B.2 Stratification

The idea behind stratification is to decompose the set of constraints into independent subsets, where each subset consists of constraints that may fire each other, and to check each component separately for weak acyclicity.

Definition 5 (Precedence relation [8, 16]). Given a set of constraints Σ and two constraints $r_1, r_2 \in \Sigma$, we say that $r_1 \prec r_2$ iff there exists a relational database instance D and two tuples of values δ_1 and δ_2 such that (i) $D \not\models r_1(\delta_1)$, (ii) $D \models r_2(\delta_2)$, (iii) $D \overset{r_1,\delta_1}{\rightarrow} J$ and (iv) $J \not\models r_2(\delta_2)$. ☐

$r_1 \prec r_2$ means that firing r_1 can cause the firing of r_2.

Definition 6 (Stratified constraints [8, 16]). The chase graph $G(\Sigma) = (\Sigma, E)$ of a set of constraints Σ contains a directed edge (r_1, r_2) between two constraints iff $r_1 \prec r_2$. We say that Σ is stratified iff the constraints in every cycle of $G(\Sigma)$ are weakly acyclic. ☐

Example 12. [8] Consider the constraint

$$r :\ E(x, y) \wedge E(y, x) \rightarrow \exists z, w\ E(x, z) \wedge E(z, w) \wedge E(w, x)$$

stating that each node having a cycle of length 2 also has a cycle of length 3. Since $r \not\prec r$ (i.e. r does not fire itself) $G(\{r\})$ is acyclic and, therefore, $\{r\}$ is stratified. ☐

In [8] it has been shown that the problem of deciding whether a set of constraints is stratified is in $co\mathcal{NP}$ and that stratification strictly generalizes weak acyclicity.

C-Stratification. Stratification guarantees, as shown in [16], that, for every database D, there is a chase sequence (but not all) which terminates in polynomial time in the size of D (see Example 5).

In order to cope with this problem, a variation of stratification, called *c-stratification*, has been proposed in [16].

Given two constraints $r_1, r_2 \in \Sigma$, we say that $r_1 \prec_c r_2$ iff there exists a relational database instance D and two tuples of values δ_1 and δ_2 such that (i) $D \not\models r_1(\delta_1)$, (ii) $D \models r_2(\delta_2)$, (iii) $D \overset{*,r_1,\delta_1}{\rightarrow} J$, and (iv) $J \not\models r_2(\delta_2)$.

The c-chase graph $G_c(\Sigma) = (V, E)$ of a set of constraints Σ contains a directed edge (r_1, r_2) between two constraints iff $r_1 \prec_c r_2$. We say that Σ is c-stratified iff the constraints in every cycle of $G_c(\Sigma)$ are weakly acyclic.

The problem of checking whether a set of constraints is c-stratified is in $co\mathcal{NP}$(as well as stratification). For every set of c-stratified constraints Σ and for every database instance D, there exists a polynomial in the size of D that bounds the length of every chase sequence of D with Σ [16].

B.3 Safety

A different extension of weak acyclicity which takes into account only affected positions has been proposed in [15]. An affected position denotes an position which could be associated with null values, that is it can also take values from *Nulls*.

Definition 7 (Affected positions). [5]. Let Σ be a set TGDs. The set of affected positions $aff(\Sigma)$ of Σ is defined as follows. Let π be a position occurring in the head of some TGD $r \in \Sigma$, then

- if an existentially quantified variable appears in π, then $\pi \in aff(\Sigma)$;
- if the same universally quantified variable x appears both in position π and only in affected positions in the body of r, then $\pi \in aff(\Sigma)$. □

Definition 8 (Safe set of TGDs [15]). Let Σ be a set of TGDs, then $prop(\Sigma) = (aff(\Sigma), E)$ denotes the propagation graph of Σ defined as follows. For every TDG $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ and for every x in $\mathbf{x}$ that occurs in ψ and every occurrence of x in ϕ in position π_1 then

- if x occurs only in affected positions in ϕ then for every occurrence of x in ψ in position π_2 there is an edge $\pi_1 \rightarrow \pi_2$ in E;
- if x occurs only in affected positions in ϕ then, for every y in $\mathbf{y}$ and for every occurrence of y in ψ in position π_2 there is a special edge $\pi_1 \overset{*}{\rightarrow} \pi_2$ in E.

