Fundamenta Informaticae 157 (2018) 221–270
DOI 10.3233/FI-2018-1627
IOS Press

Anatomy of the Chase

Gösta Grahne*
Concordia University, Montreal, Canada
grahne@cs.concordia.ca

Adrian Onet
Morgan Stanley, Montreal, Canada
adrian.onet@morganstanley.com

Abstract. A lot of research activity has recently taken place around the chase procedure, due to its usefulness in data integration, data exchange, query optimization, peer data exchange and data correspondence, to mention a few. As the chase has been investigated and further developed by a number of research groups and authors, many variants of the chase have emerged and associated results obtained. Due to the heterogeneous nature of the area it is frequently difficult to verify the scope of each result. In this paper we take closer look at recent developments, and provide additional results. Our analysis allows us create a taxonomy of the chase variations and the properties they satisfy.

Two of the most central problems regarding the chase is termination, and discovery of restricted classes of sets of dependencies that guarantee termination of the chase. The search for the restricted classes has been motivated by a fairly recent result that shows that it is undecidable (RE-complete, to be more precise) to determine whether the chase with a given dependency set will terminate on a given instance. There is a small dissonance here, since the quest has been for classes of sets of dependencies guaranteeing termination of the chase on *all* instances, even though the latter problem was not known to be undecidable. We resolve the dissonance in this paper by showing that determining whether the chase with a given set of dependencies terminates on all instances is unsolvable, and on level Π_2^0 in the Arithmetical Hierarchy. For this we use a reduction from word rewriting systems, thereby also showing the close connection between the chase and word rewriting. The same reduction also gives us the aforementioned instance-dependent RE-completeness result as a byproduct. For one of the restricted classes guaranteeing

*Address for correspondence: Department of Computer Science, Concordia University, Montreal, Canada.

Received July 2016; revised July 2017.

termination on all instances, the *stratified* sets dependencies, we provide new complexity results for the problem of testing whether a given set of dependencies belongs to it. These results rectify some previous claims that have occurred in the literature.

Keywords: Chase, Date Exchange, Data Repair, Incomplete databases, Undecidability Complexity

1. Introduction

The chase procedure was initially developed by [1] for testing logical implication between sets of dependencies, by [2, 3] for determining equivalence of database instances known to satisfy a given set of dependencies, and by [4] to determine query equivalence under database constraints. Recently the chase has experienced a revival due to its application in data integration, data-exchange, data-repair, query optimization, ontologies and data correspondence. In this paper we will focus on constraints in the form of embedded dependencies specified by sets of tuple and equality generating dependencies as defined by [5].

As an example consider a database instance $I = \{$ *Student(S01, john), Affiliation(S01,MIT), Affiliation(S02,MIT), ParkingResv(S02, mike, R03)* $\}$ over a student database containing personal information *Student(sid,name)*, affiliation *Affiliation(sid, univ)*, and parking reservation *ParkingResv(sid, name, spot)* . Consider also the foreign key constraint stating that each *sid* value in the *Affiliation* relation needs to be a valid student id, that is. it needs to also occur in the *Student* relation. This foreign key constraint is expressed using tuple generating dependency (tgd):

$$\forall sid\ \forall univ\ \big(Affiliation(sid,univ) \rightarrow \exists name\ \ Student(sid,name)\big).$$

In this example tuple *Affiliation(S02,MIT)* violates this constraint because student id *S02* is not part of the *Student* relation. In this case the chase step will simply add to instance I a new tuple *Student(S02,* x*)*, where x represents an unknown value. Consider also the following constraint that states that the student name is used consistently in *Student* and *ParkingResv* relations:

$$\forall sid\ \forall n1\ \forall n2\ \forall spot\ \ \big(Student(sid,n1), ParkingResv(sid, n2, spot) \rightarrow n1{=}n2\big).$$

In our example the newly added tuple *Student(S02,* x*)* together with the tuple *ParkingResv(S02, mike, R03)* violate the constraint because student id *S02* is associated with unknown value x and value *mike*. In this case the chase procedure will change value x in the first tuple to *mike*.

As the research on chase has progressed, several variations of the chase have evolved. As a consequence it has become difficult to determine the scope of results obtained.

As a remedy we scrutinize the four most important chase variations, both deterministic and nondeterministic, namely the *standard*, *oblivious*, *semi-oblivious*, and *core* chase. We will analyze, for each of these chase variations, the data and combined complexity of testing if the chase step is applicable for a given instance and tgd. The data complexity measures the required computation time as a function of the number of tuples in the instance, while the combined complexity also takes the size

of the dependency set into account. It didn't come as a surprise that the oblivious and semi-oblivious chase variations share the same complexity, and that the standard chase has a slightly higher complexity. The table below shows the data and combined complexities for the following problem: given an instance with n tuples and a tgd $\alpha \rightarrow \beta$, is the core/standard/oblivious/semi-oblivious chase step applicable? By $|\alpha|$ we mean the number of atoms in α, and similarly for $|\beta|$. Note that in the case of egds the chase applicability complexity is the same as of the oblivious tgd chase step.

Chase Step	Data Complexity	Combined Complexity
standard/core	$O(n^{\|\alpha\|+\|\beta\|})$	Σ_2^{P}-complete
oblivious/semi-oblivious	$O(n^{\|\alpha\|})$	NP-complete

Thus, at a first look the oblivious and semi-oblivious chase procedures will be a more appropriate choice when it comes to a practical implementation. Still, as we will show, the lower complexity comes with a price, that is the higher the complexity for a chase variation the more "likely" it is that the chase process terminates for a given instance and set of dependencies. On the other hand, [6] showed that the core chase is complete in finding universal models, meaning that a universal model exists if and only if the core chase terminates.

We next compare the semi-oblivious and standard chase when it comes to the termination problem. With this we show that:

- The standard and semi-oblivious chases are not distinguishable by most classes of dependencies developed to ensure the standard chase termination.
- The number of semi-oblivious chase steps needed to terminate remains the same as for the standard chase, namely polynomial.

This raises the following question:

- What makes a class of dependency sets that ensures termination for *all* input instances under the standard chase, also ensure termination for the semi-oblivious chase as well?

We answer this question by giving a sufficient syntactical condition for a set of dependencies to also guarantee the semi-oblivious chase termination. As we will see, most of the techniques developed to ensure termination of the standard chase indeed ensure termination of the computationally less expensive semi-oblivious and oblivious chases as well. We also provide a classification of termination behaviour for the various chase variations.

It has been known for some time, from [6, 7, 8], that it is undecidable to determine if the chase with a given set of tgds terminates on a given instance. This has spurred a quest for restricted classes of tgds

guaranteeing termination. Interestingly, these all guarantee *uniform* termination, that is, termination on *all* instances. This, even though it was only known that the problem is undecidable for a given instance. We remediate the situation by proving that (perhaps not too surprisingly) the uniform version of the termination problem is undecidable as well, and show that it is not recursively enumerable.

- We show that determining whether the core chase with a given set of dependencies terminates on all instances is Π_2^0-complete in the Arithmetical Hierarchy.

We achieve this using a reduction from the uniform termination problem for word-rewriting systems (semi-Thue systems). As a byproduct we obtain the result from [6] showing that testing if the core chase terminates for a given instance and a given set of dependencies is RE-complete. We will also show also that the same complexity result holds for testing whether the standard chase with a set of dependencies is terminating on at least one execution branch. Next we will show that by using a single denial constraint (a "headless" tgd) in our reduction, the same complexity result holds also for the standard chase termination on all instances on all execution branches. We note that after the first version of this paper was published as [9], it was shown in [10] that the coRe-completeness result holds even without considering denial constraints.

Many of the restricted classes guaranteeing termination rely on the notion of a set Σ of dependencies being *stratified*. Stratification involves two conditions, one determining a partial order between tgds in in Σ, and the other on Σ as a whole. It has been claimed by [6] that testing the partial order between tgds is in NP.

- We show that testing the partial order between tgds cannot be in NP (unless NP=coNP), by proving that the problem is at least coNP-hard.

We also prove a Δ_2^{p} upper bound for the problem. Finding matching upper and lower bounds remains an open problem.

Summary of contributions and paper outline

Our contributions can be summarized as follows:

1. We highlight four different different chase variations and analyze for each the complexity of testing whether a dependency is applicable during the chase (detailed in Section 2 and Section 3).

2. We study the different chase variations in light of their termination behaviour for various classes of dependencies (Section 4).

3. We provide a classification of termination criteria proposed in the literature, and provide the missing analysis of the question of when a chase variation is guaranteed to terminate for *all instances* on *some* or on *all* branches. We define termination classes for each of the termination criteria and determine their relationship (Section 4).

4. By a reduction from string rewriting systems we show that the most general termination problem is higly undecidable. The previously known more restricted undecidability results in the literature follow as special cases from our reduction (Section 5).

5. We re-examine classes of dependencies designed to guarantee chase termination in light of our taxonomy of termination behaviour (Section 6).

6. We provide complexity results for testing whether a set of dependencies belongs to the stratified class (Section 7).

7. We provide in the Appendix uniform definitions of the various chase variations, termination classes, and dependency classes with guaranteed termination.

This papers only focuses on constraints represented by finite sets of tgds and egds, which covers most of the constraints used in practice today. Not covered here is a whole body of work involving more general classes of constraints, where tgds are extended by allowing negated atoms in the body and disjunctions of atoms in the head (e.g. the work of [11, 12]) and also by allowing disjunctions between the atoms in the body (e.g. the work of [6]).

Related work

The complexity of the standard-chase termination problem was tackled first by Deutsch et al. in [6] and show that the problem is RE-complete where the chased instance is given and the set of dependencies consists only of tgds. The same results hold for core-chase termination problem [6]. Later on Marnette [8] showed that the RE-completeness result holds also for oblivious-chase termination problem for a given instance. In 2014 Gogacz and Marcinkowski [13] showed that the all-instances termination problem was RE-complete for the semi-oblivious and oblivious chase variations as well. Much work was also done in finding classes of dependencies that guarantees the standard-chase termination for all instances [14, 15, 16, 6, 7, 17, 8, 18, 19, 20].

2. Preliminaries

For basic definitions and concepts we refer to [21]. We will consider the complexity classes PTIME, NP, coNP, DP, Δ_2^{P}, RE, coRE, and the first few levels of the Polynomial and Arithmetical Hierarchies. For the definitions of these classes we refer to [22, 23]. We start with some preliminary notions. We will use the symbol $\subseteq$ for the subset relation, and $\subset$ for proper subset. A function f with a finite set $\{x_1, \ldots, x_n\}$ as domain, and where $f(x_i) = a_i$, will be described as $\{x_1/a_1, \ldots, x_n/a_n\}$. The reader is cautioned that the symbol $\rightarrow$ will be overloaded; the meaning should however be clear from the context.

Relational schemas and instances. A *relational schema* is a finite set $\mathbf{R} = \{R_1, \ldots, R_n\}$ of relational symbols R_i, each with an associated positive integer $arity(R_i)$. Let Cons be a countably infinite set of constants, usually denoted $a, b, c, \ldots$, possibly subscripted, and let Nulls be a countably infinite set of nulls denoted $x, y_1, y_2, \ldots$. A *relational instance* I over a schema $\mathbf{R}$ is a function that associates for each relational symbol $R \in \mathbf{R}$ a finite subset R^I of $(\mathsf{Cons} \cup \mathsf{Nulls})^{arity(R)}$.

A *relational atom* is an expression of the form $R(\bar{x})$, where $R \in \mathbf{R}$, and $\bar{x}$ is a sequence of nulls and constants, and the sequence is of length $arity(R)$. If the sequence $\bar{x}$ contains only constants, we denote it $\bar{a}$, and call $R(\bar{a})$ a *ground atom*.

We shall frequently identify an instance I with the set $\{R(\bar{x}) : (\bar{x}) \in R^I, R \in \mathbf{R}\}$ of atoms, assuming appropriate lengths of the sequence $\bar{x}$ for each $R \in \mathbf{R}$. By the same convenience, the atoms $R(x_1 \ldots, x_k)$ will sometimes be called *tuples* of relation R^I and denoted $t, t_1, t_2, \ldots$. By $dom(I)$ we mean the set of all constants and nulls occurring in the instance I, and by $|I|$ we mean the number of tuples in I.

Homomorphisms. Let I and J be instances, and $h : dom(I) \to dom(J)$ a mapping that is the identity on the constants. We extend h to tuples $(\bar{x}) = (x_1, \ldots, x_k)$ by $h(x_1, \ldots, x_k) = (h(x_1), \ldots, h(x_k))$. By our notational convenience we can thus write $h(\bar{x})$ as $h(R(\bar{x}))$, when $(\bar{x}) \in R^I$. We extend h to instances by $h(I) = \{h(t) : t \in I\}$. If $h(I) \subseteq J$ we say that h is a *homomorphism* from I to J. If $h(I) \subseteq I$, we say that h is an *endomorphism*. If $h(I) \subseteq J$, and the mapping h is a bijection, and if also $h^{-1}(J) = I$, the two instances are *isomorphic*, which we denote $I \cong J$. If both $h(I) \subseteq J$, and $g(J) \subseteq I$, for some homomorphisms h and g, we say that I and J are *homomorphically equivalent*. Note that isomorphic instances are homomorphically equivalent, but not vice versa.

