Stratification Criteria and Rewriting Techniques for Checking Chase Termination

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

Irina Trubitsyna
DEIS, Università della Calabria
87036 Rende (Cs), Italy
irina@deis.unical.it

ABSTRACT

The Chase is a fixpoint algorithm enforcing satisfaction of data dependencies in databases. Its execution involves the insertion of tuples with possible null values and the changing of null values which can be made equal to constants or other null values. Since the chase fixpoint evaluation could be non-terminating, in recent years the problem know as chase termination has been investigated. It consists in the detection of sufficient conditions, derived from the structural analysis of dependencies, guaranteeing that the chase fixpoint terminates independently from the database instance. Several criteria introducing sufficient conditions for chase termination have been recently proposed [9, 8, 13, 12].

The aim of this paper is to present more general criteria and techniques for chase termination. We first present extensions of the well-known stratification conditions and introduce a new criterion, called local stratification (LS), which generalizes both super-weak acyclicity and stratification-based criteria (including the class of constraints which are inductively restricted). Next the paper presents a rewriting algorithm, whose structure is similar to the one presented in [10]; the algorithm takes as input a set of tuple generating dependencies and produces as output an equivalent set of dependencies and a boolean value stating whether a sort of cyclicity has been detected. The output set, obtained by adorning the input set of constraints, allows us to perform a more accurate analysis of the structural properties of constraints and to further enlarge the class of tuple generating dependencies for which chase termination is guaranteed, whereas the checking of acyclicity allows us to introduce the class of acyclic constraints ($\mathcal{AC}$), which generalizes $\mathcal{LS}$ and guarantees chase termination.

Keywords: Data Exchange, Data Integration, Chase.

1. INTRODUCTION

The Chase is a fixpoint algorithm enforcing satisfaction of data dependencies in databases. It has been proposed more than thirty years ago [1, 11] and has received an increasing attention in recent years in both database theory and practical applications. Indeed, the availability of data coming from different sources easily results in inconsistent or incomplete data (i.e. data not satisfying data dependencies) and, therefore, techniques for fixing inconsistencies are crucial [2, 6]. The chase algorithm is used, directly or indirectly, on an everyday basis by people who design databases, and it is used in commercial systems to reason about the consistency and correctness of a data design. New applications of the chase in meta-data management, ontological reasoning, data exchange and data cleaning have been proposed as well [3, 5, 7].

The execution of the chase algorithm involves the insertion of tuples with possible null values and the changing of null values which can be made equal to constants or other null values. However, the insertion of tuples with new (null) values could result in a non-terminating execution. The following example shows a case where a given database does not satisfy a set of data dependencies (also called constraints) and the application of the chase algorithm produces a new consistent database by adding tuples with nulls.

EXAMPLE 1. Consider the set of constraints Σ_1:

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

where the relations N and S store normal nodes and special nodes, respectively, whereas E stores edges. The second constraint states that if there exists an edge from x to y and x is a special node, then y must be a (normal) node. The first constraint states that every normal node must have an outgoing edge. Assume that the database contains the tuples $S(a), N(a)$. Since the first constraint is not satisfied, the tuple $E(a, n_1)$, where n_1 is a new labeled null, 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. The output database is consistent as both dependencies are satisfied. □

However, it is important to observe that if we delete from the second constraint the predicate $S(x)$, the chase never terminates and adds to the database an infinite number of tuples.

The problem recently investigated, known as *chase termination*, consists in the identification of sufficient conditions, based on structural properties of the input set of constraints, guaranteeing that the chase fixpoint terminates independently from the database instance. Several criteria for chase termination have been defined. We recall here *weak acyclicity* [9], *safety* [13], *(c-)stratification* [8, 14], *super-weak acyclicity* [12], *safe restriction* and *inductive restriction* [14]. An orthogonal technique which enlarges the classes of dependencies recognized by these criteria has been proposed in [10]. This technique rewrites the input set of constraints Σ into an equivalent set Σ^α and checks criteria satisfaction on Σ^α.

Nevertheless, despite the previously mentioned results, there are still important classes of terminating data dependencies which are

Proceedings of the VLDB Endowment, Vol. 4, No. 11

not identified by none of the previous mentioned criteria. Thus, the aim of this work is the definition of more general criteria and techniques for chase termination.

Contributions. The main contributions of this paper follow.

- We start by analyzing the stratification criterion and propose a new stratification technique, called *WA-stratification*, which builds a different chase graph $\Gamma(\Sigma)$, called *firing* graph, and overcomes some limitations of (c-)stratification; since WA-stratification checks weak acyclicity over strong components of $\Gamma(\Sigma)$, two further, more powerful, conditions checking safety and super-weak acyclicity over $\Gamma(\Sigma)$ are defined. These techniques are called *SC-stratification* and *SwA-stratification*.
- SwA-stratification, the most general of the above new criteria, is not comparable with inductive restriction (*IR*). Thus, we propose a further criterion, called *Local stratification* (LS) which uses a more refined criterion of SwA during the construction of the firing graph. LS generalizes SwA-stratification and is more powerful than *IR*.
- We present a new rewriting technique which produces an equivalent set of dependencies and a boolean whose value is true if during the rewriting technique a form of cyclicity has been detected. The rewriting technique allows us, by checking criteria on the rewritten set of constraints, to further enlarge the class of terminating dependencies.
- The rewriting technique permits us to introduce a new class of terminating constraints consisting of sets of dependencies which are detected as acyclic by our rewriting algorithm. This class, called *Acyclic* ($\mathcal{AC}$) generalizes $\mathcal{LS}$ and, considering static criteria for checking chase termination, AC is the most general criterion so far proposed [1].

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 $Consts \cup Nulls$. Given an instance K, $Nulls(K)$ (resp. $Consts(K)$) denotes the set of labeled nulls (resp. constants) occurring in K. An *atomic formula* (or *atom*) is of the form $R(t_1, \ldots, t_n)$ where R is a relational predicate, $t_1, \ldots, t_n$ are terms belonging to the domain $Consts \cup Vars$ and $n = ar(R)$.

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 $K_C[S] = \{K_1[S], \ldots, K_n[S]\}$.

Given a relational schema $\mathbf{R}$, a *tuple generating dependency* (TGD) over $\mathbf{R}$ is a formula of the form

$$r : \forall \mathbf{x} \forall \mathbf{z}\ \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y}) \tag{1}$$

where $\phi(\mathbf{x}, \mathbf{z})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atomic formulas over $\mathbf{R}$; $\phi(\mathbf{x}, \mathbf{z})$ is called the *body* of r, denoted as $Body(r)$, while $\psi(\mathbf{x}, \mathbf{y})$ is called the *head* of r, denoted as $Head(r)$. An *equality generating dependency* (EGD) over $\mathbf{R}$ is a formula of the form

$$\forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow x_1 = x_2 \tag{2}$$

where x_1 and x_2 are among the variables in $\mathbf{x}$. In the following we will often omit 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.

DEFINITION 1 (HOMOMORPHISM). 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 $Consts(K_1) \cup Nulls(K_1)$ to $Consts(K_2) \cup Nulls(K_2)$ such that: (1) $h(c) = c$, for every $c \in Consts(K_1)$, 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 $h'\colon K_2 \rightarrow K_1$. □

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 $Consts(J) \cup Nulls(J)$ s.t. for every atom $R(x_1, \ldots, x_n)$ of $\phi(\mathbf{x})$ the fact $R(h(x_1), \ldots, h(x_n))$ is in J.

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 $J \models \Sigma$ (i.e. J satisfies all constraints in Σ). 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 $USol(D, \Sigma)$.

Chase step. Let K be a database instance.

1. Let r be a TGD $\phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$. Let h be a homomorphism from $\phi(\mathbf{x}, \mathbf{z})$ to K such that there is no extension of h to a homomorphism h' from $\phi(\mathbf{x}, \mathbf{z}) \wedge \psi(\mathbf{x}, \mathbf{y})$ to K[2]. 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 $K \overset{r,h}{\rightarrow} K'$.

DEFINITION 2 (CHASE [9]). 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.

[1] In this paper we will use calligraphic style $\mathcal{C}$ in order to denote the terminating class of constraints recognized by criterion C.

[2] A variant of this step is the *oblivious* one that applies to an instance K if there is a homomorphism h from $\phi(\mathbf{x}, \mathbf{z})$ to K.

- 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*. □

In [9] it has been shown that, for any instance D and set of constraints 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. Observe that whenever several alternative chase steps could be applied, 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.

Constraints equivalence. 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.
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 nonempty 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 every 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 $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv \langle \mathbf{R}_2, \Sigma_2 \rangle$.