A set of constraints Σ is said to be *safe* if $prop(\Sigma)$ has no cycles going through a special edge. □

Clearly, safety strictly generalizes weak acyclicity and the problem of checking whether a set of TGDs is safe is polynomial in the size of $|\Sigma|$. Moreover, for every Σ being the union of a safe set of TGDs with a set of EGDs, and for every database instance D, then there exists a polynomial in the size of D that bounds the length of every chase sequence of D with Σ.

A generalization of safety and c-stratification, called *Inductive Restriction*, has been proposed in [15]. The problem of checking whether a set of constraints is inductively restricted is in $co\mathcal{NP}$.

B.4 Super–weak Acyclicity

We now recall some basic definitions about super-weak acyclicity [14], a condition that guarantees the termination of oblivious chase. Let Σ be a set of TGDs and $P(\Sigma)$ the logic program obtained by skolemizing Σ. A place is a pair (a, i) where a is an atom occurring in a rule of $P(\Sigma)$ and $0 \leq i \leq arity(a)$. Given a TGD r and an existential variable y in the head of r, we define $Out(r, y)$ as the set of places (called *output places*) in the head of $P(r)$ where a term of the form $f_y^r(\ldots)$ occurs. Given a TGD r' and a universal variable x' of r', $In(r', x')$ denotes the set of places (called *input places*) in the body of r' where x' occurs.

Example 13. Consider the below set of TGDs Σ_{13}:

$$r_1 : \forall x\ [\ N(x) \rightarrow \exists y, z\ E(x, y, z)\ ]$$
$$r_2 : \forall (x, y)\ [\ E(x, y, y) \rightarrow N(y)\ ]$$

The logic program obtained by skolemizing Σ_{13} is:

$$P(\Sigma_{13}) = \begin{cases} r_1' : S(\underset{p_1}{x}) \rightarrow E(\underset{p_2}{x}, f_y^{r_1}(\underset{p_3}{x}), f_z^{r_1}(\underset{p_4}{x})) \\ r_2' : E(\underset{p_5}{x}, \underset{p_6}{y}, \underset{p_7}{y}) \rightarrow S(\underset{p_8}{y}) \end{cases}$$

and $Out(r_1, y) = \{p_3\}$, $Out(r_1, z) = \{p_4\}$, $In(r_2, y) = \{p_6, p_7\}$. □

Given a set of variables V, a substitution θ of V is a function mapping each $v \in V$ to a finite term $\theta(v)$ built upon constants and function symbols. Two places (a, i) and (a', i) are unifiable and we write $(a, i) \sim (a', i)$ iff there exist two substitutions θ and θ' of (respectively) the variables a and a' such that $a[\theta] = a'[\theta']$. Given two sets of places Q and Q' we write $Q \sqsubseteq Q'$ iff for all $q \in Q$ there exists some $q' \in Q'$ such that $q \sim q'$. Given a set Q of places, we define $Move(\Sigma, Q)$ as the smallest set of places Q' such that $Q \subseteq Q'$, and for every rule $r = B_r \rightarrow H_r$ in $P(\Sigma)$ and every variable x, if $\Gamma_x(B_r) \sqsubseteq Q'$ then $\Gamma_x(H_r) \subseteq Q'$, where $\Gamma_x(B_r)$ and $\Gamma_x(H_r)$ denote the sets of places in B_r and H_r where x occurs.

Definition 9. (Trigger relation and Super-weak Acyclicity [14]). Given a set Σ of TGDs and two TGDs $r, r' \in \Sigma$, we say that r triggers r' in Σ and we write $r \rightsquigarrow_\Sigma r'$ iff there exists an existential variable y in the head of r, and a universal variable x' occurring both in the body and head of r' such that $In(r'x') \sqsubseteq Move(\Sigma, Out(r, y))$. A set of constraints Σ is super-weakly acyclic iff the trigger $\rightsquigarrow_\Sigma$ relation is acyclic. □

Example 13 (cont.) Since $Move(\Sigma_{13}, Out(r_1, y)) = \{p_3\}$, $Move(\Sigma_{13}, Out(r_1, z)) = \{p_4\}$ and $In(r_2, y)) = \{p_6, p_7\}$, we have that $In(r_2, y) \not\sqsubseteq Move(\Sigma_{13}, Out(r_1, y))$ and $In(r_2, y) \not\sqsubseteq Move(\Sigma_{13}, Out(r_1, z))$. Consequently, r_2 is not triggered by r_1 (as well as $r_1 \not\rightsquigarrow_{\Sigma_{13}} r_1$) and, therefore, Σ_{13} is super-weakly acyclic. □

Moreover, it has been proved in [14] that the problem of deciding whether a set of constraints is super-weakly acyclic is in PTIME.