A subset I' of I is said to be a *core* of I, if there is a endomorphism h, such that $h(I) \subseteq I'$, and there is no endomorphism g such that $g(I') \subset I'$. It is well known that all cores of an instance I are isomorphic, so for our purposes we can consider the core unique, and denote it $core(I)$. The core of an instance I plays an important role as it is the smallest instance that is homomorphically equivalent to I.

Tuple generating dependencies. A *tuple generating dependency (tgd)* is a first order formula of the form

$$\forall \bar{x}, \bar{y} \left(\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z}) \right),$$

where α and β are conjunctions of relational atoms, and $\bar{x}$,$\bar{y}$ and $\bar{z}$ are sequences of variables. Occasionally we will abuse notation and write $x \in \bar{x}$ to mean that x occurs in the sequence. We assume that the variables occurring in tgds come from a countably infinite set Vars disjoint from Nulls. We also allow constants from Cons in the tgds. In the formula we call α the *body* of the tgd. Similarly we refer to β as the *head* of the tgd. If there are no existentially quantified variables the dependency is said to be *full*.

When α is the body of a tgd and $h : \mathsf{Vars} \cup \mathsf{Const} \to \mathsf{Nulls} \cup \mathsf{Const}$ is a (partial) mapping that is identity on constants, we shall conveniently regard the set of atoms in α as an instance I_α, and write $h(\alpha)$ for the set $h(I_\alpha)$. Then h is a homomorphism from α to an instance I, if $h(\alpha) \subseteq I$.

Frequently, we omit the universal quantifiers in tgd formulas. Also, when the variables and constants are not relevant in the context, we denote a tuple generating dependency $\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z})$ simply as $\alpha \to \beta$. Tgds will be denoted by the letter ξ, possibly subscripted.

Let $\xi = \alpha \to \beta$ be a tuple generating dependency, and I be an instance. Then we say that I *satisfies* ξ, if $I \models \xi$ in the standard model theoretic sense (see e.g. [24]), or equivalently, if for every homomorphism $h : \mathsf{Vars} \cup \mathsf{Const} \to \mathsf{Nulls} \cup \mathsf{Const}$, such that $h(\alpha) \subseteq I$, there is an extension h' of h, such that $h'(\beta) \subseteq I$.

Equality generating dependencies. An *equality generating dependency (egd)* is a first order formula of the form

$$\forall \bar{x} \big(\alpha(\bar{x}) \to x = y \big),$$

where α is a conjunctions of relational atoms, $\bar{x}$ is a sequence of variables and x, y are from $\bar{x}$. We assume that the variables occurring in egds come from a countably infinite set Vars disjoint from Nulls. In the formula we call α the *body* of the egd.

Similarly to the tgd case, frequently, we omit the universal quantifiers in egd formulas. Also, when the variables and constants are not relevant in the context, we denote an equality generating dependency $\alpha(\bar{x}) \rightarrow x = y$ simply as $\alpha \rightarrow x = y$.

Let $\xi = \alpha \rightarrow x = y$ be an equality generating dependency, and I be an instance. Then we say that I *satisfies* ξ, if $I \models \xi$ in the standard model theoretic sense (see e.g. [24]), or equivalently, if for every homomorphism $h : \mathsf{Vars} \cup \mathsf{Const} \rightarrow \mathsf{Nulls} \cup \mathsf{Const}$, such that $h(\alpha) \subseteq I$, it holds that $h(x) = h(y)$.

Universal Models. Given an instance I and a set of tgds Σ, a *model* of I and Σ is a database instance J such that there is a homomorphism from I to J, and J satisfies Σ. A *universal model* of I and Σ is a *finite* model of I and Σ that has a homomorphism into *every* model of I and‘Σ. Universal models are crucial in *data exchange*, where they are used to materialize the data imported into a target database from a source database. In this context the constraints Σ describe the relationship between the source schema and the target schema,

The Chase. The chase is a procedure that takes as input and instance I and a set of constraints Σ, which in the present paper will be a finite set of tgds and egds. The chase might not terminate or it might *fail*, but if it does terminate successfully (without failing) it produces a finite instance J, such that

1. $J \models \Sigma$.
2. There is a homomorphism from I to J.
3. For every (finite or infinite) instance K, if there is a homomorphism from I to K and $K \models \Sigma$, then there is a homomorphism from J to K.

The instance J produced by the chase on I with Σ is called a *canonical model* of I and Σ [14]. In the same paper it is shown that each canonical model is also a universal model. The reverse does not always hold.

For a formal definition of the chase, let Σ be a (finite) set of tgds and egds, and I an instance. A *trigger* for Σ on I is a pair (ξ, h), where ξ is either a tgd or an egd, and h is a homomorphism from α to I, i.e. $h(\alpha) \subseteq I$. If ξ is a tgd $\alpha \rightarrow \beta$, the trigger (ξ, h) is said to be *active* if there is no extension h' of h, such that $h'(\beta) \subseteq I$, and if ξ is an edg $\alpha \rightarrow x = y$, the trigger (ξ, h) is said to be active on I if $h(x) \neq h(y)$,

Let (ξ, h) be a trigger for Σ on I. To *fire* the trigger means:

- in case ξ is a tgd, transforming I into the instance $J = I \cup \{h'(\beta)\}$, where h' is a *distinct extension* of h, i.e. an extension of h that assigns new fresh nulls to the existential variables in β. By “new fresh” we mean the next unused element in some fixed enumeration of the nulls.

  The transformation is denoted $I \xrightarrow{(\xi,h)} J$, or just $I \rightarrow_\xi J$, if the particular homomorphism h and its distinct extension h' are irrelevant or understood from the context.

- in case ξ is an egd:
  - if both $h(x)$ and $h(y)$ are nulls, transforming I into instance J such that all occurrences of $h(y)$ are replaced with $h(x)$. Here we assume there is an enumeration of the variables, and that $h(x) < h(y)$ is this enumeration.
  - if one of $h(x)$ and $h(y)$ is a null and the other a constant, transforming I into instance J such that all occurrences of the null is replaced with the constant.
  - if both $h(x)$ and $h(y)$ are constants (note that $h(x) \neq h(y)$), then the firing of the trigger *fails* on I.

  In case the trigger fails we denote the transformation $I \xrightarrow{(\xi,h)} \perp$. Otherwise, the transformation is denoted $I \xrightarrow{(\xi,h)} J$, or just $I \rightarrow_\xi J$.

A (finite or infinite) sequence $I_0, I_1, I_2 \ldots$ of instances is said to be a *chase sequence with* Σ *originating from* I_0, if for all $n = 0, 1, 2, \ldots$ there is a trigger (ξ, h) on I_n, such that $I_n \xrightarrow{(\xi,h)} I_{n+1}$. If $I_n \xrightarrow{(\xi,h)} \perp$ for some I_n in the sequence we say that the sequence *fails*. Otherwise, the sequence is said to be *successful*. If there is an I_n in a successful sequence, such that there are no more active triggers for Σ on I_n, we say that the sequence *terminates*. Otherwise, the successful sequence does not terminate. A chase sequence can be generated by the following algorithm:

ALGORITHM *Standard-*$Chase_\Sigma(I)$

```
1  I_0 := I; i := 0;
2  if exists an active trigger (ξ, h) for I_i.
3     then
4          if I_i --(ξ,h)--> ⊥
5            then return FAIL
6            else I_i --(ξ,h')--> I_{i+1}; i := i + 1
7     else return I_i
8  goto 2
```

Note that there can be several chase sequences for a given Σ and I, as shown by [14]. This is reflected by the non-deterministic choice of a trigger at line 2. A successful chase sequence can be infinite, as in the following example.

Example 2.1. Consider instance $I = \{R(a, b)\}$ and tgd:

$$\xi \quad = \quad R(x, y) \rightarrow \exists z\, R(y, z).$$

One possible chase sequence is

$$
\begin{array}{c|cc}
I_0 = I & & \\
\hline
R & & \\
\hline
a & b &
\end{array}
\xrightarrow{(\xi,h_1)}
\begin{array}{c|cc}
I_1 & & \\
\hline
R & & \\
\hline
a & b & \\
b & x_1 &
\end{array}
\xrightarrow{(\xi,h_2)} \cdots \xrightarrow{(\xi,h_n)}
\begin{array}{c|cc}
I_n & & \\
\hline
R & & \\
\hline
a & b & \\
b & x_1 & \\
x_1 & x_2 & \\
 & \cdots & \\
x_{n-1} & x_n &
\end{array}
\xrightarrow{(\xi,h_{n+1})} \cdots
$$

As the next example shows there are cases when some chase sequences fail, while others are successful and do not terminate.

Example 2.2. Let $\Sigma = \{\xi_1, \xi_2, \xi_3\}$, where

$$
\begin{aligned}
\xi_1 &= R(x,y) \rightarrow T(y,x) \\
\xi_2 &= T(x,y) \rightarrow x = y \\
\xi_3 &= R(x,y) \rightarrow \exists z\, R(y,z).
\end{aligned}
$$

Let $I = \{R(a,b)\}$. One possible chase sequence is

$$\{R(a,b)\} \xrightarrow{(\xi_1,\{x/a,y/b\})} \{R(a,b), T(a,b)\} \xrightarrow{(\xi_2,\{x/a,y/b\})} \bot$$

Another possible chase sequence, that uses only ξ_3, namely

$$\{R(a,b)\} \xrightarrow{(\xi_3,\{x/a,y/b\})} \{R(a,b), T(a,x_1)\} \xrightarrow{(\xi_3,\{x/a,y/x_1\})} \cdots$$

will be infinite.

In order to avoid exhaustively choosing the same dependencies in the nondeterministic step 2 in the standard chase algorithm, as seen in Example 2.2, we introduce the notion of fairness of an infinite chase sequence. Fairness also guarantees that the structures constructed by the chase actually are models, and that all nondeterministic branches are executed until convergence (at the first infinite ordinal) or failure.

Definition 2.3. Let $I = I_0$ be an instance and Σ a set of tgds and egds. An infinite chase sequence $I_0, I_1, I_2 \ldots$ is said to be *fair*, if for all i and for all active triggers (ξ, h) for I_i, where $\xi \in \Sigma$, there exists $j \geq i$ such that either $I_j \xrightarrow{(\xi,h)} I_{j+1}$, or the trigger (ξ, h) is no longer active on I_j

From an algorithmic point of view the choice of next trigger to fire is essential. Based on this, the following variations of the chase process have been considered in the literature (for a comprehensive review of different chase variation see [25]).

1. *The standard chase* as presented by [14]. This is algorithm *Standard-*$Chase_\Sigma(I)$. In other words, the next trigger is chosen nondeterministically from the subset of current triggers that are *active*. As shown by [14], if all possible chase sequences are successful and terminate, then all the leaves in the tree of chase sequences are homomorphically equivalent. Thus the nondeterminism is of type "don't care," and we write $Chase_\Sigma(I) = J$, where J is a representative of this homomorphism class. It is also shown by [14], that if all sequences are successful, but some sequences are infinite, the results are nevertheless all homomorphically equivalent.[1]

2. *The oblivious chase* as presented by [7]. The next trigger is chosen nondeterministically from the set of all current triggers, active or not, but each trigger is fired only once in a chase sequence. The oblivious chase can be obtained from the algorithm for the standard chase, by changing line 2 to

   2 **if** exists a trigger (ξ, h) for I_i, and (ξ, h) has not been fired before

   It is shown by [7] that for Σ containing tgds only, if one oblivious chase sequence with Σ on an instance I terminates, so do all oblivious chase sequences with Σ on I, and furthermore, that all the leaves in the tree of sequences are isomorphic. Fairness, in the context of the oblivious chase means that all triggers are eventually fired. This can be achieved e.g. by placing the triggers (as they arise) in a First-In-First-Out queue, or, as shown by [7], by enumerating the triggers based on an enumeration of the constants, nulls, and dependencies.

   We note that the oblivious chase in [7] considers Skolemized versions of the tgds, where each existentially quantified variable is replaced by a so called Skolem function. For instance, the Skolemized version of the dependency $\xi = R(x, y) \rightarrow \exists z, w\ S(x, z, w)$ is $R(x, y) \rightarrow S(x, f_{\xi,2}(x, y), f_{\xi,3}(x, y))$. The subscripts in $f_{\xi,2}$ indicates that each tgd is assigned a unique Skolem function for each existentially quantified variable in the head. Skolemization is however not necessary, as long as the new nulls are generated by the distinct extension described in the chase step for tgds.