2.1 Chase termination conditions

This section presents a brief overview on the well-known chase termination conditions that guarantee for every database D the termination of all chase sequences in PTIME in the size of D.

Weak acyclicity. Let Σ be a set of TGDs 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 (*WA*) 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 $\phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ 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 in position R_i in ϕ and for every y in $\mathbf{y}$ appearing in position T_k in ψ, add a special edge $R_i \overset{*}{\rightarrow} T_k$ (if it does not already exist). Σ is *weakly acyclic* if $dep(\Sigma)$ has no cycle going through a special edge.

Safety. The safety condition (SC) [14] is based on the notion of affected positions. 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 if there is a constraint $r : \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ 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 (since $aff(\Sigma) \subseteq pos(\Sigma)$). Moreover, Σ is said to be safe if $prop(\Sigma)$ does not contain cycles with special edges.

Stratification. The idea behind *stratification* (Str) [8] is to decompose the set of constraints into independent subsets, where each subset consists of constraints that may fire each other, and check each component separately for weak acyclicity. 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 K_1 and two homomorphisms h_1 and h_2 such that i) $K_1 \not\models h_1(r_1)$ ii) $K_1 \overset{r_1,h_1}{\rightarrow} K_2$, iii) $K_2 \not\models h_2(r_2)$ and iv) $K_1 \models h_2(r_2)$. Intuitively, $r_1 \prec r_2$ means that firing r_1 can cause the firing of r_2. We say that Σ is stratified iff the constraints in every cycle of the *chase graph* $G(\Sigma) = (\Sigma, \{(r_1, r_2) | r_1 \prec r_2\})$ are weakly acyclic.
For every database D, stratification ensures the existence of a chase sequence which terminates in polynomial time in the size of D [14].
In order to guarantee the termination of all chase sequences, the *c-stratification* criterion ($CStr$) has been proposed [14]. Basically, c-stratification defines a different chase graph and applies a constraint whenever its body is satisfied.
In particular, $r_1 \prec_c r_2$ iff i) $K_1^* \overset{r_1,h_1}{\rightarrow} K_2$, ii) $K_2 \not\models h_2(r_2)$ and iii) $K_1 \models h_2(r_2)$, where the oblivious chase step $K_1^* \overset{r_1,h_1}{\rightarrow} K_2$ states that there is a homomorphism h_1' extending h_1 which associates every existentially variable y in $h_1(r_1)$ to a fresh labeled null.

Super-weak acyclicity. The super-weak acyclicity (SwA) [12] builds a *trigger graph* $\Upsilon(\Sigma) = (\Sigma, E)$ 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.
Let Σ be a set of TGDs and let $sk(\Sigma)$ be the logic program obtained by skolemizing Σ, i.e. by replacing each existentially quantified variable y appearing in the head of a TGD r by the skolem function $f_y^r(\mathbf{x})$, where $\mathbf{x}$ is the set of variables appearing both in the body and in the head of r. A *place* is a pair (a, i) where a is an atom of $sk(\Sigma)$ and $0 \leq i \leq ar(a)$. Given a TGD r and an existential variable y in the head of r, $Out(r, y)$ denotes the set of places (called *output places*) in the head of $sk(r)$ where a term of the form $f_y^r(\mathbf{x})$ occurs. Let r be a TGD r and let x be a universal variable of r, $In(r, x)$ denotes the set of places (called *input places*) in the body of r where x occurs.
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 $q \sim q'$.
For any set Q of places, $Move(\Sigma, Q)$ denotes the smallest set of places Q' such that $Q \subseteq Q'$, and for every constraint $r = B_r \rightarrow H_r$ in $sk(\Sigma)$ and every variable x, if $\Pi_x(B_r) \sqsubseteq Q'$ then $\Pi_x(H_r) \subseteq Q'$, where $\Pi_x(B_r)$ and $\Pi_x(H_r)$ denote the sets of places in B_r and H_r where x occurs.
Given a set Σ of TGDs and two TGDs $r_1, r_2 \in \Sigma$, we say that r_1 triggers r_2 in Σ and write $r_1 \rightsquigarrow r_2$ iff there exists an existential variable y in the head of r_1, and a universal variable x_2 occurring both in the body and head of r_2 such that $In(r_2, x) \sqsubseteq Move(\Sigma, Out(r_1, y))$. A set of constraints Σ is super-weakly acyclic iff the trigger graph $\Upsilon(\Sigma) = (\Sigma, \{(r_1, r_2) | r_1 \rightsquigarrow r_2\})$ 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 SC [10] but is not comparable with $CStr$.

Safe restriction. A more refined extension of stratification and safety has been proposed in [13, 14] under the name of *safe restriction* (*SR*). Basically, safe restriction refines stratification by considering constraints firing and possible propagation of null values together.
For any set of positions P and a TGD r, $aff(r, P)$ denotes the set

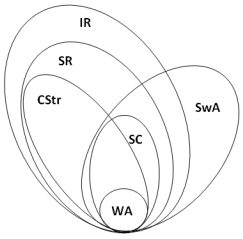

Figure 1: Criteria Relationships.

of positions π from the head of r such that i) for every universally quantified variable x in π, x occurs in the body of r only in positions from P or ii) π contains an existentially quantified variable.
For any $r_1, r_2 \in \Sigma$ and $P \subseteq pos(\Sigma)$, $r_1 \prec_P r_2$ if 1) $r_1 \prec_c r_2$ (i.e. there exists a database instance K_1 and two homomorphisms h_1 and h_2 such that i) $K_1 \overset{r_1,h_1}{\rightarrow} K_2$, ii) $K_2 \not\models h_2(r_2)$ and iii) $K_1 \models h_2(r_2)$) and 2) there is null value propagated from the body to the head of $h_2(r_2)$ s.t. it occurs in K_1 only in the positions from P.
A 2-restriction system is a pair $(G'(\Sigma), P)$, where $G'(\Sigma) = (\Sigma, E)$ is a directed graph and $P \subseteq pos(\Sigma)$ such that: i) for all $(r_1, r_2) \in E$: if r_1 is TGD, then $aff(r_1, P) \cap pos(\Sigma) \subseteq P$, whereas if r_2 is TGD, then $aff(r_2, P) \cap pos(\Sigma) \subseteq P$, and ii) $r_1 \prec_P r_2 \Rightarrow (r_1, r_2) \in E$.
Σ is called *safely restricted* if and only if there is a restriction system $(G'(\Sigma), P)$ for Σ such that every strongly connected component in $G'(\Sigma)$ is safe.

Inductive restriction. *Inductive restriction* (*IR*) refines *SR* by partitioning constraints in a more refined way. In particular, it first computes the system $(G'(\Sigma), P)$ and partition Σ into $\Sigma_1, ..., \Sigma_n$, where each Σ_i is a set of dependencies defining a strongly connected components in $G'(\Sigma)$, next, if $n = 1$ the safety criterion is applied to Σ, otherwise the *IR* criterion is applied inductively to each Σ_i.

Criteria relationships. A complete characterization of the relationships among termination condition criteria is shown in Figure 1.

2.2 Constraints Rewriting

A technique for checking chase termination by rewriting the source set of constraints into an equivalent set has been presented in [10]. The technique 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, based on the use of adornments to perform a deeper pattern analysis, allows to recognize larger classes of constraints for which chase termination is guaranteed – if Σ satisfies chase termination conditions C, then the rewritten set Σ^α satisfies C as well, but the vice versa is not true, that is there are significant classes of constraints for which Σ^α satisfies C and Σ does not.

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*. Given a TGD $r : \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ and let α be a coherent adornment for the body atoms, then $HeadAdn(r, \phi^\alpha(\mathbf{x}, \mathbf{z}))$ denotes the *adorned head* of r (with respect to the adorned body $\phi^\alpha(\mathbf{x}, \mathbf{z})$) 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. 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.