C. PROOFS

This appendix presents some of the proofs of the results introduced in the paper. For space limitation proofs (requiring large space) are omitted or just sketched. Complete proofs of the results reported here will appear in the full version of the paper.

PROPOSITION 1 $\mathcal{SwA} \nparallel \mathcal{CStr}$.

PROOF. The set of constraints of Example 12 is (c-)stratified, but not super-weakly acyclic, whereas the set of constraints of Example 3 is super-weakly acyclic, but not stratified, therefore, the two criteria are not comparable. $\square$

COROLLARY 1 $\mathcal{SwA} \nparallel \mathcal{IR}$.

PROOF. Since the set of constraints of Example 3 is not inductively restricted we also have that $\mathcal{SwA} \nparallel \mathcal{IR}$. $\square$

For any place $p_i = (A(x_1, ...x_n), i)$ where $1 \leq i \leq n$, $\Pi(p_i)$ denotes the corresponding position A_i. Analogously, for a given set of places Q, $\Pi(Q) = \{\Pi(j) | j \in Q\}$ denotes the set of positions associated with the places in Q.

LEMMA 2. *Let Σ be a set of TGDs and y an existentially quantified variable appearing in a constraint $r \in \Sigma$. Then,* $\Pi(Move(\Sigma, Out(r, y))) \subseteq aff(\Sigma)$.

PROOF. (Base case): $\Pi(Out(r, y)) \subseteq aff(\Sigma)$ since the variable y is existentially quantified.
(Inductive case): $Move(\Sigma, Out(r, y))$ contains, other than places in $Out(r, y)$, places associated with some head, universally quantified variable x whose body instance place unifies with some with some place in $Move(\Sigma, Out(r, y))$. By definition of affected position, the position of x in the head of the constraint is affected as well. $\square$

THEOREM 1 $\mathcal{SC} \subsetneq \mathcal{SwA}$.

PROOF. If $r \rightsquigarrow_\Sigma r'$ then there exist an existentially quantified variable y in the head of r and a universally quantified variable x appearing in both the body and head of r' s.t. $In(r'x) \sqsubseteq Move(\Sigma, Out(r, y))$. Observe that in this case we have that $\Pi(In(r', x)) \subseteq \Pi(Move(\Sigma, Out(r, y)))$. The corresponding subgraph in $prop(\Sigma)$ contains as nodes $\Pi(Move(\Sigma, Out(r, y)))$ and positions where existentially quantified variables appear in the head of r'. Among positions in $\Pi(Move(\Sigma,\ Out(r, y)))$ there are only normal edges according to how a null value is copied from $\Pi(Out(r, y))$ to other positions in $\Pi(Move(\Sigma, Out(r, y)))$. Special edges appear only from positions in $\Pi(In(r', x))$ to positions where existentially quantified variables appear in the head of r'. Thus we have that if $r \rightsquigarrow_\Sigma r'$ then the corresponding subgraph in $prop(\Sigma)$ is acyclic w.r.t. special edges. From this fact it follows that whenever the trigger graph is cyclic, the graph $prop(\Sigma)$ has a cycle going through special edges, too. $\square$

THEOREM 3 $\mathcal{T} \subsetneq Adn\text{-}\mathcal{T}$ for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{SwA}\}$.

PROOF. Assume that Σ is in $\mathcal{T}$ and is not in $Adn\text{-}\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC},\ \mathcal{S}tr, \mathcal{CS}tr, \mathcal{SwA}\}$. This means that $Base(\Sigma) \cup Derived(\Sigma)$ is not in $\mathcal{T}$, since $In(\Sigma)$ and $Out(\Sigma)$ are acyclic and do no affect termination conditions. However, as $Adn^{-1}(Base(\Sigma)\ \cup Derived(\Sigma)) = \Sigma$, we have that Σ is not in $\mathcal{T}$ as well (which contradicts the hypothesis). In addition, $\mathcal{T} \subsetneq Adn\text{-}\mathcal{T}$ since there are sets of TGDs Σ such that Σ is in $Adn\text{-}\mathcal{T}$, but not in $\mathcal{T}$ (see, for instance, Example 7). $\square$

LEMMA 1 For every set of TGDs Σ the function Adn^+ always terminates.