3. *The semi-oblivious chase* as presented by [8]. This is a slight variation of the oblivious chase. Let ξ be a tgd $\alpha(\bar{x}, \bar{y}) \rightarrow \beta(\bar{x}, \bar{z})$. Then triggers (ξ, h) and (ξ, g) are considered equivalent if $h(\bar{x}) = g(\bar{x})$. The semi-oblivious chase works as the oblivious one, except that exactly one trigger from each equivalence class is fired in a sequence. For this, we change line 2 of the standard chase algorithm to

   2 **if** exists a trigger (ξ, h) for I_i, and no trigger (ξ, g), where $g(\bar{x}) = h(\bar{x})$ has been fired before

   We note that the semi-oblivious chase by [8] also considers the Skolemized versions of the tgds. The Skolemization is slightly different from the one used in the oblivious chase, reflecting the slightly different choice of trigger to fire. For instance, the semi-oblivious Skolemized version of $\xi = R(x, y) \rightarrow \exists z, w\ S(x, z, w)$ is $R(x, y) \rightarrow S(x, f_{\xi,1}(x), f_{\xi,2}(x))$. As in the oblivious

[1] An infinite chase sequence $I_0, I_1, I_2, \ldots, I_n, \ldots$ results in the infinite instance $\bigcup_{n=0}^{\infty} I_n$.

chase, the Skolemization is however not necessary for the semi-oblivious chase, as long as the new nulls are generated by the distinct extension described in the chase step for tgds.

4. *The core chase* as presented by [6]. At each step, all currently active triggers are fired in parallel, and then the core of the union of the resulting instances is computed before the next step. Note that this makes the chase process deterministic and also fair. The core chase algorithm is described below.

Core-Chase$_\Sigma(I)$

1 $I_0 := I$; $i := 0$;
2 **if** exists an active egd trigger (ξ, h) for I_i
3 **then**
4 **if** $I_i \xrightarrow{(\xi,h)} \perp$
5 **then return** FAIL
6 **else** $I_i \xrightarrow{(\xi,h)} I_{i+1}$; $i := i + 1$ **goto** 2
7 **if** exists active tgd triggers for I_i
8 **then**
9 **for** all n active tgd triggers (ξ, h) for I_i
10 **do** compute in parallel $I_i \xrightarrow{(\xi,h)} J_j$, for $j = 1, \ldots, n$
11 $I_{i+1} := core(J_1 \cup \cdots \cup J_n)$; $i := i + 1$
12 **else**
13 **return** I_i
14 **goto** 2

To illustrate the difference between these chase variations, consider dependency set $\Sigma = \{\xi\}$, where ξ is the tgd $R(x, y) \rightarrow \exists z\, S(x, z)$ and instance I_0 below:

I_0
$R(a, b)$
$R(a, c)$
$S(a, d)$

There are two triggers for the set Σ on instance I_0, namely $(\xi, \{x/a, y/b\})$ and $(\xi, \{x/a, y/c\})$. Since $I_0 \models \xi$ neither of the triggers is active, so the standard chase will terminate at I_0. The core chase will also terminate at I_0. In contrast, both the oblivious and semi-oblivious chase will fire the first trigger, resulting in instance $I_1 = I_0 \cup \{S(a, z_1)\}$. The semi-oblivious chase will terminate at this point, while the oblivious chase will fire the second trigger, and then terminate in $I_2 = I_1 \cup \{S(a, z_2)\}$.

Let Σ be a set of tgds and egds, and I an instance, such that all chase sequences generated by Σ on I (use any of the 4 chase variations presented) terminate. The all leaves of the tree of chase sequences are homomorphically equivalent. Furthermore, if the chase algorithm is fair, then all leaves are homomorphically equivalent even if some sequences are infinite. [14] also showed that if the chase fails on one sequence, it will also fail for all fair sequences.

3. Complexity of the chase step

In this subsection we'll review the complexity of the chase step. We consider only the chase steps for tgds. The complexity of a $\star$ - chase step for egds is the same as the complexity of oblivious-chase tgd step, where $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$.

Algorithmically, there are two problems to consider. For knowing when to terminate the chase, we need to determine whether for a given instance I and tgd ξ there exists a homomorphism h such that (ξ, h) is a trigger on I. This pertains to the oblivious and semi-oblivious variations. The second problem relates to the standard and core chase: given an instance I and a tgd ξ, is there a homomorphism h, such that (ξ, h) is an *active* trigger on I. We call these problems the *trigger existence* problem, and the *active trigger existence* problem, respectively. The *data complexity* of these problems considers ξ fixed, and in the *combined complexity* both I and ξ are part of the input. The following theorem gives the combined and data complexities of the two problems.

Theorem 3.1. Let ξ be a tgd and I an instance. Then

1. For a fixed ξ, testing whether there exists a trigger or an active trigger on a given I is polynomial.
2. Testing whether there exists a trigger for a given ξ on a given I is NP-complete.
3. Testing whether there exists an active trigger for a given ξ and a given I is Σ_2^{p}-complete.

Proof:
The polynomial cases can be verified by checking all homomorphisms from the body of the dependency into the instance. For the active trigger problem we also need to consider, for each such homomorphism, if it has an extension that maps the head of the dependency into the instance. These tasks can be carried out in $O(|I|^{|vars(\alpha)|})$ and $O(|I|^{|vars(\alpha \cup \beta)|})$ time, respectively.

It is easy to see that the trigger existence problem is NP-complete in combined complexity, as the problem is equivalent to testing whether there exists a homomorphism between two instances (in our case α and I); a problem shown by [26] to be NP-complete.

For the combined complexity of the active trigger existence problem, we observe that it is in Σ_2^{P}, since one may guess a homomorphism h from α into I, and then use an NP oracle to verify that there is no extension h' of h, such that $h'(\beta) \subseteq I$. For the lower bound we will reduce the following problem to the active trigger existence problem. Let $\phi(\bar{x}, \bar{y})$ be a Boolean formula in 3CNF over the variables in $\bar{x}$ and $\bar{y}$. Is the formula

$$\exists \bar{x}\, \neg\big(\exists \bar{y}\, \phi(\bar{x}, \bar{y})\big)$$

true? This problem by [27] is a variation of the standard $\exists\forall$-QBF problem by [28].

For the reduction, let ϕ be given. We construct an instance I_ϕ and a tgd ξ_ϕ. The instance I_ϕ is as below:

F		
1	0	0
0	1	0
0	0	1
1	1	0
1	0	1
0	1	1
1	1	1

N	
0	1
1	0

The tgd $\xi_\phi = \alpha \rightarrow \beta$ is constructed as follows. For each variable $x \in \bar{x}$ in $\phi(\bar{x}, \bar{y})$, the body α will contain the atom $N(x, x')$ (x' is used to represent $\neg x$). The head β is existentially quantified over that set $\bigcup_{y \in \bar{y}} \{y, y'\}$ of variables. For each conjunct C of ϕ, we place in β an atom $F(x, y, z)$, where x, y and z are the variables in C, with the convention that if variable x is negated in C, then x' is used in the atom. Finally for each $y \in \bar{y}$, we place in β the atom $N(y, y')$, denoting that y and y' should not have the same truth assignment.

Suppose now that the formula $\exists \bar{x}\, \neg\big(\exists \bar{y}\, \phi(\bar{x}, \bar{y})\big)$ is true. This means that there is a $\{0, 1\}$-valuation h of $\bar{x}$ such that for any $\{0, 1\}$-valuation h' of $\bar{y}$, the formula $\phi(h(\bar{x}), h'(\bar{y}))$ is false. It is easy to see that $h(\alpha) \subseteq I$. Also, since $\phi(h(\bar{x}), h'(\bar{y}))$ is false for any valuation h', for each h' there must be an atom $F(x, y, z) \in \beta$, such that $h' \circ h(F(x, y, z))$ is false, that is, either $h' \circ h(F(x, y, z)) = F(0, 0, 0) \notin I_\phi$, or h' assigns for some existentially quantified variables non-Boolean values. Consequently the trigger (ξ, h) is active on I_ϕ.

For the other direction, suppose that there exists a trigger (ξ, h) which is active on I_ϕ, i.e., $h(\alpha) \subseteq I_\phi$ and $h'(\beta) \notin I$, for any extension h' of h. This means that for any such extension h', either h' is not $\{0, 1\}$-valuation, or that the atom $F(0, 0, 0)$ is in $h'(\beta)$. Thus the formula $\exists \bar{x}\, \neg\big(\exists \bar{y}\, \phi(\bar{x}, \bar{y})\big)$ is true. □

Note that the trigger existence relates to the oblivious and semi-oblivious chase variations, whereas the active trigger existence relates to the standard and the core chase. This means that the oblivious and semi-oblivious chase variations have the same complexity. This is not the case for the standard and the core chase. because the core chase step applies all active triggers in parallel and also involves the core computation for the resulted instance. [29] have shown that computing the core involves a DP-complete decision problem.

4. Chase termination questions

From the previous section we know that we may determine if the chase can continue at any given step by checking the existence of trigger/active trigger. On the other hand, testing if the chase process terminates, even when considering only tgds, as we will see, is undecidable. In the following subsection

we show that for any set Σ of tgds and egds there is a set Σ^{tgd} of tgds only, such that for any instance I and $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$, if the $\star$-chase for I and Σ^{tgd} terminates (for some chase sequences), then so does the $\star$-chase for I and Σ.

The different chase variations have dissimilar termination behaviors, so in the second subsection we introduce some notions that will help to distinguish them.

4.1. A sufficient condition for egds and tgds

In this subsection we present a rewriting technique that transforms a set of tgds and egds into a new set of tgds only, such that the chase termination on the new set will guarantee the chase termination on the initial set. Same rewritting for egds was used in [30] by Gottlob and Nash in order to prove that the core solution in data exchange can be computed in polynomial time. In our case the egd rewritting is used to find a sufficient condition for the chase termination in case the set of dependencies consists of both tgds and egds.

Let E be a new binary relational symbol outside the schema $\mathbf{R}$. Intuitively $E(x, y)$ will mean that elements x and y are equal. Given a set Σ of egds and tgds, by Σ^{tgd} we denote the set of tgds constructed as follows,

1. If $\xi \in \Sigma$ and ξ is a tgd, then add ξ to Σ^{tgd}.
2. For each egd $\alpha \rightarrow x = y$ from Σ, add the tgd $\alpha \rightarrow E(x, y), E(y, x)$ to Σ^{tgd}.
3. For each predicate symbol $R \in \mathbf{R}$ used in Σ and for each integer i such that $1 \leq i \leq arity(R)$, add the following tgd to Σ^{tgd}:

$$E(x, y), R(x_1, x_2, \ldots, , x_{i-1}, x, x_{i+1}, \ldots, x_{arity(R)}) \rightarrow \\ R(x_1, x_2, \ldots, , x_{i-1}, y, x_{i+1}, \ldots, x_{arity(R)})$$

Note that for any set Σ the transformation into Σ^{tgd} is polynomial in the size of Σ.

Example 4.1. As an example of the transformation, consider Σ containing the following two dependencies:

$$\begin{aligned} R(x, x) &\rightarrow \exists y, z\ S(x), R(y, z) \\ R(x, y) &\rightarrow x = y \end{aligned}$$

In this case Σ^{tgd} will contain the following set of tgds:

$$\begin{aligned} R(x, x) &\rightarrow \exists y, z\ S(x), R(y, z) \\ R(x, y) &\rightarrow E(x, y), E(y, x) \\ E(x, y), R(z, y) &\rightarrow R(z, x) \\ E(x, y), R(y, z) &\rightarrow R(x, z) \\ E(x, y), S(x) &\rightarrow S(y) \end{aligned}$$

The following theorem shows that the chase termination on the rewritten set Σ^{tgd} of tgds ensures chase termination for the initial set Σ of both tgds and egds.

Theorem 4.2. Let Σ be a set of tgds and egds, let I be an instance, and let $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$. If the $\star$-chase on I with Σ^{tgd} terminates (for some chase sequences), then so does the $\star$-chase on I with Σ.

Proof:
We will prove the theorem only for the standard-chase case, for the other variations the proof is similar. Let I be an instance, Σ a set of tgds and egds, and $I_0 = I, I_1, I_2, \ldots$ a fair successful standard chase sequence on I with Σ. We will show that there exists a fair standard chase sequence $J_0 = I, J_1, J_2, \ldots$ on I with Σ^{tgd}, such that for each I_k in the first sequence there exists a J_ℓ in the second sequence with $I_k \subseteq J_\ell$. We will construct the J-sequence inductively on the length of the I-sequence.

For the base step of the induction we have $J_0 = I = I_0$. For the inductive step, suppose that there exists a sequence $J_0, \ldots, J_\ell$, and for all I_i, where $i \leq k$, in the I-sequence, there exists an instance J_j in the J-sequence, such that $I_i \subseteq J_j$. We need to show that for $k+1$ there exists a positive integer ℓ' such that $J_0 = I, J_1, J_2, \ldots, J_\ell, \ldots, J_{\ell'}$ is (a prefix of) a fair chase sequence with Σ^{tgd}, and that $I_{k+1} \subseteq J_{\ell'}$.

From the definition of a chase sequence we have that $I_k \xrightarrow{(\xi,h)} I_{k+1}$, where $h(body(\xi)) \subseteq I_k$, and (ξ, h) is a trigger active on I_k. If ξ is a tgd, then $\xi \in \Sigma^{tgd}$, and since $h(body(\xi)) \subseteq I_k \subseteq J_\ell$, it follows that either $I_{k+1} \subseteq J_\ell$, or the trigger (ξ, h) is active on J_ℓ. Thus $J_\ell \xrightarrow{(\xi,h)} J_{\ell+1}$, and $I_{k+1} \subseteq J_{\ell+1}$. In other words, $\ell' = \ell + 1$.