EXAMPLE 2. Consider the constraints Σ_1 of Example 1. Initially, $Adn(\Sigma_1)$ contains two constraints derived by adorning the body variables as bound ($Base(\Sigma_1)$)

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

In the second step two new constraints are generated ($Derived(\Sigma_1)$). Due to the new predicate E^{bf}, the following constraint, derived from constraint r_2, is 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" ($In(\Sigma_1)$):

$$r_5 : \quad N(x) \rightarrow N^b(x)$$
$$r_6 : \quad S(x) \rightarrow S^b(x)$$
$$r_7 : \quad E(x, y) \rightarrow E^{bb}(x, y)$$

and TGDs mapping tuples of adorned relations into "output" relations ($Out(\Sigma_1)$):

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

It is important to observe that the set of constraints Σ_1 is neither inductively restricted nor super-weakly acyclic, while $Adn(\Sigma_1)$ is weakly acyclic. In fact, $dep(Adn(\Sigma_1))$, without considering edges in $In(\Sigma_1)$ and $Out(\Sigma_1)$, which do not affect the analysis of termination conditions, contains only the following edges: $N_1^b \rightarrow E_1^{bf}$, $N_1^b \overset{*}{\rightarrow} E_2^{bf}$, $E_2^{bb} \rightarrow N_1^b$, $E_2^{bf} \rightarrow N_1^f$, $N_1^f \rightarrow E_1^{ff}$, $N_1^f \overset{*}{\rightarrow} E_2^{ff}$.
To show the equivalence between Σ and Σ^α the following definition has been introduced. 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) $Adn(\mathbf{R}, \Sigma) = \mathbf{R} \cup \{p^\alpha(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in$

$\mathbf{R} \wedge p^{\alpha}$ *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, (iii) $Map(\mathbf{R}) = \mathbf{R} \cup \hat{\mathbf{R}}$ the union of the input and output schemas, and (iv) $Map(\Sigma) = \Sigma \cup \{p(x_1, ..., x_n) \rightarrow \hat{p}(x_1, ..., x_n) | \ p(A_1, ..., A_n) \in \mathbf{R}\}$ 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.

THEOREM 1. [10] 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.
Let $\mathcal{C}$ denote the class of TGDs satisfying criterion C, Adn-$\mathcal{C}$ denotes the class of TGDs Σ such that $Adn(\Sigma)$ satisfies criterion C. The below theorem states that the rewriting technique allows to recognize (by using classical criteria) larger classes of constraints for which chase termination is guaranteed.

THEOREM 2. [10] For any $C, C' \in \{WA, SC, CStr, SwA, SR, IR\}$, i) $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C}$ and ii) $\mathcal{C} \subseteq \mathcal{C}'$ implies $Adn\text{-}\mathcal{C} \subseteq Adn\text{-}\mathcal{C}'$. □

3. LOCAL STRATIFICATION

In this section we present some improvements for termination conditions discussed in Section 2 and then introduce the class of *locally stratified dependencies*, that generalizes previously known classes, for which termination of the chase algorithm is guaranteed. The idea underlying stratification, also used in its extensions (e.g. $CStr$, *SR*) and in the super-weak acyclicity, is to consider in the propagation of nulls how constraints may fire each other. However, there are simple cases where current criteria are not able to understand that all chase sequences are finite (see, for instance, the following examples 4 and 9). Thus, in this section we first introduce a new version of stratification, called *WA-stratification* (*WA*-Str) which generalizes $CStr$ and guarantees, for all databases, termination of all chase sequences.
(C-)stratification does not specify what kind of cycles are checked (i.e. simple or general).Checking simple cycles is not correct as it may not consider all possible chase sequences, but checking general cycles, means that for each strongly connected component there is one cycle including all nodes in the component which subsumes all other cycles on the same component (in terms of constraints to be considered).

EXAMPLE 3. Consider the following set of TGDs Σ_3:

$$\begin{array}{lrcl} r_1: & P(x) & \rightarrow & \exists y\, Q(x,y) \\ r_2: & Q(x,y) & \rightarrow & R(x,y) \\ r_3: & R(x,y) & \rightarrow & P(x) \\ r_4: & R(x,y) & \rightarrow & S(y,x) \\ r_5: & S(x,y) & \rightarrow & Q(x,y) \end{array}$$

We have that $r_1 \prec_c r_2$, $r_2 \prec_c r_3$, $r_3 \prec_c r_1$, $r_2 \prec_c r_4$, $r_4 \prec_c r_5$ and $r_5 \prec_c r_2$. The c-chase graph contains two simple cycles, i.e. $\{r_1, r_2, r_3\}$ and $\{r_2, r_4, r_5\}$, that are both weakly acyclic, and a general cycle involving all the TGDs in Σ_3 that is not weakly acyclic. □

Although considering the constraints involved in every cycle is not wrong, this is equivalent to just considering the subsets of constraints involved in every strongly connected component, since if the weak acyclicity property is satisfied by a set of constraints it is satisfied by all its subsets as well. Moreover, the number of cycles in a graph could be exponential, whereas the number of strongly connected components is polynomial. Thus, a first observation on (c-)stratification (in terms of correctness, if simple cycles are considered, or in terms of efficiency, if all cycles are considered) is that it refers to cycles instead of strongly connected components. A further observation is that it uses oblivious chase for checking termination of standard chase and, as previously said, its applicability is limited.

DEFINITION 3 (WA-STRATIFICATION). Given a set of dependencies Σ and $r_1, r_2 \in \Sigma$, we say that $r_1 < r_2$ iff there exist a relational database instance K, homomorphisms h_1, h_2 and a set S of atoms, such that (i) $K \not\models h_1(r_1)$, (ii) $K \overset{r_1,h_1}{\rightarrow} J$, (iii) $K \cup S \models h_2(r_2)$, (iv) $J \cup S \not\models h_2(r_2)$ and (v) $Null(S) \cap (Null(J) - Null(K)) = \emptyset$ (i.e. S does not contain new null values introduced in J).
We say that Σ is *WA-stratified* (*WA*-Str) iff the constraints in every nontrivial strongly connected component of the *firing graph* $\Gamma(\Sigma) = (\Sigma, \{(r_1, r_2) | r_1 < r_2\})$ are weakly acyclic. □

With respect to stratification, *WA*-Str also considers in the satisfaction of constraint r_2, in addition to the database K, a set of atoms S (cond. (iii)) and atoms in S cannot contain null values introduced in the application of the constraint r_1 (cond. (v)). Moreover, since we are considering strongly connected components (instead of cycles) these components must not be trivial, that is they must have at least one edge, otherwise the constraint cannot be fired cyclically. As a further important observation, in the above definition we consider standard chase for both constructing the graph $\Gamma(\Sigma)$ and checking weak acyclicity.

EXAMPLE 4. The set of constraints Σ_4 consisting of the TGD

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

is not c-stratified, but is WA-stratified. □

The following proposition states that *WA*-Str criterion is more general than $CStr$ and is not comparable with SC. Consequently it is not comparable even with SwA as $\mathcal{SC}$ is strictly contained in $\mathcal{SwA}$ and $\mathcal{CS}tr$ is not comparable with $\mathcal{SwA}$.

PROPOSITION 1. $\mathcal{CS}tr \subsetneq \mathcal{WA}\text{-}\mathcal{S}tr$ and $\mathcal{SC} \not\parallel \mathcal{WA}\text{-}\mathcal{S}tr$. □

It is important to observe that *WA*-Str criterion could be improved by testing safety instead of weak acyclicity over the firing graph. Further improvements could be obtained by considering super-weak acyclicity instead of safety.

DEFINITION 4. Given a set of TDGs Σ, we say that Σ is *SC-stratified* (*SC*-Str) if the constraints in every strongly connected component of the firing graph $\Gamma(\Sigma)$ are safe.
Moreover, we say that Σ is *SwA-stratified* (*SwA*-Str) if the constraints in every strongly connected component of the firing graph $\Gamma(\Sigma)$ are super-weak acyclic. □

We now analyze the complexity of the above criteria starting by defining a bound on the complexity of the firing problem, i.e. the complexity of checking whether $r_1 < r_2$.

LEMMA 1. Let $r_1 : \ \phi_1 \rightarrow A_1 \wedge \cdots A_k$ and $r_2 : \ B_1 \wedge \cdots B_n \rightarrow \psi_2$ be two TGDs. The problem of checking whether $r_1 < r_2$ is bounded by $O((k+1)^n)$. □

Although the theoretical complexity of the "firing" problem is exponential, in most cases it is very low (e.g. inclusion dependencies, multivalued dependencies), as usually the number n of body atoms in the fired constraint r_2 is small and the number of atoms in the head of constraint r_1 which could be used to fire r_2 through their unification with B_i (i.e. $k_i > 1$) is even smaller. Indeed, if the number of atoms in the body of r_2 is bounded by a constant, the firing problem is in PTIME. Significative subclasses of constraints for which the firing problem becomes polynomial could be identified, but this is outside the aim of this work. In the following, for a given set of constraints Σ, we shall denote with C_{ij} the complexity of the problem of checking whether $r_i < r_j$, for $r_i, r_j \in \Sigma$, and with $C_m = max\{C_{ij} | r_i, r_j \in \Sigma\}$.