PROOF. *(Sketch)* In the function $Adn^+(\Sigma)$ a new constraint $B' \rightarrow H'$ is added to the set $Derived^+$ only if for all $r^\beta \in Derived^+$, B' and $Body(r^\beta)$ are *renaming independent*, that is there not exist a substitution θ such that $B'\theta = Body(r^\beta)$. This means that the set $Derived^+$ cannot contain two constraints whose bodies coincide if we rename subscripts. Consequently, since the number of possible renaming independent body adornments for a constraint r is finite, the set of constraints in $Derived^+(\Sigma)$ is finite as well. $\square$

PROPOSITION 2 For any set of TGDs Σ over a database schema $\mathbf{R}$, $Adn^+(\Sigma) \subseteq Adn(\Sigma)$.

PROOF. *(Sketch)* Clearly $In^+(\Sigma) = In(\Sigma)$ and $Adn^{-s}(Base^+(\Sigma)) = Base(\Sigma)$ because in the rewriting constraints we adorn existentially quantified variables with symbols of the form f_i instead of f and subscripts are next deleted.
$Adn^{-s}(Derived^+(\Sigma)) \subseteq Derived(\Sigma)$ because after deleting subscripts we obtain constraints which are also generated by the function $Adn(\Sigma)$. Moreover, since $Adn^{-s}(Derived^+(\Sigma)) \subseteq Derived(\Sigma)$ and $Adn^{-s}(Base^+(\Sigma)) = Base(\Sigma)$, we have that $Adn^{-s}(Out^+(\Sigma)) \subseteq Out(\Sigma)$. $\square$

COROLLARY 2 $\mathcal{T} \subsetneq Adn\text{-}\mathcal{T} \subsetneq Adn^+\text{-}\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{SwA}\}$.

PROOF. $Adn\text{-}\mathcal{T} \subseteq Adn^+\text{-}\mathcal{T}$ derives from Proposition 2. Moreover, $Adn\text{-}\mathcal{T} \subsetneq Adn^+\text{-}\mathcal{T}$ since there are sets of TGDs Σ s.t. Σ in $Adn^+\text{-}\mathcal{T}$, but not in $Adn\text{-}\mathcal{T}$ (see for instance Example 8). $\square$

THEOREM 6 $Adn^+\text{-}\mathcal{T} \subsetneq Adn^*\text{-}\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{SwA}\}$.

PROOF. *(Sketch)* Let Σ be a set of TGDs. It is possible to define a mapping $\mathcal{M}$ from $Adn^+(\Sigma)$ to $Adn^*(\Sigma)$ which is non-injective and surjective. $\mathcal{M}$ maps constraints in $Adn^+(\Sigma)$ to constraints in $Adn^*(\Sigma)$ by simply rewriting adornments of predicate symbols (e.g. a predicate symbol p^α occurring in a constraint $r \in Adn^+(\Sigma)$ is replaced by a predicate symbol p^β by substituting some $\alpha_i = f$ with b. Clearly, if $Adn^*(\Sigma)$ does not satisfy the structural properties provided by $\mathcal{T}$, $Adn^+(\Sigma)$ does not satisfy these properties as well (recall that $\mathcal{M}$ is surjective and just replaces adornments of predicates). Consequently, $Adn^*\text{-}\mathcal{T}$ is more general than $Adn^+\text{-}\mathcal{T}$. $\square$

PROPOSITION 3 Given a set of constraints Σ, a relation schema $r(A_1, \ldots, A_n)$ and a functional dependency $f : A_{i_1}...A_{i_k} \rightarrow A_j$ with $k \geq 0$, if $A_{i_1}...A_{i_k}$ are in $AFF^-(\Sigma)$, then A_j is in $AFF^-(\Sigma)$ as well.

PROOF. By contradiction. If there is an infinite instance with an infinite number of labeled nulls in position A_j, then there must be some position in $A_{i_1}...A_{i_k}$ with an infinite number of distinct values (i.e. distinct labeled nulls), otherwise the functional dependency is not satisfied. $\square$

D. NOTATION

Symbols	Description	Used in
Σ	Set of source constraints	
$\mathbf{R}$	Database schema (input schema)	
Σ^α	Set of adorned constraints	
$\hat{\mathbf{R}}$	Output schema derived from the input schema $\mathbf{R}$	
$dep(\Sigma)$	Dependency graph of Σ	WA
$aff(\Sigma)$	Set of affected positions in Σ	SC
$prop(\Sigma)$	Propagation graph of Σ	SC
$AFF^-(\Sigma)$	Weakly affected positions	
$AFF^+(\Sigma)$	Strongly affected positions	
$aff^+(\Sigma)$	Affected positions belonging to or depending on cycles with special arcs	Adn^*
$aff^-(\Sigma)$	$aff(\Sigma) - aff^+(\Sigma)$	Adn^*
$Aff^-(\Sigma)$	Weakly affected positions in Σ statically determined by also considering FDs	
$Aff^+(\Sigma)$	$aff(\Sigma) - Aff^-(\Sigma)$	