Suppose then that ξ is an egd $\alpha \rightarrow x = y$. This means that Σ^{tgd} will contain the tgd $\alpha \rightarrow E(x,y), E(y,x)$, and for each relational symbol R and i, where $1 \leq i \leq arity(R)$, the set Σ^{tgd} will contain a tgd of the form (1). Now $I_k \xrightarrow{(\alpha \rightarrow x=y,\, h)} I_{k+1}$, and all occurrences of $h(y)$ in I_k have been replaced by $h(x)$ in I_{k+1} (assuming $h(x) < h(y)$ in the enumeration of variables). Clearly $(\alpha \rightarrow E(x,y), E(y,x), h)$, is a trigger for Σ^{tgd} on J_ℓ. Thus $J_\ell \xrightarrow{(\alpha \rightarrow E(x,y), E(y,x), h)} J_{\ell+1}$, and $J_{\ell+1} = J_\ell \cup \{E(h(x), h(y)), E(h(y), h(x))\}$. For each tuple $R(\ldots, h(y), \ldots)$ in $I_k \subseteq J_\ell$ there will be a tuple-generating dependency $\xi = E(x,y), R(\ldots, y, \ldots) \rightarrow R(\ldots, x, \ldots)$ in Σ^{tgd}. We then take the chase step $J_{\ell+1} \xrightarrow{(\xi,h)} J_{\ell+2}$ which adds tuple $R(\ldots, h(x), \ldots)$ to $J_{\ell+2}$ If there are m occurrences of $h(y)$ in J_ℓ, we repeat similar chase steps $m-1$ times, and arrive at instance $J_{\ell+m}$ which clearly satisfies $I_{k+1} \subseteq J_{\ell+m}$. This concludes the inductive proof and the claim of the theorem. □

Note that the converse of the previous theorem does not always holds. Consider for example the set Σ consiting of one tgd $R(x,y) \rightarrow \exists z R(y,z)$ and one egd $R(x,y) \rightarrow x = y$. It is easy to show that for input instance $I = \{R(a,a)\}$ any fair $*$-chase sequence will terminate but the chase will not terminate for I with Σ^{tgd}. Also worth mentioning that because Σ^{tgd} consists only on tgds for any input instance I the chase of I with Σ^{tgd} will never fail, whereas it may fail for Σ. For this just consider in the previous example instance $I = \{R(a,b)\}$.

With Theorem 4.2 we have showed that it suffices to check if the standard chase with Σ^{tgd} on I terminates, in order to infer that the standard chase with Σ (containing both tgd's and egds) on I will

terminate. Note also that in case Σ contains only egds and full tgds, then Σ^{tgd} will contain only full tgds ensuring that the chase terminates. The theorem can easily be extended to all instances as follows:

Corollary 4.3. Let Σ be a set if tgds and egds. If all $\star$-chase sequences with Σ^{tgd} terminates on all instances, then all $\star$-chase sequences with Σ will also terminate for all instances, where $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$.

The previous results gives us a sufficient way to test termination for sets of egds and tgds. Thus, one may use to test if Σ^{tgd} belongs to one of the known termination classes, as presented in the following subsection, in order to infer that Σ terminates. Thus, from now one will consider only classes of tgds.

4.2. Termination classes

Let $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$, corresponding to the chase variations introduced in Section 2, and let Σ be a set of tgds. If there exists a terminating $\star$-chase sequence with Σ on I, we say that the $\star$-chase terminates for *some* branch on instance I, and denote this as $\Sigma \in \mathsf{CT}^{\star}_{I\exists}$. Here $\mathsf{CT}^{\star}_{I\exists}$ is thus to be understood as the class of all sets of tgds for which the $\star$-chase terminates on some branch on instance I. Likewise, $\mathsf{CT}^{\star}_{I\forall}$ denotes the class of all sets of tgds for which the $\star$-chase with Σ on I terminates on *all* branches. From the definition of the chase variations it is easy to observe that any trigger applicable by the standard chase step on an instance I is also applicable by the semi-oblivious and oblivious chase steps on the same instance. Similarly, all the triggers applicable by the semi-oblivious chase step on an instance I are also applicable by the oblivious chase step on instance I. Thus $\mathsf{CT}^{\mathsf{obl}}_{I\exists} \subseteq \mathsf{CT}^{\mathsf{sobl}}_{I\exists} \subseteq \mathsf{CT}^{\mathsf{std}}_{I\exists}$. It is also easy to verify that $\mathsf{CT}^{\mathsf{std}}_{I\forall} \subseteq \mathsf{CT}^{\mathsf{std}}_{I\exists}$, and that $\mathsf{CT}^{\mathsf{sobl}}_{I\forall} \subseteq \mathsf{CT}^{\mathsf{std}}_{I\forall}$. We also observe that for the oblivious and semi-oblivious chase each sequence will fire (eventually, in case of infinite sequences) the same set of triggers. From this, and the fairness property, it directly follows that $\mathsf{CT}^{\star}_{I\exists} = \mathsf{CT}^{\star}_{I\forall}$ for $\star \in \{\mathsf{obl}, \mathsf{sobl}\}$.

The following proposition shows that these results can be strengthened to strict inclusions:

Proposition 4.4. For any instance I we have:

$$\mathsf{CT}^{\mathsf{obl}}_{I\exists} = \mathsf{CT}^{\mathsf{obl}}_{I\forall} \subset \mathsf{CT}^{\mathsf{sobl}}_{I\forall} = \mathsf{CT}^{\mathsf{sobl}}_{I\exists} \subset \mathsf{CT}^{\mathsf{std}}_{I\forall} \subset \mathsf{CT}^{\mathsf{std}}_{I\exists}.$$

Proof:
(Sketch) For the strict inclusion $\mathsf{CT}^{\mathsf{obl}}_{I\forall} \subset \mathsf{CT}^{\mathsf{sobl}}_{I\forall}$ consider

$$I = \{R(a,b)\} \text{ and } \Sigma = \{R(x,y) \rightarrow \exists z\, R(x,z)\}.$$

Then the oblivious chase sequence will look like

$$\{R(a,b)\} \xrightarrow{(\xi,\{x/a,y/b\})} \{R(a,b), R(a,z_1)\} \xrightarrow{(\xi,\{x/a,y/z_1\})} \{R(a,b), R(a,z_1), R(a,z_2)\}$$
$$\xrightarrow{(\xi,\{x/a,y/z_2\})} \ldots \{R(a,z_1), \ldots, R(a,z_n)\} \xrightarrow{(\xi,\{x/z_n,y/z_{n+1}\})} \ldots$$

which will converge only at the infinite instance

$$\bigcup_{n\geq 1}\{R(z_n)\}\cup\{R(a)\}.$$

The semi-oblivious chase, on the other hand, will terminate at the instance $\{R(a,b),R(a,z_1)\}$. We conclude that $\Sigma\in \mathsf{CT}^{\mathsf{sobl}}_{I\forall}$ but $\Sigma\notin \mathsf{CT}^{\mathsf{obl}}_{I\exists}$.

For the second strict inclusion $\mathsf{CT}^{\mathsf{sobl}}_{I\exists}\subset \mathsf{CT}^{\mathsf{std}}_{I\forall}$, consider

$$I=\{S(a,a)\} \text{ and } \Sigma=\{S(x,y)\rightarrow\exists z\, S(y,z)\}.$$

Because $I\models\Sigma$ it follows that $\Sigma\in \mathsf{CT}^{\mathsf{std}}_{I\forall}$. On the other hand, the semi-oblivious chase will converge only at the infinite instance

$$\bigcup_{n\geq 1}\{S(z_n,z_{n+1})\}\cup\{S(a,a),S(a,z_1)\},$$

and thus $\Sigma\notin \mathsf{CT}^{\mathsf{sobl}}_{I\exists}$.

For the final strict inclusion $\mathsf{CT}^{\mathsf{std}}_{I\forall}\subset \mathsf{CT}^{\mathsf{std}}_{I\exists}$, let

$$I=\{S(a,b),R(a)\} \text{ and } \Sigma=\{S(x,y)\rightarrow\exists z\, S(y,z);\;\; R(x)\rightarrow S(x,x)\}.$$

It is easy to see that any standard chase sequence that starts by firing the trigger based on the first tgd will not terminate as it will generate new tuple $S(b,z_1)$ and this will fire an infinite chase sequence on any branch. On the other hand, if we first fire the trigger based on the second tgd the standard chase will terminate after one step. □

The next questions are whether all or some $\star$-chase sequences terminate on *all* instances. The corresponding classes of sets of tgds are denoted $\mathsf{CT}^{\star}_{\forall\forall}$ and $\mathsf{CT}^{\star}_{\forall\exists}$, respectively. Obviously $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}\subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$. Similarly to the instance dependent termination classes, $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}\subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$ and $\mathsf{CT}^{\star}_{\forall\forall}=\mathsf{CT}^{\star}_{\forall\exists}$, for $\star\in\{\mathsf{obl},\mathsf{sobl}\}$. We can relate the oblivious, semi-oblivious and standard chase termination classes as follows:

Theorem 4.5. $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}=\mathsf{CT}^{\mathsf{obl}}_{\forall\exists}\subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}=\mathsf{CT}^{\mathsf{sobl}}_{\forall\exists}\subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}\subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$.

Proof:
We will only show the strict inclusion parts of the theorem. For the first inclusion, let $\mathbf{R}=\{R\}$, and $\Sigma=\{\xi\}$, where

$$\xi=\{R(x,y)\rightarrow\exists z\, R(x,z)\}.$$

Let I be an arbitrary non-empty instance. Then I contains at least one tuple, say, $R(a,b)$, and we can write I as

$$\{R(a,b)\}\cup J,$$

for some (possible empty) instance J. The oblivious chase will generate the sequence

$$\{R(a,b)\}\cup J\xrightarrow{(\xi,\{x/a,y/b\})}\{R(a,b),R(a,z_1)\}\cup J\xrightarrow{(\xi,\{x/a,y/z_1\})}\dots$$

and thus converge in (possibly a superset of) the infinite instance

$$\bigcup_{n\geq 1}\{R(z_n, z_{n+1}\} \cup \{R(a,b), R(a,z_1)\} \cup J.$$

For the semi-oblivious chase we observe that the list of triggers $(\xi, \{x/a, y/b\})$, $(\xi, \{x/a, y/z_1\})$, $(\xi, \{x/a, y/z_2\}), \ldots$ are equivalent, since they all map the only common variable x, from the by and the head of the tgd, to a. Therefore the semi-oblivious chase will terminate with the following instance: $\{R(a,b), R(a,z_1)\} \cup J$. We conclude that $\Sigma \in \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall} \setminus \mathsf{CT}^{\mathsf{obl}}_{\forall\exists}$.

The second strict inclusion $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ is more intricate, as most sets of dependencies in $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ are also in $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$. To distinguish the two classes, let $\Sigma = \{\xi_1, \xi_2\}$, where

$$\begin{aligned} \xi_1 &= R(x) \rightarrow \exists z\, S(z), T(z,x), \text{ and} \\ \xi_2 &= S(x) \rightarrow \exists z'\, R(z'), T(x,z'). \end{aligned}$$

Let I be an arbitrary non-empty instance, and suppose that

$$I = \{R(a_1), \ldots, R(a_n), S(b_1), \ldots, S(b_m)\}.$$

There is no loss of generality, since if the standard chase with Σ on I terminates, then it will also terminate even if the initial instance contains atoms over the relational symbol T. It is easy to see that all standard chase sequences with Σ on I will terminate in the instance $A \cup B$, where

$$\begin{aligned} A &= \bigcup_{i=1}^{n}\{R(a_i), S(z_i), T(z_i, a_i)\}, \text{ and} \\ B &= \bigcup_{i=1}^{m}\{R(z'_i), S(b_i), T(b_i, z'_i)\}. \end{aligned}$$

On the other hand, all semi-oblivious chase sequences will converge only at the infinite instance

$$\bigcup_{k\geq 1}\big(C_k \cup D_k\big) \cup A \cup B,$$

where

$$\begin{aligned} C_k = &\bigcup_{i=1}^{m}\{S(z_{kn+(k-1)m+i}), T(z_{kn+(k-1)m+i}, z'_{(k-1)m+(k-1)n+i})\} \cup \\ &\bigcup_{j=1}^{n}\{S(z_{kn+km+j}), T(z_{kn+km+j}, z'_{km+(k-1)n+j})\}, \\ D_k = &\bigcup_{i=1}^{n}\{R(z'_{km+(k-1)n+i}), T(z_{(k-1)n+(k-1)m+i}, z'_{km+(k-1)n+i})\} \cup \\ &\bigcup_{j=1}^{m}\{R(z'_{km+kn+j}), T(z_{kn+(k-1)m+j}, z'_{km+kn+j})\}. \end{aligned}$$

For the last inclusion $\mathsf{CT}^{\mathsf{std}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$, consider set of tgds $\Sigma = \{R(x,y) \rightarrow R(y,y);\ \ R(x,y) \rightarrow \exists z\, R(y,z)\}$. Clearly $\Sigma \in \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$, because for any instance, all chase sequences that start by firing the first tgd will terminate. On the other hand $\Sigma \notin \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$, as the standard chase with Σ does not terminate on $I = \{R(a,b)\}$ whenever the second tgd is applied first. □

Note that for any $\star \in \{\mathsf{std}, \mathsf{obl}, \mathsf{sobl}, \mathsf{core}\}$, and for any non-empty instance I, we have that $\mathsf{CT}^{\star}_{\forall\forall} \subset \mathsf{CT}^{\star}_{I\forall}$ and $\mathsf{CT}^{\star}_{\forall\exists} \subset \mathsf{CT}^{\star}_{I\exists}$.