PROPOSITION 2. Let Σ be a set of TGDs, D be a database Then:

- the problem of checking whether Σ is WA-stratified (resp. SC-stratified, SwA-stratified) is bounded by $O(C_m \times |\Sigma|^2)$;
- if Σ is WA-stratified (resp. SC-stratified, SwA-stratified), the length of every chase sequence of Σ over D is polynomial in the size of D. □

The class of constraints satisfying criterion C-Str will be denoted by $\mathcal{C}$-$\mathcal{S}tr$. The next theorem states the relationships among the above mentioned criteria and other previously defined conditions.

THEOREM 3.

1. $\mathcal{WA}$-$\mathcal{S}tr \subsetneq \mathcal{SC}$-$\mathcal{S}tr \subsetneq \mathcal{SwA}$-$\mathcal{S}tr$,
2. for $C \in \{WA, SC, SwA\}$, $\mathcal{C} \subsetneq \mathcal{C}$-$\mathcal{S}tr$ and
3. $\mathcal{SR} \not\Vdash \mathcal{SwA}$-$\mathcal{S}tr$ and $\mathcal{IR} \not\Vdash \mathcal{SwA}$-$\mathcal{S}tr$. □

It is trivial that more powerful criteria could be defined by composing criteria which are not comparable. We next present a different generalization of super-weak acyclicity which also generalizes the class $\mathcal{IR}$.

We start by introducing a notion of *fireable place*. We say that a place q appearing in the body of constraint r could be fired by a place q' appearing in the head of constraint r', denoted by $q' < q$, if $q \sim q'$ and $r' < r$. Given two sets of places Q and Q' we say that Q could be fired by Q', denoted by $Q' < Q$ iff for all $q \in Q$ there exists some $q' \in Q'$ such that $q' < q$.

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 constraint $r = B_r \rightarrow H_r$ in $sk(\Sigma)$ and every variable x, if $Q' < \Pi_x(B_r)$ then $\Pi_x(H_r) \subseteq Q'$, where $\Pi_x(B_r)$ and $\Pi_x(H_r)$ denote the sets of places in B_r and H_r where x occurs.

With respect to the function $Move$, the new function $MOVE$ here considered takes into account the firing of places and not only the unification of places.

DEFINITION 5 (LOCAL STRATIFICATION). Given a set Σ of TGDs and two TGDs $r_1, r_2 \in \Sigma$, we say that r_1 triggers r_2 in Σ and write $r_1 \hookrightarrow r_2$ iff there exists an existential variable y in the head of r_1, and a universal variable x occurring both in the body and head of r_2 such that $MOVE(\Sigma, Out(r_1, y)) < In(r_2, x)$. A set of constraints Σ is *locally stratified* (LS) iff the trigger graph $\Delta(\Sigma) = \{(r_1, r_2) | r_1 \hookrightarrow r_2\}$ is acyclic. □

PROPOSITION 3. For every set of TGDs Σ and for every database D

- the problem of checking whether Σ is locally stratified is bounded by $O(C_m \times |\Sigma|^2)$;

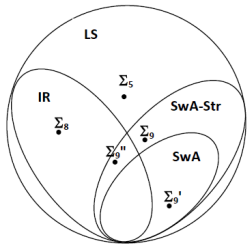

Figure 2: Criteria Relationships.

- if Σ is locally stratified, the length of every chase sequence of Σ over D is polynomial in the size of D. □

The below theorem states that the class of locally stratified constraints (denoted by $\mathcal{LS}$) is more general than $\mathcal{SwA}$-$\mathcal{S}tr$ and $\mathcal{IR}$.

THEOREM 4. $\mathcal{SwA}$-$\mathcal{S}tr \subsetneq \mathcal{LS}$ and $\mathcal{IR} \subsetneq \mathcal{LS}$. □

The next example shows that the containment of $\mathcal{SwA}$-$\mathcal{S}tr$ and $\mathcal{IR}$ in $\mathcal{LS}$ is strict, whereas Figure 2 resumes the relationships among the above discussed criteria.

EXAMPLE 5. The following set of constraints is neither in $\mathcal{IR}$ nor in $\mathcal{SwA}$-$\mathcal{S}tr$, but it belongs to the class $\mathcal{LS}$:

$$r_1 : N(x) \rightarrow \exists y \exists z\, E(x, y) \wedge S(z, y)$$
$$r_2 : E(x, y) \wedge S(x, y) \rightarrow N(y)$$
$$r_3 : E(x, y) \rightarrow E(y, x)$$ □

The below corollary shows that we can further improve classes of terminating constraints by using rewriting techniques [10].

COROLLARY 1. For $C \in \{WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$, $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C}$. □

4. CHECKING CHASE TERMINATION BY CONSTRAINTS REWRITING

Corollary 1 in the previous section evidenced that the rewriting function Adn can still be useful to improve termination criteria and enlarge the class of constraints for which chase termination is guaranteed. Moreover, there are still simple sets of constraints which are not recognized by any technique so far considered, even if constraints are rewritten. The below example shows such a case.

EXAMPLE 6. Consider the set of constraints Σ_6:

$$r_1 : R(\overset{p_1}{x}) \rightarrow S(\overset{p_2}{x}, \overset{p_3}{x})$$
$$r_2 : S(\overset{p_4}{x_1}, \overset{p_5}{x_2}) \rightarrow \exists z\, T(\overset{p_6}{x_2}, \overset{p_7}{z}) \wedge Q(\overset{p_8}{x_2})$$
$$r_3 : T(\overset{p_9}{x_1}, \overset{p_{10}}{x_2}) \wedge T(\overset{p_{11}}{x_1}, \overset{p_{12}}{x_3}) \wedge T(\overset{p_{13}}{x_3}, \overset{p_{14}}{x_1}) \rightarrow R(\overset{p_{15}}{x_2})$$

The set Σ_6 is not LS since $r_2 \hookrightarrow r_2$. Indeed, $r_2 < r_3 < r_1 < r_2$ and $MOVE(\Sigma_6, Out(r_2, z)) = \{p_7, p_{15}, \mathbf{p_2}, p_3, p_6, p_8\} < In(r_2, x_1) = \{\mathbf{p_4}\}$. Σ_6 is not in Adn-$\mathcal{LS}$ as well since $Adn(\Sigma_6)$ is not in $\mathcal{LS}$ (see Appendix B for more details). However it is easy to check that the chase terminates for all database instances. □

Thus, in this section we present a new rewriting technique which allow us to detect larger classes of constraints for which chase termination is guaranteed. Our rewriting algorithm is inspired by the one presented in [10] and similar to the algorithm Adn^+ there presented, but uses different adornments for free variables. Before

presenting our rewriting algorithm let us recall some further notations. Let $Adn(\cdot)$ be the function rewriting a set of constraints Σ, defined over a database schema $\mathbf{R}$, into an adorned set Σ^α (as proposed in [10]) we shall denote with $Adn^{-1}(\cdot)$ the function taking 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. Since the new rewriting function here introduced adorns constraints using different free adornments of the form f_i, we shall denote with $src(r^\alpha)$ (or simply r) the constraint in the source set Σ from which an adorned one r^α has been derived.[3]

Head adornment. Given a TGD $r\colon \phi(\mathbf{x},\mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x},\mathbf{y})$ and let $\phi^\alpha(\mathbf{x},\mathbf{z})$ be a coherent adornment for the body atoms, then $SkHeadAdn(r, \phi^\alpha(\mathbf{x},\mathbf{z}))$ denotes the adorned head of r obtained as follows: i) every universally quantified variable has the same adornment of the body occurrences, ii) existentially quantified variables appearing in constraints with empty body and constants are adorned as b; iii) every existentially quantified variable y is adorned with an adornment f_i where the subscript is an integer value associated to the skolem function $f^r_y(\alpha[\mathbf{x}])$, (here $\alpha[\mathbf{x}]$ denotes the substring of α corresponding to $\mathbf{x}$). For instance, for $r:\ R(x,z) \rightarrow \exists y\, R(x,y)$ we have that $SkHeadAdn(r, R^{bb}(x,z)) = \exists y\, R^{bf_1}(x,y)$, where $f_1 = f^r_y(b)$, and $SkHeadAdn(r, R^{bf_1}(x,z)) = \exists y\, R^{bf_1}(x,y)$.

Adornment substitution. In order to have terminating sequences we will also use substitutions for adornments sequences. 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.

The algorithm builds a graph E storing dependencies among adorned predicates. In particular, an edge (p^α_r, q^β_s) states that the predicate p^α, appearing in the head of an adorned constraint derived from r, has caused the creation of an adorned predicate q^β appearing in the head of a constraint derived from s. The graph is built for checking cyclic dependencies among adorned predicates denoting possible non-terminating chase sequences.