Classes of constraints	
$\mathcal{WA}$	Class of weakly acyclic sets of constraints
$\mathcal{SC}$	Class of safe sets of constraints
$\mathcal{S}tr$	Class of stratified sets of constraints
$\mathcal{CS}tr$	Class of c-stratified sets of constraints
$\mathcal{S}w\mathcal{A}$	Class of super-weakly acyclic sets of constraints
Adn-$\mathcal{WA}$	Class of sets of constraints Σ such that $Adn(\Sigma)$ is weakly acyclic
Adn-$\mathcal{SC}$	Class of sets of constraints Σ such that $Adn(\Sigma)$ is safe
Adn-$\mathcal{S}tr$	Class of sets of constraints Σ such that $Adn(\Sigma)$ is stratified
Adn-$\mathcal{CS}tr$	Class of sets of constraints Σ such that $Adn(\Sigma)$ is c-stratified
Adn-$\mathcal{S}w\mathcal{A}$	Class of sets of constraints Σ such that $Adn(\Sigma)$ is super-weakly acyclic
Adn^+-$\mathcal{WA}$	Class of sets of constraints Σ such that $Adn^+(\Sigma)$ is weakly acyclic
Adn^+-$\mathcal{SC}$	Class of sets of constraints Σ such that $Adn^+(\Sigma)$ is safe
Adn^+-$\mathcal{S}tr$	Class of sets of constraints Σ such that $Adn^+(\Sigma)$ is stratified
Adn^+-$\mathcal{CS}tr$	Class of sets of constraints Σ such that $Adn^+(\Sigma)$ is c-stratified
Adn^+-$\mathcal{S}w\mathcal{A}$	Class of sets of constraints Σ such that $Adn^+(\Sigma)$ is super-weakly acyclic
Adn^*-$\mathcal{WA}$	Class of sets of constraints Σ such that $Adn^*(\Sigma)$ is weakly acyclic
Adn^*-$\mathcal{SC}$	Class of sets of constraints Σ such that $Adn^*(\Sigma)$ is safe
Adn^*-$\mathcal{S}tr$	Class of sets of constraints Σ such that $Adn^*(\Sigma)$ is stratified
Adn^*-$\mathcal{CS}tr$	Class of sets of constraints Σ such that $Adn^*(\Sigma)$ is c-stratified
Adn^*-$\mathcal{S}w\mathcal{A}$	Class of sets of constraints Σ such that $Adn^*(\Sigma)$ is super-weakly acyclic

Functions & Symbols	
$HeadAdn(B \rightarrow H, B^\alpha)$	Adornment of the head H of the TGD $B \rightarrow H$ w.r.t. the body adornment α
$Adn(\Sigma)$	Set of adorned constraints derived from Σ by applying the function Adn of Figure 1
$Base(\Sigma)$	Set of constraints derived by adorning the body variable as bound
$Derived(\Sigma)$	Set of adorned constraints derived by using the new predicate in the heads of $Base(\Sigma)$ constraints
$In(\Sigma)$	TGDs mapping input predicates into bounded predicates
$Out(\Sigma)$	TGDs mapping adorned predicates into output predicates
$Adn^{\text{-}1}(\Sigma^\alpha)$	Set of unadorned constraints derived from Σ^α by deleting adornments
$Adn(\mathbf{R}, \Sigma)$	Set of input, output and adorned predicates
$Map(\Sigma)$	Σ plus set of TGDs mapping tuples over the input schema to tuples over the output schema
$Map(\mathbf{R})$	Union of the input ($\mathbf{R}$) and output ($\hat{\mathbf{R}}$) schema
$FHeadAdn(B \rightarrow H, B^\alpha)$	As $HeadAdn$, but with the use of subscripts (f_i) for free adornments
$Adn^{\text{-}s}(\cdot)$	Function that remove subscripts from f (possibly b) adornments
$Adn^+(\Sigma)$	Set of adorned constraints derived from Σ by applying the function Adn^+ of Figure 2
$FHeadAdn^*(B \rightarrow H, B^\alpha)$	As $FHeadAdn$, but with the use of adornment b_i instead of f_i for weakly affected positions aff^-
$Adn^*(\Sigma)$	Set of adorned constraints derived from Σ by applying the function Adn^*