We now turn our attention to the core chase. Note the core chase is deterministic since all active triggers are fired in parallel, before taking the core of the result. Thus we have:

Proposition 4.6. $\mathsf{CT}^{\mathsf{core}}_{I\forall} = \mathsf{CT}^{\mathsf{core}}_{I\exists}$ and $\mathsf{CT}^{\mathsf{core}}_{\forall\forall} = \mathsf{CT}^{\mathsf{core}}_{\forall\exists}$.

It is well known that all here considered chase variations compute a finite *universal model* of I and Σ when they terminate, as shown by [14, 6, 7, 8]. In particular, [6], showed that if $I \cup \Sigma$ has a finite universal model, the core chase will terminate in an instance that is the core of all universal models (which are homomorphically equivalent). The standard chase, on the other hand, might have to choose the correct sequence in order to terminate. The example above, with $I = \{S(a,b), R(a)\}$ and $\Sigma = \{S(x,y) \rightarrow \exists z S(y,z);\ R(x) \rightarrow S(x,x)\}$, serves as an illustration of the importance of "the correct sequence." We have:

Proposition 4.7. 1. $\mathsf{CT}^{\mathsf{std}}_{I\exists} \subset \mathsf{CT}^{\mathsf{core}}_{I\forall}$, for any instance I.

2. $\mathsf{CT}^{\mathsf{std}}_{\forall\exists} \subset \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$.

Proof:
(Sketch) To see that the inclusion in part (2) of the proposition is strict, let

$$I = \{R(a)\} \text{ and } \Sigma = \{R(x) \rightarrow \exists z\, R(z), S(x)\}.$$

In this setting there will be exactly one active trigger at each step, and the algorithm will converge only at the infinite instance

$$\bigcup_{i\geq 2} \{R(z_i), S(z_{i-1})\} \cup \{R(a), S(a), R(z_1)\}.$$

From this, it follows that $\Sigma \notin \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ and $\Sigma \notin \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$. Note that for any $i > 0$, the core of I_i is $\{R(a), S(a)\}$. Thus the core chase will terminate at instance $I_1 = \{R(a), S(a)\}$. □

The following Corollary highlights the relationship between the termination classes.

Corollary 4.8. $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists} \subset \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$.

5. Undecidability of termination

Theorem 5.1.

1. $\mathsf{CT}^{\mathsf{std}}_{I\forall}$ and $\mathsf{CT}^{\mathsf{std}}_{I\exists}$ are RE-complete *[6]*.

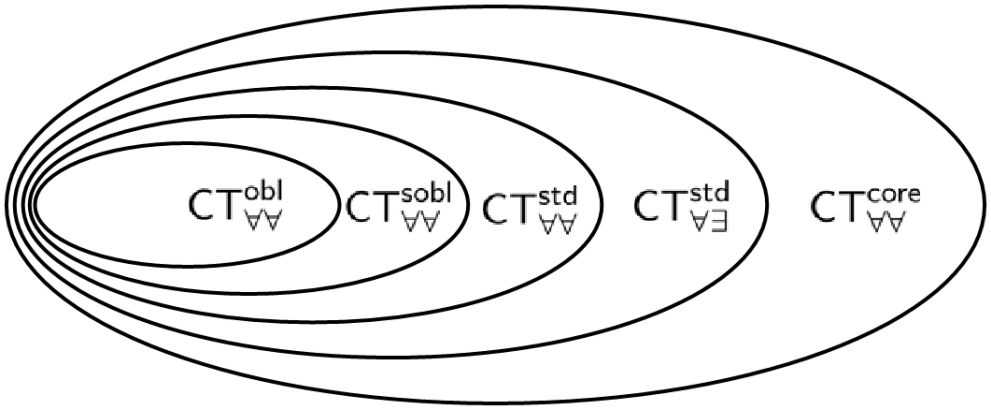


Figure 1. Termination classes for chase variations.

2. $\mathsf{CT}^{\mathsf{core}}_{I\forall} = \mathsf{CT}^{\mathsf{core}}_{I\exists}$, and both sets are RE-complete *[6]*.

3. $\mathsf{CT}^{\mathsf{sobl}}_{I\forall} = \mathsf{CT}^{\mathsf{sobl}}_{I\exists}$, and both sets are RE-complete *[8]*.

4. $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall} = \mathsf{CT}^{\mathsf{sobl}}_{\forall\exists}$, and both sets are RE-complete *[13]*.

5. $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall} = \mathsf{CT}^{\mathsf{obl}}_{\forall\exists}$, and both sets are RE-complete by *[13]*.

6. Let Σ be a set of guarded[2] tgds as introduced by *[7]*. Then the question $\mathsf{CT}^{\mathsf{core}}_{I\forall}$ is decidable *[31]*.

The proof in [6] of part 1 of Theorem 5.1 encodes a Turing machine M with an initial blank tape as a set Σ_M of tgds, such that a chase sequence using Σ_M mimics the computation history of M. Then [6] show that M halts if and only if the standard chase with Σ_M on the empty instance terminates. The dependencies in Σ_M are constructed so that some chase sequence terminates if and only if all chase sequences terminate. [6] also show that if the standard chase terminates, then so does the core chase, thus yielding them part 2 of Theorem 5.1.

However, the encoding used by [6], using the empty tape as input to M, does not allow for input instances to the chase other than the empty one. Thus their proof cannot be used to determine whether the chase terminates on *all* input instances, i.e. to determine the complexity of sets $\mathsf{CT}^{\star}_{\forall\forall}$ and $\mathsf{CT}^{\star}_{\forall\exists}$, for $\star \in \{\mathsf{std}, \mathsf{core}\}$.

The proof of part 3 of Theorem 5.1 by [8] first shows with a proof similar to [6] that the termination of the semi-oblivious chase on the empty instance is an RE-complete problem. [8] then shows that if the semi-oblivious chase with a set Σ of tgds terminates on a special "critical instance," it terminates on all instances, thus obtaining the RE-completeness result for $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.

[2] A tgd is guarded if its body contains an atom called guard that covers all variables occurring in the body.

[13] use a reduction from the halting problem for three-counter automata for showing that $\mathsf{CT}^{\mathsf{obl}}_{\forall\exists}$ is RE-complete. They then apply the "critical instance" technique to obtain the RE-completeness result for $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$.

This leaves the complexity of the sets $\mathsf{CT}^{\star}_{\forall\forall}$ and $\mathsf{CT}^{\star}_{\forall\exists}$ open, for $\star \in \{\mathsf{std}, \mathsf{core}\}$. We show that all of these sets are Π^0_2-complete in the Arithmetical Hierarchy. Our proofs use reductions from word rewriting systems, which we feel are symbolically closer to the chase than computation histories of Turing machines. Rather interestingly, as a corollary of our results, it follows that the "critical instance" technique is not applicable to the core and standard chases.

Parts 1 and 2 of Theorem 5.1 are also directly obtainable from our reduction. We note that our reduction is technically more involved than the one used by [6], where the empty database instance was used in the reduction. This is because here we also need to show termination on all input instances, not just on a given input instance.

Word rewriting systems. Let Δ be a finite set of symbols, denoted $a, b, \ldots$, possibly subscripted. A *word* over Δ is a finite sequence $a_1a_2\ldots a_n$, where each a_i is a symbol from Δ. The *empty* word ($n = 0$) is denoted ϵ. The *length* of a word $w = a_1a_2\ldots a_n$, denoted $|w|$, is n. The *concatenation* of two words $u = a_1a_2\ldots a_n$ and $v = b_1b_2\ldots b_m$, is the word $a_1a_2\ldots a_nb_1b_2\ldots b_m$, denoted uv. We have $|uv| = n + m$, and $\epsilon u = u\epsilon = u$, for all words u. We say that a word u is a *factor* of a word v, if $v = xuy$, for some words x and y. If $x = \epsilon$ we say that u is a *prefix* of v, and if $y = \epsilon$ we say that u is a *suffix* of v. Note that both x and y might be the empty word ϵ, so the factor relation is reflexive. We let Δ^* denote the set of all finite words over Δ.

A pair $\mathcal{R} = (\Delta^*, \Theta)$, where Θ is a finite subset of $\Delta^* \times \Delta^*$, is called a *word rewriting system*. Treating each pair in Θ as a *rule*, the relation Θ gives rise to a *rewriting relation* $\rightarrow_{\mathcal{R}} \subseteq \Delta^* \times \Delta^*$ defined as

$$\{(u, v) : u = x\ell y, v = xry, (\ell, r) \in \Theta, x, y \in \Delta^*\}.$$

We use the notation $u \rightarrow_{\mathcal{R}} v$ instead of $\rightarrow_{\mathcal{R}} (u, v)$. If $\mathcal{R}$ is understood from the context we will simply write $u \rightarrow v$. If we want to emphasize which rule $\rho \in \Theta$ was used we write $u \rightarrow_\rho v$. When $u \rightarrow_\rho v$ we say that v is obtained from u by a *rewriting step (based on ρ)*.

A sequence $w_0, w_1, w_2, \ldots$ of words from Δ^* is said to be a *$\mathcal{R}$-derivation sequence* (or simply a derivation sequence), if $w_i \rightarrow_{\mathcal{R}} w_{i+1}$ for all $i = 0, 1, 2, \ldots$. A derivation sequence might be finite or infinite. A derivation sequence $w_0, w_1, w_2, \ldots$ will sometimes also be written $w_0 \rightarrow_{\mathcal{R}} w_1 \rightarrow_{\mathcal{R}} w_2 \rightarrow_{\mathcal{R}} \cdots$, or simply $w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots$, when $\mathcal{R}$ is understood from the context. A word $w \in \Delta^*$ is said to be in *normal form* if there is no word $v \in \Delta^*$, $v \neq w$, such that $w \rightarrow_{\mathcal{R}} v$. Note that normal forms are not necessarily unique.

Definition 5.2. Let $\mathcal{R} = (\Delta^*, \Theta)$ be a word rewriting systems.

1. The *termination problem for $\mathcal{R}$ and a word $w_0 \in \Delta^*$*, is to determine whether all derivation sequences $w_0 \rightarrow_{\mathcal{R}} w_1 \rightarrow_{\mathcal{R}} w_2 \rightarrow_{\mathcal{R}} \cdots$ originating from w_0 are finite.

2. The *uniform termination problem for $\mathcal{R}$* is to determine whether for *all* words $w_0 \in \Delta^*$, it holds that all derivation sequences $w_0 \rightarrow_{\mathcal{R}} w_1 \rightarrow_{\mathcal{R}} w_2 \rightarrow_{\mathcal{R}} \cdots$ originating from w_0 are finite.

3. The *uniform normal form existence problem for* $\mathcal{R}$ is to determine whether all words in Δ^* have a (not necessarily unique) normal form, i.e. if all words have some terminating derivation sequence.

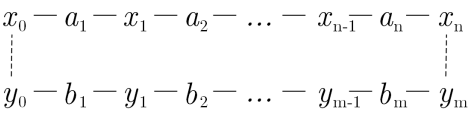
It is known that the termination problem is RE-complete, and that the uniform termination problem as well as the normal form existence problem are complete for level Π_2^0 in the arithmetic hierarchy [32].

The reduction. Let $\mathcal{R} = (\Delta^*, \Theta)$ we a word rewriting system. We now describe our reduction $\mathcal{R} \mapsto \Sigma_{\mathcal{R}}$. We assume without loss of generality that $\Delta = \{0, 1\}$. The tgd set $\Sigma_{\mathcal{R}}$ is over schema $\mathbf{S}_{\mathcal{R}} = (E, L, R)$, and is defined as $\Sigma_{\mathcal{R}} = \Sigma_\Theta \cup \Sigma_{LR}$.