Adornment algorithm. The rewriting of constraints is performed by the function Adn++ reported in Figure 3 which differs from the Adn function defined in [10] in several aspects: (i) in the generation of adorned predicates it also considers how constraints may fire each other; (ii) the adornment of the head is done by applying the function $SkHeadAdn$ which introduces adornments with subscripts and these different free adornments appear in the output set; (iii) when a new adorned constraint r^α is generated, if there is an adorned constraint r^β and a substitution θ such that $r^\alpha\theta = r^\beta$, r^α is replaced by $Body(r^\alpha) \rightarrow Head(r^\beta)$, to avoid the creation of an infinite set of adorned constraints; (iv) in addition to the adorned set of TGDs, the algorithm also returns a boolean taking into account the fact that a form of cyclicity has been detected and a substitution has been used to unify two adorned constraints (to avoid an infinite set of adorned TGDs).

It is worth noting that if there are constraints with constants it is possible to generate adorned constraints with (adorned) body predicates not depending on the source predicates; these constraints are useless and, therefore, could be dropped.

Analogously to the function Adn, the function Adn++ receives in input a set of TGDs Σ, but differently from Adn, it returns a pair consisting of an adorned set of TGDs (with different free symbols) and a boolean value giving information about the detection of a form of cyclicity. The two elements returned by the algorithm (set of constraints and boolean value) are denoted by Adn++$(\Sigma)[1]$ and Adn++$(\Sigma)[2]$, respectively. Adn++$(\Sigma)[1]$ consists of four different subsets: $Base(\Sigma)$, $Derived(\Sigma)$, $In(\Sigma)$ and $Out(\Sigma)$, denoting, respectively, base, derived, input and output constraints.

[3] We shall use $src(r^\alpha)$, instead of $Adn^{-1}(r^\alpha)$, when we are interested in the constraint identifier rather than in the constraint definition.

Function Adn++(Σ);
Input Set of TGDs Σ over schema $\mathbf{R}$;
Output Set of (adorned) TGDs $Base \cup Derived \cup In \cup Out$
Boolean value Cyc;
begin
$Base = Derived = In = Out = New\_Pred = E = \emptyset$;
$Cyc = false$;
// Let $Body^b(r)$ be the conjunction obtained by adorning atoms
// in $Body(r)$ with strings of b symbols
$Used\_Pred = \{p^b_r \mid \exists r \in \Sigma$ and $\exists p^b$ in $Body^b(r)\}$;
for *each applicable* $r \in \Sigma$ **do begin**
$Base = Base \cup \{Body^b(r) \rightarrow SkHeadAdn(r, Body^b(r))\}$;
$New\_Pred = New\_Pred \cup \{p^\alpha_r | p^\alpha(t) \in SkHeadAdn(r, Body^b(r))\} - Used\_Pred$;
end for;
while $(New\_Pred \neq \emptyset)$ **do begin**
Select nondeterministically $p^{\alpha_1 \ldots \alpha_n}_s \in New\_Pred$;
$New\_Pred = New\_Pred - \{p^{\alpha_1 \ldots \alpha_n}_s\}$;
$Used\_Pred = Used\_Pred \cup \{p^{\alpha_1 \ldots \alpha_n}_s\}$;
for *each* $r \in (Base \cup Derived)$ *s.t.* $s < src(r)$ **do**
for *each* $p^\beta(x_1, ..., x_n) \in Body(r)$ **do begin**
$B' = Body(r) - \{p^\beta(x_1, ..., x_n)\} \cup \{p^{\gamma_1 \ldots \gamma_n}(x_1, ..., x_n)\}$;
// $\gamma_i = b$ *if* $x_i \in Consts$;
// $\gamma_j = \alpha_i$ *if* $x_i \in Vars$; $(i \in [1..n])$
if B' *is coherent* **then begin**
Let $H' = SkHeadAdn(Adn^{-1}(r), B')$;
if $(\exists r^\beta \in Derived$ *and* $\exists subst.\ \theta$ *s.t.* $(B' \rightarrow H')\theta = r^\beta)$
then begin
$Derived = Derived \cup \{B' \rightarrow Head(r^\beta)\}$;
$E = E \cup \{(p^{\alpha_1 \ldots \alpha_n}_s, p^\omega_{src(r)}) | p^\omega(t) \in Head(r^\beta)\}$;
if (r is exist. quantified **and** E is cyclic) **then**
$Cyc = true$;
end;
else begin
$Derived = Derived \cup \{B' \rightarrow H'\}$;
$E = E \cup \{(p^{\alpha_1 \ldots \alpha_n}_s, p^\omega_{src(r)}) | p^\omega(t) \in H'\}$;
$New\_Pred = New\_Pred \cup \{p^\omega_{src(r)} | p^\omega(t) \in H' \wedge p^\omega_{src(r)} \notin Used\_Pred\}$;
end;
end;
else $Derived = Derived \cup \{B' \rightarrow Head(Adn^{-1}(r))\}$;
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 $\langle Base \cup Derived \cup In \cup Out, Cyc \rangle$;
end.

Figure 3: Constraint Rewriting Function Adn++.

EXAMPLE 7. Consider the set of constraints Σ_7

$$\begin{array}{ll} r_1:\ N(x) & \rightarrow \exists y\, E(x,y) \\ r_2:\ E(x,y) & \rightarrow N(y) \end{array}$$

The following set of base adorned constraints are first introduced:

$$\begin{array}{ll} s_1:\ N^b(x) & \rightarrow \exists y\, E^{bf_1}(x,y) \\ s_2:\ E^{bb}(x,y) & \rightarrow N^b(y) \end{array}$$

Next, the below adorned constraints are introduced in the set $Derived$:

$$\begin{array}{ll} s_3:\ E^{bf_1}(x,y) & \rightarrow N^{f_1}(y) \\ s_4:\ N^{f_1}(x) & \rightarrow \exists y\, E^{f_1 f_2}(x,y) \\ s_5:\ E^{f_1 f_2}(x,y) & \rightarrow N^{f_2}(y) \\ s_6:\ N^{f_2}(x) & \rightarrow \exists y\, E^{f_2 f_3}(x,y) \\ s_7:\ E^{f_2 f_3}(x,y) & \rightarrow N^{f_3}(y) \end{array}$$

At this point the constraint

$$s':\ N^{f_3}(x) \rightarrow \exists y\, E^{f_3 f_4}(x,y)$$

should be added, but since there is a substitution $\theta = \{f_3/f_1, f_4/f_2\}$, the TGD

$$s_8:\ N^{f_3}(x) \rightarrow \exists y\, E^{f_1 f_2}(x,y)$$

is added and the generation of derived constraints terminates. Moreover, the graph E contains a cycle $E^{f_1 f_2} \rightarrow N^{f_2} \rightarrow E^{f_2 f_3} \rightarrow N^{f_3} \rightarrow E^{f_1 f_2}$ and the adornment function returns the boolean value $Adn{+}{+}(\Sigma_7)[2] = true$. □

The next results state that the rewriting function $Adn{+}{+}$ terminates and gives in output a set of TGDs equivalent to the input set.

LEMMA 2. For every set of TGDs Σ the function $Adn{+}{+}$ always terminates. □

THEOREM 5. For every set of TGDs Σ over a database schema $\mathbf{R}$, $\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn{+}{+}(\mathbf{R}, \Sigma), Adn{+}{+}(\Sigma)[1]\rangle$, where $Adn{+}{+}(\mathbf{R}, \Sigma) = \mathbf{R} \cup \{p^{\alpha}(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in \mathbf{R} \wedge p^{\alpha}$ *appears in* $Adn{+}{+}(\Sigma)[1]\} \cup \hat{\mathbf{R}}$. □

Let $\mathcal{C}$ denote a class of TGDs for which chase termination is guaranteed by checking a given criterion (e.g. $\mathcal{C} \in \{\mathcal{WA}, \mathcal{SC}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{S}w\mathcal{A}, \mathcal{SR}, \mathcal{IR}, \mathcal{WA}\text{-}\mathcal{S}tr, \mathcal{SC}\text{-}\mathcal{S}tr, \mathcal{S}w\mathcal{A}\text{-}\mathcal{S}tr, \mathcal{LS}\}$), we shall denote with $Adn{+}{+}\mathcal{C}$ the class of TGDs Σ such that $Adn{+}{+}(\Sigma)[1]$ is in $\mathcal{C}$. The following theorem shows that the rewriting technique here introduced is useful to enlarge the class of constraints which are recognized as terminating by a given criterion C.

THEOREM 6. $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C} \subsetneq Adn{+}{+}\mathcal{C}$, *for* $C \in \{WA, SC, Str, CStr, SwA, SR, IR, WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$. □

It is important to recall that a rewriting technique using adornments with subscripts has been first defined in [10]. Moreover, differently from the previous rewriting technique, where new subscripts are introduced any time a new adorned constraint with existentially quantified variables is generated, the current technique controls the introduction of subscript (i.e. different free symbols) using skolem functions applied to adornments of head universally quantified variables. In addition, different free symbols appear in the output set of the rewriting function here presented, whereas previous techniques returned in output a set of constraints with just two adornment symbols (b and f). To better understand the advantages provided by the current rewriting technique we refer to Example 11 in Appendix B.

We conclude by introducing our final class of constraints for which chase termination is guaranteed.

DEFINITION 6 (ACYCLICITY). A set of TGDs Σ is said to be *Acyclic (AC)* if $Adn{+}{+}(\Sigma)[2]$ is false. □

The class of acyclic constraint is denoted by $\mathcal{AC}$. The next theorem shows that the hierarchy criteria here introduced collapse and coincide with the class $\mathcal{AC}$ when constraints are rewritten using the adornment function $Adn{+}{+}$.

THEOREM 7. $\mathcal{AC} = Adn{+}{+}\mathcal{WA} = Adn{+}{+}\mathcal{LS}$. □

The below corollaries derive straightforwardly from the previous theorem and state that AC is the most general chase termination criterion and, for acyclic constraints, the length of all chase sequences is polynomial in the size of the input database.

COROLLARY 2. Let Σ be a set of acyclic TGDs. Then for every database D all chase sequences of Σ over D terminate in polynomial time in the size of D. □

COROLLARY 3. $\mathcal{LS} \subsetneq \mathcal{AC}$. □

The complete relationships among the previously discussed criteria is reported in Figure 4.

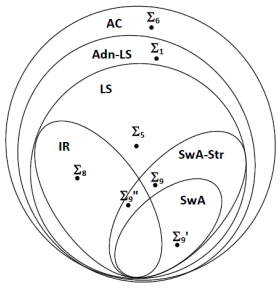

Figure 4: Criteria Relationships.

5. CONCLUSIONS

The paper has presented new criteria and a rewriting technique which allow to recognize larger classes of constraints for which the chase fixpoint algorithm always terminates, independently of the database instance. In this work we have not considered equality generating dependencies. Future work will address this important aspect. A possible solution to this problem could be based on simple transformations replacing EGDs with TGDs, such as the ones made in [12]. Moreover, as discussed in [15, 4, 10], there are simple cases of EGDs that could be easily treated such as those defining, in the framework of ontology languages, functional restriction on roles [15] and keys which are not conflicting with TGDs [4].

6. REFERENCES

[1] A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM TODS*, 4(3):297–314, 1979.

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

[3] L. E. Bertossi, S. Kolahi, and L. V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. In *ICDT*, pages 268–279, 2011.

[4] A. Cali, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS*, pages 77–86, 2009.

[5] A. Cali, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *PVLDB*, 3(1):554–565, 2010.

[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, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *TCS*, 336(1), 2005.

[10] S. Greco and F. Spezzano. Chase termination: A constraints rewriting approach. *PVLDB*, 3(1):93–104, 2010.

[11] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM TODS*, 4(4), 1979.

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

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

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

[15] A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

Appendix A - Proofs

Proposition 1. $\mathcal{C}Str \subsetneq \mathcal{WA}\text{-}Str$ and $\mathcal{SC} \nparallel \mathcal{WA}\text{-}Str$.

PROOF. *(sketch)* $\mathcal{C}Str \subseteq \mathcal{WA}\text{-}Str$ derives from the fact that the firing graph $\Gamma(\Sigma)$ is contained in the chase graph $G_c(\Sigma)$ used by c-stratification. The containment is strict as it is possible to define constraints which are WA-stratified and do not satisfy the c-stratification criterion (see, for instance, the set of constraints Σ_4 of Example 4).
The relationship $\mathcal{SC} \nparallel \mathcal{WA}\text{-}Str$ derives from the fact that there are examples belonging to one of these classes, but not to the other. For instance, the following set Σ is safe, but not WA-stratified:

$$\alpha: S(x_2, x_3) \wedge R(x_1, x_2, x_3) \rightarrow \exists y\, R(x_2, y, x_1)$$
$$\beta: R(x_1, x_2, x_3) \rightarrow S(x_1, x_3)$$

On the other hand, Σ_4 is WA-stratified but not safe. □

Lemma 1. Let $r_1: \phi_1 \rightarrow A_1 \wedge \cdots A_k$ and $r_2: B_1 \wedge \cdots B_n \rightarrow \psi_2$ be two TGDs. The problem of checking whether $r_1 < r_2$ is bounded by $O((k+1)^n)$.

PROOF. *(sketch)* We can define a database instance D consisting of body atoms $B_1 \cdots B_n$ (assuming that variables denote constants), and define a set of atoms satisfying firing conditions S and homomorphisms h_1, h_2 so that atoms in $B_1, ..., B_n$ unify with some of head atoms $A_1 \cdots A_k$. Let us denote with $k_i = |\psi_1(B_i)|+1$ the number of atoms in the head of r_1 unifying with the atom B_i appearing in the body of r_2 plus one (representing that B_i may not unify with any head atom). Then, we have k_1 choices for unification of B_1, $k_1 \times k_2$ choices for the sequence B_1, B_2 and so on. Finally, for the sequence $B_1, ..., B_n$ we have $\prod_{i=1}^{n} k_i$ choices. Thus, the number of choice arcs in the exploration tree is bounded by $O(\sum_{j=1}^{n}(\prod_{i=0}^{j-1} k_i) \times k_j)$, where $k_0 = 1$. The complexity of defining the sets $\psi_1(B_i)$, for $i \in [1..n]$, is bounded by $n \times k$. Since each k_i is bounded by $O(k+1)$, the global complexity is $O(k \times n + (k+1)^n) = O((k+1)^n)$. □

Proposition 2. Let Σ be a set of TGDs, D be a database. Then:

- the problem of checking whether Σ is WA-stratified (resp. SC-stratified, SwA-stratified) is bounded by $O(C_m \times |\Sigma|^2)$;
- if Σ is WA-stratified (resp. SC-stratified, SwA-stratified), the length of every chase sequence of Σ over D is polynomial in the size of D.

PROOF. *(sketch)*

1. (i) The cost of checking whether $r_i < r_j$ is denoted by C_{ij}.
   (ii) The cost of constructing the firing graph is bounded by $O(\sum_{r_i, r_j \in \Sigma} C_{ij}) = O(|\Sigma|^2 \times C_m)$ where $C_m = max\{C_{ij} | r_i < r_j\}$.
   (iii) The detection of strongly connected components in the firing graph is bounded by $O(|\Sigma|^2)$. Consequently, checking whether Σ is WA-stratified (resp. SC-stratified, SwA-stratified) is bounded by $O(C_m \times |\Sigma|^2)$.
2. Σ can be partitioned into $\Sigma_1, ..., \Sigma_k$ with $k \leq |\Sigma|$, where each Σ_i consists of the constraints belonging to a strongly connected component of $\Gamma(\Sigma)$. For any database D, the universal solutions of (D, Σ) can be computed by taking one Σ_i at time, following the topological order of the firing graph $\Gamma(\Sigma)$. Since each Σ_i is weakly acyclic (resp. safe, super-weakly acyclic), the length of all chase sequences of Σ_i over D is polynomial in the size of D and, therefore, the length of every chase sequence of Σ over D is polynomial in the size of D. □

Theorem 3.

1. $\mathcal{WA}\text{-}Str \subsetneq \mathcal{SC}\text{-}Str \subsetneq \mathcal{SwA}\text{-}Str$,
2. for $C \in \{WA, SC, SwA\}$, $\mathcal{C} \subsetneq \mathcal{C}\text{-}Str$ and
3. $\mathcal{SR} \nparallel \mathcal{SwA}\text{-}Str$ and $\mathcal{IR} \nparallel \mathcal{SwA}\text{-}Str$.

PROOF.

1. It follows from the fact that $\mathcal{WA} \subsetneq \mathcal{SC} \subsetneq \mathcal{SwA}$.
2. It follows from the fact that the $C\text{-}Str$ criterion first divides the set of constraints Σ into subsets (strongly connected components of the firing graph) and then checks the specific criterion C on each subset.
3. It is possible to find examples of sets of constraints that are in $\mathcal{SR}$ (and $\mathcal{IR}$) but not in $\mathcal{SwA}\text{-}Str$ and viceversa (see Examples 8 and 9 in Appendix B). □

Proposition 3. For every set of TGDs Σ and for every database D

- the problem of checking whether Σ is locally stratified is bounded by $O(|\Sigma|^2 \times C_m)$;
- if Σ is locally stratified, the length of every chase sequence of Σ over D is polynomial in the size of D.