We let $\Sigma_\Theta = \{\xi_\rho : \rho \in \Theta\}$, where ξ_ρ is

$$E(x_0, a_1, x_1), E(x_1, a_2, x_2), \ldots, E(x_{n-1}, a_n, x_n) \rightarrow$$
$$\exists\, y_0 \ldots \exists\, y_m\; L(x_0, y_0), E(y_0, b_1, y_1), E(y_1, b_2, y_2), \ldots, E(y_{m-1}, b_m, y_m), R(x_n, y_m),$$

when $\rho = (a_1 \ldots a_n, b_1 \ldots b_m)$. Intuitively, we model a word $w = a_1 a_2 \ldots a_n$ as the line-graph $E(x_0, a_1, x_1), E(x_2, a_2, x_3), \ldots, E(x_{n-1}, a_n, x_n)$. The effect of tgd $\xi_{(a_1 a_2 \ldots a_n, b_1 b_2 \ldots b_m)}$ is to transform the line graph for $a_1 a_2 \ldots a_n$ to a grid:

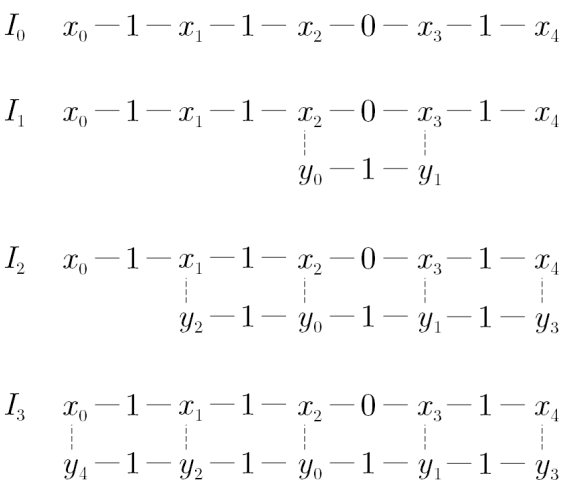


Since however a rewriting rule $\xi_{(a_1 a_2 \ldots a_n, b_1 b_2 \ldots b_m)}$ is applicable to longer words, such as for example for word $c_1 \ldots c_k a_1 a_2 \ldots a_n d_1 \ldots d_p$, resulting in word $c_1 \ldots c_k b_1 b_2 \ldots b_m d_1 \ldots d_p$, we need rules that copy the $c_1 \ldots c_k$ and $d_1 \ldots d_p$ parts to the new line of the grid. This is achieved by the following "grid creation" rules, using "left" L and "right" R predicates.

$$\begin{aligned}
\xi_{L_0} &= E(x_0, 0, x_1), L(x_1, y_1) \rightarrow \exists\, y_0\; L(x_0, y_0), E(y_0, 0, y_1)\\
\xi_{L_1} &= E(x_0, 1, x_1), L(x_1, y_1) \rightarrow \exists\, y_0\; L(x_0, y_0), E(x_0, 1, y_1)\\
\xi_{R_0} &= R(x_0, z_0), E(x_0, 0, x_1) \rightarrow \exists\, z_1\; E(z_0, 0, z_1), R(x_1, z_1)\\
\xi_{R_1} &= R(x_0, z_0), E(x_0, 1, x_1) \rightarrow \exists\, z_1\; E(z_0, 1, z_1), R(x_1, z_1)
\end{aligned}$$

We now define $\Sigma_{LR} = \{\xi_{L_0}, \xi_{L_1}, \xi_{R_0}, \xi_{R_1}\}$.

As an illustration, consider the rewrite rule $\rho = (0, 1) \in \Theta$ applied to word $w = 1101$. This gives us a rewrite step $1101 \rightarrow_\rho 1111$. However, in the corresponding core chase sequence we will need three chase-steps as illustrated below.



We still need a few more notions. A *path* of an instance I over schema $\mathbf{S}_\mathcal{R}$ is a set

$$\pi = \{E(x_0, a_1, x_1), E(x_1, a_2, x_2), \ldots, E(x_{n-1}, a_n, x_n)\}$$

of atoms of I, such that $\{a_1, a_2, \ldots, a_n\} \subseteq \Delta$ (recall that Δ is the alphabet of the rewriting system Θ), and the x_i's are pairwise distinct nulls from Nulls. The *word spelled by the path* π is

$$word(\pi) = a_1 a_2 \ldots a_n.$$

A *max-path* π in an instance I is a path, such that no other path in I is a strict superset of π. We can now relate words and instances as follows: Let I be an instance. We define

$$\begin{aligned} paths(I) &= \{\pi : \pi \text{ is a max-path in } I\} \text{ and,} \\ words(I) &= \{word(\pi) : \pi \in paths(I)\}. \end{aligned}$$

Clearly $paths(I)$ is finite, for any finite instance I. Conversely, let $w = a_1 a_2 \ldots a_n \in \Delta^*$. We define

$$I_w = \{E(x_0, a_1, x_1), E(x_1, a_2, x_2), \ldots, E(x_{n-1}, a_n, x_n)\},$$

where the x_i's are pairwise distinct nulls from Nulls. Clearly

$$\begin{aligned} words(I_w) &= \{w\}, \text{ and} \\ I_{word(\pi)} &\cong \pi. \end{aligned}$$

In order to relate core chase sequences to word derivation sequences, we shall look the core chase sequence in a finer granularity. Let $I_0, I_1, I_2, \ldots$ be a core chase sequence, where $I_0 = I_w$ for some $w \in \Delta^*$. Since the core chase step fires all dependencies in $\Sigma_\mathcal{R}$ in parallel, we will in the sequel write the sequence $I_0, I_1, I_2, \ldots$ as $I_0 \to_{\Sigma_\mathcal{R}} I_1 \to_{\Sigma_\mathcal{R}} I_2 \to_{\Sigma_\mathcal{R}} \cdots$. Consider a particular step $I_n \to_{\Sigma_\mathcal{R}} I_{n+1}$. Suppose wlog the tgds that fired on I_n were $\xi_1, \xi_2, \ldots, \xi_{k_n}$. We can think of the core chase step $I_n \to_{\Sigma_\mathcal{R}} I_{n+1}$ as k_n standard chase steps $I_n \to_{\xi_m} I_{nm}$, where $m = 1, 2, \ldots, k_n$, performed in parallel, followed by the computation of $I_{n+1} = core(I_{n1} \cup I_{n2} \cup \cdots \cup I_{nk_n})$. This allows us to prove the following lemma.

Lemma 5.3. Let $w \in \Delta^*$, and $I_w = I_0 \rightarrow_{\Sigma_\mathcal{R}} I_1 \rightarrow_{\Sigma_\mathcal{R}} I_2 \rightarrow_{\Sigma_\mathcal{R}} \cdots$ be a core chase sequence. Then $paths(I_{n1} \cup I_{n2} \cup \cdots \cup I_{nk_n}) = paths(I_{n+1})$, for all pairs of instances I_n, I_{n+1} in the sequence.

Proof:
Denote the instance $I_{n1} \cup I_{n2} \cup \cdots \cup I_{nk_n}$ with J. If there were a path π, such that $\pi \in paths(J) \setminus paths(core(J))$, there would have to be atoms of the form $L(x,x)$ and $R(x,x)$ in J. But such L and R atoms do not occur in I_w, neither are they created by any of the rules $\xi_\rho, \xi_{L_1}, \xi_{L_2}, \xi_{R_1}$ or ξ_{R_2}. □

We are now ready for the following result.

Theorem 5.4. For each derivation sequence

$$w_0 \rightarrow_\mathcal{R} w_1 \rightarrow_\mathcal{R} \cdots \rightarrow_\mathcal{R} w_n$$

there is a sequence $1 \leq j_1 < j_2 < \cdots < j_n$ of indices, such that

$$I_{w_0} = I_0 \rightarrow_{\Sigma_\mathcal{R}} \cdots \rightarrow_{\Sigma_\mathcal{R}} I_{j_1} \rightarrow_{\Sigma_\mathcal{R}} \cdots \rightarrow_{\Sigma_\mathcal{R}} I_{j_2} \rightarrow_{\Sigma_\mathcal{R}} \cdots \rightarrow_{\Sigma_\mathcal{R}} I_{j_n}$$

is a core chase sequence, and there is a path $\pi_n \in paths(I_{j_n}) \setminus paths(I_{j_{n-1}})$, with $word(\pi_n) = w_n$.

Proof:
We prove the claim by an induction on n.

For the base case, we note that if $w_0 \rightarrow_\mathcal{R} w_1$, then there must be a rule $(\ell, r) \in \Theta$, such that $w_0 = x\ell y$ and $w_1 = xry$. Then the dependency $\xi_{(\ell,r)} \in \Sigma_\Theta$ will fire on I_{w_0} (perhaps along with other tgds from $\Sigma_\mathcal{R}$), resulting in instance I_1, such that, by Lemma (5.3), there is a path π_1 in $paths(I_1) \setminus paths(I_0)$ with $word(\pi_1) = r$. If $|x| = |y| = 0$, then $w_1 = r$. Otherwise, let $m = max\{|x|, |y|\}$. The "copy" dependencies in Σ_{LR} will fire at steps $2, \ldots, 1+m$, and from Lemma (5.3) it follows there will be a path π_1 in $paths(I_{1+m}) \setminus paths(I_1)$, such that $word(\pi_1) = xry = w_1$. Hence $j_1 = 1 + m$.

As inductive hypothesis, suppose that there is a path π_n in $paths(I_{j_n}) \setminus paths(I_{j_{n-1}})$, such that $word(\pi_n) = w_n$. Suppose that $w_n \rightarrow_\mathcal{R} w_{n+1}$, for a word w_{n+1}, where $w_n = x\ell y$, $w_{n+1} = xry$, and $(l, r) \in \Theta$. Let $m = max\{|x|, |y|\}$. Similarly to the base case there will be a path π_{n+1} in $I_{n+m} \setminus I_n$, such that $word(\pi_{n+1}) = w_{n+1}$. Hence $j_{n+1} = n + m$. □

We need a couple of more notions. Let w be a word in Δ^*. The *level n derivation tree of w*, is a Δ^*-labeled tree $\mathcal{T}_{w,n}$, such that $\mathcal{T}_{w,0}$ is a single node labeled w, and the tree $\mathcal{T}_{w,n+1}$ is obtained by adding a node labeled v as a child of a leaf node in $T_{w,n}$ labeled u, whenever $u \rightarrow_\rho v$, for some $\rho \in \Theta$. By $\mathcal{T}_w$, the *derivation tree of w*, we mean the tree $\bigcup_{n \geq 0} \mathcal{T}_{w,n}$.

Theorem 5.5. For each $w_0 \in \Delta^*$ and $n \geq 0$, such that $I_{w_0} = I_0 \rightarrow_{\Sigma_\Theta} I_1 \rightarrow_{\Sigma_\Theta} \cdots \rightarrow_{\Sigma_\Theta} I_n$ is a core chase sequence, there is a injection μ_n from $paths(I_n)$ to $\mathcal{T}_{w_0,n}$, such that the label of $\mu_n(\pi)$ contains $word(\pi)$ as a factor, for each $\pi \in paths(I_n)$.

Proof:
We do an induction on n. For the base case we note that $paths(I_{w_0}) = \{\pi\}$, where $word(\pi) = w_0$ is the label of the root $\mathcal{T}_{w_0,0}$.

For the inductive step, consider the core chase step $I_n \rightarrow_{\Sigma_\mathcal{R}} I_{n+1}$. For each path $\pi \in paths(I_{n+1}) \cap paths(I_n)$, we assign $\mu_{n+1}(\pi) = \mu_n(\pi)$. Each path in $\pi \in paths(I_{n+1}) \setminus paths(I_n)$ must have been created by a dependency. For each $\xi \in \Sigma_\mathcal{R}$ that fired at step $n+1$ there are two possibilities.

1. $\xi \in \Sigma_\Theta$. Then there is a rewrite rule $(\ell, r) \in \Theta$, and a path $\pi_n \in paths(I_n)$, such that $word(\pi_n) = x\ell y$, for some $x, y \in \Delta^*$. Furthermore, there is a path $\pi_{n+1} \in paths(I_{n+1}) \setminus paths(I_n)$, with $word(\pi_{n+1}) = r$. By the inductive hypothesis, the label of the node $\mu_n(\pi_n)$ in $\mathcal{T}_{w_0,n}$ contains $x\ell y$ as a factor, i.e. the label of $\mu_n(\pi_n)$ is $x'\ell y'$, where x is a suffix of x' and y is a prefix of y'. Now $\mathcal{T}_{w_0,n+1}$ will have a child of $\mu_n(\pi_n)$ labeled $x'ry'$. We assign $\mu_{n+1}(\pi_{n+1})$ to be that child. Since r is a factor of $x'ry'$, the claim follows.

2. $\xi \in \Sigma_{LR}$. Then there must be a path $\pi_n \in paths(I_n)$, (and possibly another dependency $\xi' \in \Sigma_{LR}$), such that firing ξ (together with ξ'), creates a path $\pi_{n+1} \in paths(I_{n+1}) \setminus paths(I_n)$, such that $\pi_n \subset \pi_{n+1}$. By the inductive hypothesis, the label of the node $\mu_n(\pi_n)$ in $\mathcal{T}_{w_0,n}$ contains $word(\pi_n)$ as a factor. From the construction of Σ_{LR} and μ_n it follows that $word(\pi_{n+1})$ also is a factor of the label of $\mu_n(\pi_n)$. Since $\pi_n \subset \pi_{n+1}$, it means that $\pi_n \notin paths(I_{n+1})$, as π_n no longer is a max-path in I_{n+1}. Therefore we can assign $\mu_{n+1}(\pi_{n+1}) = \mu_n(\pi_n)$. As $\mu_n(\pi_n)$ also is a node in $\mathcal{T}_{w_0,n+1}$ the claim follows. □

Theorem 5.6. Let $\mathcal{R} = (\Delta^*, \Theta)$ be a word rewriting system, and let $w_0 \in \Delta^*$. Then the core chase sequence $I_{w_0} = I_0 \rightarrow_{\Sigma_\mathcal{R}} I_1 \rightarrow_{\Sigma_\mathcal{R}} I_2 \rightarrow_{\Sigma_\mathcal{R}} \cdots$ is infinite if and only if there is an infinite derivation $w_0 \rightarrow_\mathcal{R} w_1 \rightarrow_\mathcal{R} w_2 \rightarrow_\mathcal{R} \cdots$.

Proof:
Suppose there is an infinite derivation sequence $w_0 \rightarrow_\mathcal{R} w_1 \rightarrow_\mathcal{R} w_2 \rightarrow_\mathcal{R} \cdots$. It then follows from Theorem 5.4 that the core chase sequence $I_{w_0} = I_0 \rightarrow_{\Sigma_\mathcal{R}} I_1 \rightarrow_{\Sigma_\mathcal{R}} I_2 \rightarrow_{\Sigma_\mathcal{R}} \cdots$ is infinite as well.

Conversely, suppose that the core chase sequence $I_{w_0} = I_0 \rightarrow_{\Sigma_\mathcal{R}} I_1 \rightarrow_{\Sigma_\mathcal{R}} I_2 \rightarrow_{\Sigma_\mathcal{R}} \cdots$ is infinite. At each core chase step n, if at least one dependency from Σ_Θ fires, we have $|paths(I_n)| > |paths(I_{n-1})|$, and if only dependencies from Σ_{LR} fire, we have $|paths(I_n)| = |paths(I_{n-1})|$. It is however easy to see that there can only be a finite number of consecutive chase steps that only fire dependencies from Σ_{LR}. Consequently $\bigcup_{n\geq 0} I_n$ contains an infinite number of paths. From Theorem (5.5) it then follows that $\mathcal{T}_{w_0}$ is infinite as well, and since $\mathcal{T}_{w_0}$ is finitely branching, König's Lemma tells us that $\mathcal{T}_{w_0}$ has an infinite branch. Let the labels on this path be, in order, $w_0, w_1, w_2, \ldots$. This means that the derivation $w_0 \rightarrow_\mathcal{R} w_1 \rightarrow_\mathcal{R} w_2 \rightarrow_\mathcal{R} \ldots$ is infinite. □

This theorem, together with the RE-completeness of rewriting termination, yields the undecidability result of [6] for core chase termination on a given instance.

Corollary 5.7. The set $\mathsf{CT}^{\mathsf{core}}_{I\forall}$ is RE-complete.

The uniform case. Next we shall relate the uniform termination problem with the set $\mathsf{CT}^{\mathsf{core}}_{\forall\forall}$. This means that we need to consider arbitrary instances, not just instances of the form I_w, for $w \in \Delta^*$. If the arbitrary instance I is cyclic the behavioral correspondence between word derivations in $\mathcal{R}$ and the core chase sequence with $\Sigma_{\mathcal{R}}$ on I breaks down. We therefore need to extend the schema $\mathbf{S}_{\mathcal{R}}$ to include relational symbols D and E^*. Intuitively, the unary relation D will hold the active domain of an instance, and the binary relation E^* will hold the transitive closure of the graph obtained from the instance.

The following tgd set Σ_{AD} computes $dom(I) \cup \Delta$ in relation D.

$$\begin{aligned} & \rightarrow && D(0), D(1) \\ E(x,z,y) & \rightarrow && D(x), D(z), D(y) \\ L(x,y) & \rightarrow && D(x), D(y) \\ R(x,y) & \rightarrow && D(x), D(y) \\ E^*(x,y) & \rightarrow && D(x), D(y) \end{aligned}$$

Given an instance I, by *the graph of* I we mean the graph with edge set:

$$G_I = \{(x,y) \,:\, E(x,z,y) \in I \text{ or } E^*(x,y) \in I \text{ or } L(x,y) \in I \text{ or } R(x,y) \in I\}.$$

The following set Σ_{TC} computes in E^* the transitive closure of G_I:

$$\begin{aligned} E(x,z,y) & \rightarrow && E^*(x,y) \\ L(x,y) & \rightarrow && E^*(x,y) \\ R(x,y) & \rightarrow && E^*(x,y) \\ E^*(x,y), E^*(y,z) & \rightarrow && E^*(x,z) \end{aligned}$$

If G_I has a cycle, the chase will eventually place an atom of the form $E^*(v,v)$ in the instance. Once an instance contains such a tuple, the dependencies in the following "saturation" set Σ_{SAT} will be fired:

$$\begin{aligned} E^*(v,v), D(x), D(z), D(y) & \rightarrow && E(x,z,y) \\ E^*(v,v), D(x), D(y) & \rightarrow && L(x,y), R(x,y), E^*(x,y) \end{aligned}$$

From here on, we assume that the schema $\mathbf{S}_{\mathcal{R}}$ is $\{E, L, R, D, E^*\}$, and that the reduction $\mathcal{R} \mapsto \Sigma_{\mathcal{R}}$ gives $\Sigma_{\mathcal{R}} = \{\Sigma_{\Theta}, \Sigma_{LR}, \Sigma_{AD}, \Sigma_{TC}, \Sigma_{SAT}\}$.

We denote by H_I the Herbrand base of instance I, i.e. the instance where, for each relation symbol $R \in \mathbf{S}_{\mathcal{R}}$, the interpretation R^{H_I} contains all tuples (of appropriate arity) that can be formed from the constants in $(dom(I) \cap \mathsf{Cons}) \cup \Delta$. The proof of the following lemma is straightforward:

Lemma 5.8. $H_I \models \Sigma_{\mathcal{R}}$, and $core(I) = H_I$, whenever H_I is a subinstance of I.

Lemma 5.9. Let I_0 be an arbitrary instance over schema $\mathbf{S}_{\mathcal{R}}$, and let $I_0, I_1, I_2, \ldots$ be the core chase sequence with $\Sigma_{\mathcal{R}}$ on I_0. If there is an $n \geq 0$, and a constant or variable v, such that $E^*(v, v) \in I_n$ (i.e. the graph G_{I_m} is cyclic for some $m \leq n$), then the core chase sequence is finite.

Proof:
(Sketch) First we note that $H_{I_n} = H_{I_0}$ for any instance I_n in the core chase sequence, since the chase does not add any new constants. If the core chase does not terminate at the instance I_n from the chase sequence, it follows that the dependencies in the set Σ_{SAT} will fire at I_n and generate H_{I_0} as a subinstance. It then follows from Lemma 5.8 that $I_{n+1} = H_{I_0}$. Since $H_{I_0} \models \Sigma_{\mathcal{R}}$ the core chase will terminate at instance I_{n+1}. □

Intuitively, the previous lemma guarantees that whenever we have a cycle in the initial instance the core chase sequence will terminate. Thus, in the following we will assume that the instances we consider are acyclic. Furthermore, in order not to unnecessarily terminate a core chase sequence, we need the following lemma that ensures us the core chase with $\Sigma_{\mathcal{R}}$ on an acyclic instance will not create any cycles.

Lemma 5.10. Let I_0 be an arbitrary instance over schema $\mathbf{S}_{\mathcal{R}}$, such that G_{I_0} is acyclic, and let $I_0, I_1, I_2, \ldots$ be the core chase sequence with $\Sigma_{\mathcal{R}}$ on I_0. Then G_{I_n} is acyclic, for all instances I_n in the sequence.

Proof:
(Sketch) Suppose to the contrary that G_{I_n} is cyclic, for some I_n in the sequence. Wlog we assume that I_n is the first such instance in the sequence. Clearly $n \geq 1$. This means that by applying all active triggers on I_{n-1} will add a cycle (note that the taking the core cannot add a cycle). Let $\Sigma' \subseteq \Sigma_{\mathcal{R}}$ be the dependencies that fired at I_{n-1}. First, it is easy to see that $\Sigma' \cap \Sigma_{LR} \varnothing$. This is because these dependencies do not introduce any new edges in G_{I_n} between vertices in $G_{I_{n-1}}$, they only add a new vertex into G_{I_n} which will have two incoming edges from vertices already in $G_{I_{n-1}}$. A similar reasoning shows that none of the tgds in Σ_{TC} or in Σ_{AD} can be part of Σ'. Finally, the dependencies in the set Σ_{SAT} may introduce cycles and may thus be part of Σ'. But the dependencies in Σ_{SAT} are fired only when $E^*(x, x) \in I_{n-1}$, which means that $G_{I_{n-1}}$ already contains a cycle, namely the self-loop on x. This contradicts our assumption that I_n is the first instance in the chase sequence that contains a cycle. □

We are going to transform an arbitrary instance I into an instance

$$I^* = \bigcup_{\pi \in paths(I)} I_{word(\pi)},$$

such that the core chase sequence with $\Sigma_{\mathcal{R}}$ on I terminates if and only if it terminates on I^*. In order for the construction to work, we first need to "chase out" of I all the Σ_{LR} dependencies.

Lemma 5.11. Let I be an arbitrary acyclic instance, and J the finite instance in which the core chase with Σ_{LR} on I terminates. Then the core chase sequence $I = I_0 \rightarrow_{\mathcal{R}} I_1 \rightarrow_{\mathcal{R}} I_2 \rightarrow_{\mathcal{R}} \cdots$ is finite if and only if the core chase sequence $J = J_0 \rightarrow_{\mathcal{R}} J_1 \rightarrow_{\mathcal{R}} J_2 \rightarrow_{\mathcal{R}} \cdots$ is finite.

Proof:
Since $I_0 \subseteq J$, the if-direction follows from the monotonicity of the chase [14]. The only-if direction follows from the observation that since J is finite, and $\Sigma_{LR} \subseteq \Sigma_{\mathcal{R}}$, there must be an $n \geq 0$, such that $J \subseteq I_n$. □

The next lemma is key to showing that $\mathcal{R}$ is uniformly terminating is and only if the core chase with $\Sigma_{\mathcal{R}}$ terminates on all instances.

Lemma 5.12. Let I be an arbitrary acyclic instance such that $I \models \Sigma_{LR}$, and let $I = I_0, I_1, I_2, \ldots, I_n$ and $I^* = J_0, J_1, J_2, \ldots, J_n$, $n \geq 0$, be core chase sequences with $\Sigma_{\mathcal{R}}$ on I and I^*, respectively. Then there is an bijection μ_n from $paths(I_n)$ to $paths(J_n)$, such that $word(\pi) = word(\mu_n(\pi))$, for each $\pi \in paths(I_n)$.

Proof:
We do an induction on n. For the base case, μ_0 maps each path $\pi \in paths(I_0)$ to the single element in $paths(I_{word(\pi)}) \subseteq paths(I_0^*)$.

For the inductive step, let $\pi_{n+1} \in paths(I_{n+1}) \setminus paths(I_n)$. Then π_{n+1} must have been created by the firing of a dependency from Σ_{Θ}, or by firing one or two dependencies from Σ_{LR}. In the first case there must be a path $\pi_n \in paths(I_n)$, such that the dependency fired on π_n creating the path π_{n+1} in I_{n+1}. By the inductive hypothesis, there is a unique path $\mu_n(\pi_n) \in paths(J_n)$. Since μ_n maps constants to themselves, the same dependency will fire on J_n, creating a unique path $\tau \in paths(I_{n+1})$ with $word(\tau) = word(\pi_{n+1})$. We then assign $\mu_{n+1}(\pi_{n+1}) = \tau$.

In the second case the path π_{n+1} was created by firing one or two dependencies from Σ_{LR}. Then there must be a unique path π_n in I_n, such that the dependency (or dependencies) fired on π_n, extending it to a path π_{n+1} in I_{n+1}. By the inductive hypothesis $\mu_n(\pi_n) \in paths(J_n)$. Again, the same dependency (or dependencies) fire on $\mu_n(\pi_n)$ in J_n and create a path τ in J_{n+1}, such that $\mu_n(\pi_n) \sqsubset \tau$. Then $\mu_n(\pi_n)$ is no longer a path in I_{n+1}, so we can assign $\mu_{n+1}(\pi_{n+1}) = \tau$. □

We are now ready for the main result of this section.

Theorem 5.13. A word rewriting system $\mathcal{R}$ is uniformly terminating if and only if the core chase with $\Sigma_{\mathcal{R}}$ terminates on all instances over $\mathbf{S}_{\mathcal{R}}$ (i.e. iff $\Sigma_{\mathcal{R}} \in \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$).

Proof:
First let us suppose that $\Sigma_{\mathcal{R}} \in \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$. Let $w \in \Delta^*$ be an arbitrary word. Because $\Sigma_{\mathcal{R}} \in \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$ it follows that the core chase will terminate also on instance I_w. From this and Theorem 5.6 it follows that the all derivation sequences in $\mathcal{R}$ originating from w are finite.