PROOF. *(sketch)*

1. From the polynomial complexity of the function $Move$ [12] and the complexity of building the firing graph, bounded by $O(|\Sigma|^2 \times C_m)$, it follows that the complexity of building the trigger graph $\Delta(\Sigma)$ is bounded by $O(|\Sigma|^2 \times C_m)$. Since the complexity of checking whether the trigger graph $\Delta(\Sigma)$ is acyclic has a cost bounded by $O(|\Sigma|^2)$, the global complexity is bounded by $O(|\Sigma|^2 \times C_m)$.
2. The proof follows the one of Proposition 2. □

Theorem 4. $\mathcal{SwA}\text{-}Str \subsetneq \mathcal{LS}$ and $\mathcal{IR} \subsetneq \mathcal{LS}$.

PROOF. *(sketch)* From the definitions of SwA and LS criteria it follows that i) $MOVE(\Sigma, Out(r, y)) \subseteq Move(\Sigma, Out(r, y))$ and ii) if $Q' < Q$ then $Q \sqsubseteq Q'$. As a consequence, $\mathcal{SwA}\text{-}Str \subseteq \mathcal{LS}$.
Assume that Σ is *SR* and that $\langle G'(\Sigma), P\rangle$ is the associated 2-restricted system. If $r_i \hookrightarrow r_j$ (i.e. $(r_i, r_j) \in \Delta(\Sigma)$), then there exists a path from r_i to r_j in $G'(\Sigma)$ because, when we construct the set $MOVE(\Sigma, Out(r_i, y))$, we take into account both i) the firing relation $<$ between places, and ii) the propagation of null values introduced in the position associated with the existentially quantified variable y.
If the relation $\hookrightarrow$ is cyclic, then there exists a cycle in $G'(\Sigma)$ containing the constraints involved in the cycle in $\Delta(\Sigma)$ and, consequently, if the constraints are not locally stratified, then they are neither SwA and nor safe. Moreover, the constraints appearing in the cycle also belong to the same partition defined by *IR*. As a consequence, $\mathcal{IR} \subseteq \mathcal{LS}$. To show that the containments are strict it is sufficient to consider the set of constraints Σ_5 which is in $\mathcal{LS}$, but neither in $\mathcal{IR}$ nor in $\mathcal{SwA}\text{-}Str$. □

Corollary 1. For $C \in \{WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$, $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C}$.

PROOF. Obviously, $\mathcal{C} \subseteq Adn\text{-}\mathcal{C}$ for $C \in \{WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$. Moreover, Σ_1 is in $Adn\text{-}\mathcal{C}$, but not in $\mathcal{C}$, for all $C \in \{WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$. □

Lemma 2. For every set of TGDs Σ the function Adn++ always terminates.

PROOF. *(Sketch).* Termination of the rewriting algorithm is guaranteed by the use of substitutions which collapse adorned dependencies deriving from the same source constraint. □

Next the definition of *core chase step* is recalled, as it is used in the proof of Theorem 5.

DEFINITION 7. *(Core chase step)[8]* Let Σ be a set of TGDs and let I, J, K be three database instances defined over a database schema $\mathbf{R}$. We say that I is derived from J through a *parallel chase step* and write $J \xrightarrow{\Sigma} I$ if i) $J \not\models \Sigma$ and ii) $I = \bigcup_{r\in\Sigma, J \xrightarrow{r,h} J'} J'$. Moreover, we say that K is derived from J through a *core chase step* and write $J \xrightarrow{\Sigma\downarrow} K$ if $J \xrightarrow{\Sigma} I$ and $K = core(I)$. □

The definition of core chase sequence derives from the chase sequence by using a core chase step instead of a chase step. Core chase sequences are determined up to isomorphism. In [8] it has been shown that if I is an instance and Σ is a set of TGDs and EGDs, then there exists a universal model for Σ and I iff the core chase of I with Σ terminates and yields such a model.

Theorem 5. For every set of TGDs Σ over a database schema $\mathbf{R}$, $\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn{+}{+}(\mathbf{R},\Sigma), Adn{+}{+}(\Sigma)[1]\rangle$, where $Adn{+}{+}(\mathbf{R},\Sigma) = \mathbf{R} \cup \{p^\alpha(A_1,...,A_n) \mid p(A_1,...,A_n) \in \mathbf{R} \wedge p^\alpha$ *appears in* $Adn{+}{+}(\Sigma)[1]\} \cup \hat{\mathbf{R}}$.

PROOF. *(Sketch).* We have to prove that for every database D over $\mathbf{R}$, $USol(D, Map(\Sigma))[\hat{\mathbf{R}}] = USol(D, Adn{+}{+}(\Sigma)[1])[\hat{\mathbf{R}}]$, that is for every database $J \in USol(D, Map(\Sigma))$ there is a database $K \in USol(D, Adn{+}{+}(\Sigma)[1])$ such that $J[\hat{\mathbf{R}}] = K[\hat{\mathbf{R}}]$ and vice versa. To simplify the notation we shall use Σ^α denoting $Adn{+}{+}(\Sigma)[1]$ and $\bar{\Sigma}$ to denote $Map(\Sigma)$.

- (Base case - Step 0) Let $D \xrightarrow{\Sigma^\alpha\downarrow} K_0$, we have that $Adn^{-1}(K_0) = J_0 = D$ and $J_0[\hat{\mathbf{R}}] = K_0[\hat{\mathbf{R}}] = \emptyset$.
- (Inductive case - Step i) Assume that J_{i-1} and K_{i-1} are s.t. $J_{i-1} \subseteq Adn^{-1}(K_{i-1})$, $J_{i-1}[\hat{\mathbf{R}}] = K_{i-1}[\hat{\mathbf{R}}]$ (up to nulls renaming).

  Let $J_{i-1} \xrightarrow{\bar{\Sigma}} J'_i$, $K_{i-1} \xrightarrow{\Sigma^\alpha} K'_i$ and $J_i = core(J'_i)$, $K_i = core(K'_i)$. We have that $J_i \subseteq Adn^{-1}(K_i)$ and $J_i[\hat{\mathbf{R}}] = K_i[\hat{\mathbf{R}}]$ since the set of retracted tuples derived by means of Σ^α is contained (up to variable renaming) in the set of retracted tuples derived by means of $\bar{\Sigma}$. Indeed, two tuples $p(t_1)$ and $p(t_2)$ in J'_i may 'correspond' to tuples $p^{\alpha_1}(t_1)$ and $p^{\alpha_2}(t_2)$ in K'_i and therefore, $p(t_1)$ and $p(t_2)$ may be isomorphic, while $p^{\alpha_1}(t_1)$ and $p^{\alpha_2}(t_2)$ are not isomorphic because of the different adornments. On the other side if there are two isomorphic tuples in K'_i, the 'corresponding' tuples in J'_i are isomorphic as well. However, since the relations in $J'_i[\hat{\mathbf{R}}]$ and $K'_i[\hat{\mathbf{R}}]$ 'eliminate' adornments for any two isomorphic tuples in $J'_i[\hat{\mathbf{R}}]$ the 'corresponding' tuples in $K'_i[\hat{\mathbf{R}}]$ are isomorphic too, and vice versa. □

In the following proof we shall use the function $Adn^{-s}(\cdot)$ which receives in input an adorned first order formula F, where free symbols are of the form f_i, and gives in output the first order formula derived from F by replacing every symbol f_i with f. The function can be trivially extended to sets of first order formulae.

Theorem 6. $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C} \subsetneq Adn{+}{+}\mathcal{C}$, for $C \in \{WA, SC, Str, CStr, SwA, SR, IR, WA\text{-}Str, SC\text{-}Str, SwA\text{-}Str, LS\}$.

PROOF. *(sketch)* $\mathcal{C} \subsetneq Adn\text{-}\mathcal{C}$ has been stated in Theorem 2 ([10]) and Corollary 1. To show that $Adn\text{-}\mathcal{C} \subsetneq Adn{+}{+}\mathcal{C}$, we first show $Adn\text{-}\mathcal{C} \subseteq Adn{+}{+}\mathcal{C}$ by contradiction. Suppose that there is a Σ and a criterion C such that $Adn(\Sigma) \in \mathcal{C}$ and $Adn{+}{+}(\Sigma)[1] \notin \mathcal{C}$. Moreover, since $Adn^{-s}(Adn{+}{+}(\Sigma)[1]) \subseteq Adn(\Sigma)$, every criterion for chase termination C verified by $Adn{+}{+}(\Sigma)[1]$ is verified by $Adn(\Sigma)$ as well. The strict containment, for all criteria considered, is proved by the fact that the set of constraints Σ_6 is in $Adn{+}{+}\mathcal{WA}$, but not in $Adn\text{-}\mathcal{LS}$. □

Theorem 7. $\mathcal{AC} = Adn{+}{+}\mathcal{WA} = Adn{+}{+}\mathcal{LS}$.

PROOF. *(sketch)* To show $Adn{+}{+}\mathcal{WA} = Adn{+}{+}\mathcal{LS}$ it is sufficient to demonstrate that $Adn{+}{+}\mathcal{WA} \supseteq Adn{+}{+}\mathcal{LS}$, as $Adn{+}{+}\mathcal{WA} \subseteq Adn{+}{+}\mathcal{LS}$ is trivial, that is if there is a Σ such that $Adn{+}{+}(\Sigma)[1] \notin \mathcal{WA}$, then $Adn{+}{+}(\Sigma)[1] \notin \mathcal{LS}$ also holds. Indeed, for any $\Sigma^\alpha = Adn{+}{+}(\Sigma)[1]$, $\Sigma^\alpha \notin \mathcal{WA}$ means that there is a cycle in $dep(\Sigma^\alpha)$ with a special edge. It can be easily shown that in such a case there is a cycle in the firing graph $\Gamma(\Sigma^\alpha)$ and a cycle in the trigger graph $\Delta(\Sigma)$ as well.
We show now that $Adn{+}{+}\mathcal{WA} = \mathcal{AC}$. We first consider the case of $\mathcal{AC} \subseteq Adn{+}{+}\mathcal{WA}$ showing that for any Σ in $\mathcal{AC}$ the set of constraints $Adn{+}{+}(\Sigma)[1]$ is weakly acyclic. In fact, as said above, a cycle in $dep(\Sigma^\alpha)$ means that there is a cycle C in $\Gamma(\Sigma^\alpha)$ where nulls are created and propagated. This means that in the rewriting there is a constraint belonging to C which is used more than once and, therefore, a substitution is employed to avoid an infinite number of adorned constraints. Moreover, each predicate symbol in the (head of a TGD contained in the) path C depends on the other predicate symbols. Consequently, in such a case $Adn{+}{+}(\Sigma)[2] = true$. Viceversa, $Adn{+}{+}\mathcal{WA} \subseteq \mathcal{AC}$ holds as for any $\Sigma \notin \mathcal{AC}$ we have that $\Sigma \notin Adn{+}{+}\mathcal{WA}$. $Adn{+}{+}(\Sigma)[2] = true$ means that a cycle where nulls may be propagated has been detected; this implies that $\Gamma(\Sigma^\alpha)$ is cyclic and that $dep(\Sigma^\alpha)$ contains a cycle with a special arc. □

Appendix B - Additional examples

The following two examples show that both SR and IR criteria are not comparable with $SwA\text{-}Str$.

EXAMPLE 8. Consider the below set of constraints Σ_8:

$$r_1: N(x) \rightarrow \exists y \exists z\, E(x,y) \wedge S(z,y)$$
$$r_2: E(x,y) \wedge S(x,y) \rightarrow N(y)$$
$$r_3: E(x,y) \rightarrow E(x,x)$$

The set Σ_8 is *SR* (and, obviously *IR*), but not $SwA\text{-}Str$. Indeed, $r_1 < r_3 < r_2 < r_1$ and Σ_8 is not SwA. □

EXAMPLE 9. Consider the following set of constraints Σ'_9:

$$r_1: N(x) \rightarrow \exists y \exists z\, E(x,y,z)$$
$$r_2: E(x,y,z) \rightarrow T(x,y,z)$$
$$r_3: T(x,y,y) \rightarrow N(y)$$

The set Σ'_9 is not *IR* (and, obviously, *SR*) since $r_1 <_P r_2 <_P r_3 <_P r_1$, where $P = \{E_2, E_3, T_2, T_3, N_1, E_1, T_1\}$ and the unique component is not safe (i.e. $N_1 \xrightarrow{*} E_2$, $E_2 \rightarrow T_2$, $T_2 \rightarrow N_1$). On the other hand, the below set of constraints Σ"$_9$

$$r_4: R(x,y) \wedge R(y,x) \rightarrow \exists u \exists v\, R(x,u), R(u,v), R(v,x)$$

is not SwA, but it is stratified and, therefore, safely restricted. The set of constraints $\Sigma_9 = \Sigma'_9 \cup \Sigma$"$_9$, is neither IR, nor SwA, but it belongs to the class of $\mathcal{S}w\mathcal{A}\text{-}\mathcal{S}tr$ constraints. □

EXAMPLE 10. Recall the set of constraints Σ_6 of Example 6:

$$r_1 : R(x) \rightarrow S(x,x)$$
$$r_2 : S(x_1,x_2) \rightarrow \exists\, z\, T(x_2,z) \wedge Q(x_2)$$
$$r_3 : T(x_1,x_2) \wedge T(x_1,x_3) \wedge T(x_3,x_1) \rightarrow R(x_2)$$

The set $Adn(\Sigma_6)$ is as follows:

$$s_1 : R^b(x) \rightarrow S^{bb}(x,x)$$
$$s_2 : S^{bb}(x_1,x_2) \rightarrow \exists\, z\, T^{bf}(x_2,z) \wedge Q^b(x_2)$$
$$s_3 : T^{bb}(x_1,x_2) \wedge T^{bb}(x_1,x_3) \wedge T^{bb}(x_3,x_1) \rightarrow R^b(x_2)$$
$$s_4 : T^{bf}(x_1,x_2) \wedge T^{bb}(x_1,x_3) \wedge T^{bb}(x_3,x_1) \rightarrow R^f(x_2)$$
$$s_5 : R^f(x) \rightarrow S^{ff}(x,x)$$
$$s_6 : S^{ff}(x_1,x_2) \rightarrow \exists\, z\, T^{ff}(x_2,z) \wedge Q^f(x_2)$$
$$s_7 : T^{bf}(x_1,x_2) \wedge T^{ff}(x_1,x_3) \wedge T^{ff}(x_3,x_1) \rightarrow R^f(x_2)$$
$$s_8 : T^{ff}(x_1,x_2) \wedge T^{ff}(x_1,x_3) \wedge T^{ff}(x_3,x_1) \rightarrow R^f(x_2)$$

$Adn(\Sigma_6)$ is not in $\mathcal{LS}$ as constraints s_5, s_6, s_8 reproduce the same structure of the source constraints which, as discussed in Example 6, are not in $\mathcal{LS}$. □

EXAMPLE 11. Considering the below set of constraints:

$$c_1 : R(x,z) \rightarrow \exists y\, T(x,y)$$
$$c_2 : T(x,y) \rightarrow R(x,y)$$

the current rewriting technique Adn++ generates the dependencies:

$$r_1 : R^{bb}(x,z) \rightarrow \exists y\, T^{bf_1}(x,y)$$
$$r_2 : T^{bb}(x,y) \rightarrow R^{bb}(x,y)$$
$$r_3 : T^{bf_1}(x,y) \rightarrow R^{bf_1}(x,y)$$
$$r_4 : R^{bf_1}(x,z) \rightarrow \exists y\, T^{bf_1}(x,y)$$

which are acyclic and, therefore, terminating, whereas the technique Adn+ proposed in [10] generates the below constraints:

$$s_1 : R^{bb}(x,z) \rightarrow \exists y\, T^{bf_1}(x,y)$$
$$s_2 : T^{bb}(x,y) \rightarrow R^{bb}(x,y)$$
$$s_3 : T^{bf_1}(x,y) \rightarrow R^{bf_1}(x,y)$$
$$s_4 : R^{bf_1}(x,z) \rightarrow \exists y\, T^{f_1f_2}(x,y)$$
$$s_5 : T^{f_1f_2}(x,y) \rightarrow R^{f_1f_2}(x,y)$$
$$s_6 : R^{f_1f_2}(x,z) \rightarrow \exists y\, T^{f_2f_3}(x,y)$$
$$s_7 : T^{f_2f_3}(x,y) \rightarrow R^{f_2f_3}(x,y)$$
$$s_8 : R^{f_2f_3}(x,z) \rightarrow \exists y\, T^{f_3f_4}(x,y)$$
$$s_9 : T^{f_3f_4}(x,y) \rightarrow R^{f_3f_4}(x,y)$$
$$s_{10} : R^{f_3f_4}(x,z) \rightarrow \exists y\, T^{f_4f_5}(x,y)$$

At this point the rewriting procedure stops because was not able to detect the termination and returns a set of adorned constraints without subscripts. The output set does not give any advantage in terms of analysis of its structural properties. Therefore, the rewriting performed by the function Adn++ is more effective of the one presented in [10].