For the other direction suppose that $\Sigma_{\mathcal{R}} \notin \mathsf{CT}^{\mathsf{core}}_{\forall\forall}$. Then there exists an instance I, such that the core chase sequence $I = I_0, I_1, I_2, \ldots$ with $\Sigma_{\mathcal{R}}$ is infinite. From Lemma 5.9 we know that I is acyclic. From Lemma 5.12 it follows that the core chase of I^* with $\Sigma_{\mathcal{R}}$ must be infinite as well. Similarly to the second part of the proof of Theorem 5.6, there must be path $\pi \in I^*$ such that $\Sigma_{\mathcal{R}}$ admits an infinite derivation starting from $word(\pi)$. But this means that $\mathcal{R}$ is not uniformly terminating. □

Using the previous result, and the unsolvability of the uniform termination problem for word rewriting systems, we now have the main theorem.

Theorem 5.14. The membership problem for the set $\mathsf{CT}^{\mathsf{core}}_{\forall\forall}$ is Π^0_2-complete in the Arithmetical Hierarchy. ∎

Up to here we proved the undecidability of the $\mathsf{CT}^{\mathsf{core}}_{\forall\forall}$ class (and the $\mathsf{CT}^{\mathsf{core}}_{\forall\exists}$ class). Next we will show that this result can be extended with some minor changes to other termination classes as well.

Theorem 5.15. The membership problem for the set $\mathsf{CT}^{\mathsf{std}}_{\forall\exists}$ is Π^0_2-complete in the Arithmetical Hierarchy.

Proof:
(Sketch) It is easy to see that the same $\Sigma_{\mathcal{R}}$ reduction works for the $\mathsf{CT}^{\mathsf{std}}_{\forall\exists}$ case as well by choosing the branch that first applies all the dependencies in $\Sigma_{AD} \cup \Sigma_{TC} \cup \Sigma_{SAT}$. This is because in case the initial arbitrary instance contains a cycle the full dependencies $AD \cup TC \cup S$ will saturate the instance and the standard chase will terminate. If I does not contain any cycles then, as we showed, during the chase process no cycles are added and the termination proof is the same as for the core chase, mutatis mutandis. □

To show that the basic $\mathcal{R} \mapsto \Sigma_{\mathcal{R}}$ reduction cannot be used for the $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ class. Consider the word rewriting system $\mathcal{R} = (\{0,1\},(1,0))$ and instance $I = \{E(x,0,x), L(x,y)\}$. It is easy to see that the branch that applies the ξ_{L_0} dependency first will not terminate as it will generate the following infinite set of tuples:

E		
a	0	a
x_1	0	b
x_2	0	x_1
	...	
x_n	0	x_{n-1}
	...	

L	
a	b
a	x_1
a	x_2
...	
a	x_n
...	

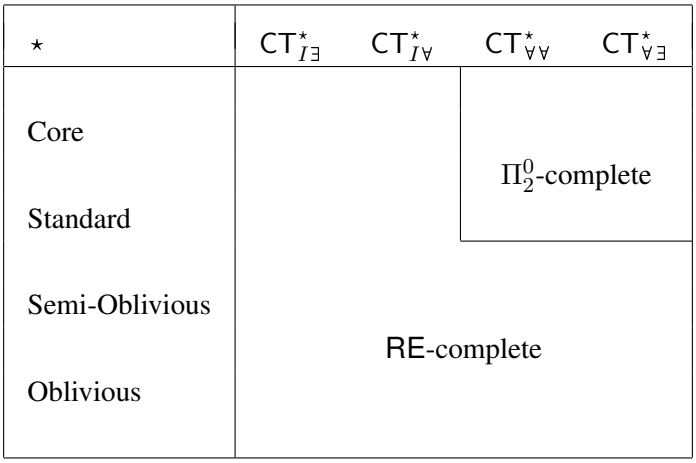
On the other hand, it is clear that the word rewriting system $\mathcal{R} = (\{0,1\},(1,0))\}$ is uniformly terminating.

The unsolvability result can still be obtained for the $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ class if we allow *denial constraints*, i.e. tgds where the head is the constant *False*. A denial constraint $\alpha \rightarrow \bot$ satisfied by an instance I if there is no homomorphism h, such that $h(\alpha) \subseteq I$. If a denial constraint is violated, the chase terminates (in failure). Then we simply define $\Sigma_F = \{E^*(x,x) \rightarrow \bot\} \cup \Sigma_{\mathcal{R}} \setminus \Sigma_{SAT}$.

Theorem 5.16. Let Σ be a set of tgds and one denial constraint. Then the membership problem $\Sigma \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ is Π^0_2-complete in the Arithmetical Hierarchy.

Proof:
(Sketch) Similarly to the proof of Theorem 5.15 it is easy to see that if an arbitrary instance I contains a cycle, then the standard chase on I with Σ_F will terminate on all branches. This is because the fairness conditions guarantees that the denial constraint will be fired, and the chase will terminate. □

We can now summarize the results of this section together with the results cited in Theorem 5.1, yielding the following table.



6. Guaranteed termination

To overcome the undecidability of chase termination, a flurry of restricted classes of tgds have been proposed in the literature. In this section we review these restricted classes with the purpose of determining whether structure and termination properties applies to other chase variations and not only for the ones that they were initially studied. We will show that many such termination classes (mostly developed for the standard chase) also ensures termination of less computationally-expensive chase variations.

The termination of the oblivious chase can be related to the termination of the standard chase by using the *enrichment* transformation introduced by [19]. The enrichment takes a tgd of the form $\xi = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z})$ over schema $\mathbf{R}$ and converts it into tuple generating dependency $\hat{\xi} = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z}), H(\bar{x},\bar{y})$, where H is a new relational symbol that does not appear in $\mathbf{R}$. For a set Σ of tgds the transformed set is $\widehat{\Sigma} = \{\hat{\xi} : \xi \in \Sigma\}$. Using the enrichment notion the following was shown.

Theorem 6.1.*[19]* $\Sigma \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ if and only if $\widehat{\Sigma} \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

Proof:
Let $\Sigma \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ be a set of tgds. Suppose now that there is an instance I for which the standard chase with $\widehat{\Sigma}$ does not terminate. This means that there is an infinite standard-chase sequence

$$I = I_0 \xrightarrow{(\hat{\xi}_0,h_0)} I_1 \xrightarrow{(\hat{\xi}_1,h_1)} \ldots\ldots \xrightarrow{(\hat{\xi}_{n-1},h_{n-1})} I_n \xrightarrow{(\hat{\xi}_n,h_n)} \ldots.$$

Thus $h_i(body(\hat{\xi}_i)) \subseteq I_i$, for all $i \geq 0$. Since $body(\hat{\xi}_i) = body(\xi_i)$, we have that

$$I = J_0 \xrightarrow{(\xi_0,h_0)} J_1 \xrightarrow{(\xi_1,h_1)} \ldots\ldots \xrightarrow{(\xi_{n-1},h_{n-1})} J_n \xrightarrow{(\xi_n,h_n)} \ldots.$$

where J_i is the same as I_i restricted to the atoms in initial schema, is an infinite oblivious-chase sequence with Σ and I. From this it follows by contraposition that $\Sigma \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ implies $\widehat{\Sigma} \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

For the second part, suppose that there is an instance I for which the oblivious chase with Σ does not terminate. Then there is an infinite oblivious-chase sequence

$$I = I_0 \xrightarrow{(\xi_0,h_0)} I_1 \xrightarrow{(\xi_1,h_1)} \ldots\ldots \xrightarrow{(\xi_{n-1},h_{n-1})} I_n \xrightarrow{(\xi_n,h_n)} \ldots.$$

Let $J_0 = I_0$, and for all $i \geq 0$, let $J_{i+1} = I_{i+1} \cup \{H(h_i(\bar{x}), h_i(\bar{y}))\}$, where H is the enrichment atom used in $\hat{\xi}_i$. We claim that

$$I = J_0 \xrightarrow{(\hat{\xi}_0,h_0)} J_1 \xrightarrow{(\hat{\xi}_1,h_1)} \ldots\ldots \xrightarrow{(\hat{\xi}_{n-1},h_{n-1})} J_n \xrightarrow{(\hat{\xi}_n,h_n)} \ldots.$$

is an infinite standard-chase sequence with $\widehat{\Sigma}$ and I. Towards a contradiction, suppose it is not. Then there must be an $i \geq 0$, such that the standard-chase step cannot be applied with h_i and $\hat{\xi}_i$ on J_i. Let $\xi_i = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z})$. Then $\hat{\xi}_i = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z}), H(\bar{x},\bar{y})$. If $(\hat{\xi}_i, h_i)$ is not an active trigger for J_i, there exists an extension h_i' of h_i such that $h_i'(body(\hat{\xi}_i)) \subseteq J_i$. Since h_i' is an extension of h_i, it follows that $h_i'(\bar{x}) = h_i(\bar{x})$ and $h_i'(\bar{y}) = h_i(\bar{y})$, meaning that $H(h_i(\bar{x}), h_i(\bar{y})) \in J_i$. Because the facts over H are only introduced by the standard chase, it follows that the homomorphism h_i has already been applied with $\hat{\xi}_i$ earlier in the standard-chase sequence. But then h_i must also have been applied with ξ_i at the same earlier stage in the oblivious-chase sequence. This is a contradiction, since it entails that trigger (ξ_i, h_i) would have been applied twice in the oblivious-chase sequence. □

To relate the termination of the semi-oblivious chase to the standard chase termination, we use a transformation similar to the enrichment. This transformation is called *semi-enrichment* and takes a tuple generating dependency $\xi = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z})$ over a schema $\mathbf{R}$ and converts it into the tgd $\tilde{\xi} = \alpha(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\ \beta(\bar{x},\bar{z}), H(\bar{x})$, where H is a new relational symbol which does not appear in $\mathbf{R}$. For a set Σ of tgds defined on schema $\mathbf{R}$, the transformed set is $\widetilde{\Sigma} = \{\tilde{\xi} : \xi \in \Sigma\}$. Using the semi-enrichment notion, the standard and the semi-oblivious chase can be related as follows.

Theorem 6.2. $\Sigma \in \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$ if and only if $\widetilde{\Sigma} \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

Proof:
Similar to the proof of Theorem 6.1. □

A class of sets of tgds $\mathscr{C}$ is said to be *closed under enrichment* if $\Sigma \in \mathscr{C}$ implies that $\widehat{\Sigma} \in \mathscr{C}$. Using this notation together with Theorem 6.1 gives us a sufficient condition for a class of dependencies to belong to $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$:

Proposition 6.3. Let $\mathscr{C} \subseteq \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ such that $\mathscr{C}$ is closed under enrichment. Then $\mathscr{C} \subseteq \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$.

Proof:
Follows directly from Theorem 6.1. □

Using this proposition we will reveal classes of dependencies that ensure termination for the oblivious chase. Similarly we define the notion of *semi-enrichment closure* for classes of dependency sets. The semi-enrichment closure property gives a sufficient condition for the semi-oblivious chase termination.

Proposition 6.4. Let $\mathscr{C} \subseteq \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ such that $\mathscr{C}$ is closed under semi-enrichment. Then $\mathscr{C} \subseteq \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.

Proof:
Follows directly from Theorem 6.2. □

As we will see next, most of the known classes that ensure the standard chase termination are closed under semi-enrichment, and thus those classes actually guarantee the semi-oblivious chase termination as well. As we saw in Section 3, the semi-oblivious chase has a lower complexity that the standard chase.

Acyclicity based classes

As full tgds do not generate any new nulls during the chase, any sequence with a set of full tgds will terminate since there only is a finite number of tuples that can be formed out of the elements of the domain of the initial instance. The cause of non-termination lies in the existentially quantified variables in the head of the dependencies. Most restricted classes thus rely on restricting the tgds in a way that prevents these existential variables to participate in any recursion.

The class WA of *weakly acyclic* sets of tgds, introduced by [14], was one of the first restricted classes to be proposed. Consider

$$\Sigma_1 \quad = \quad \{R(x,y) \rightarrow \exists z S(z,x)\}.$$

Let $(R,1)$ denote the first position in R, and $(S,2)$ the second position in S, and so on. In a chase step based on this dependency the values from position $(R,1)$ get copied into the position $(S,2)$, whereas the value in position $(R,1)$ "cause" the generation of a new null value in $(S,1)$. This structure can been seen in the *dependency graph* of Σ_1 that has a "copy" edge from vertex $(R,1)$ to vertex $(S,2)$, and a "generate" edge from vertex $(R,1)$ to vertex $(S,1)$. Note that the graph does not consider any edges from $(R,2)$ because variable y does not contribute to the generated values. The

chase will terminate since there is no recursion going through the $(S, 2)$ position. By contrast, the dependency graph of

$$\Sigma_2 \quad = \quad \{R(x, y) \rightarrow \exists z\, R(y, z)\}$$

has a generating edge from $(R, 2)$ to $(R, 2)$. It is the generating self-loop at $(R, 2)$ which causes the chase, for example the instance $\{R(a, b)\}$, to converge only at the infinite instance $\{R(a, b), R(b, z_1)\} \cup \{R(z_i, z_{i+1}) : i > 0\}$. The class of weakly acyclic tgds (WA) is defined by [14] to be those sets of tgds whose dependency graph doesn't have any cycles involving a generating edge. It is easy to observe that the class WA is closed under semi-enrichment but it is not closed under enrichment. This is because in the case of semi-enrichment the new relational symbol H considered for each dependency contains only variables that appears both in the body and the head of the dependency, and the new H atoms appear only in the heads of the semi-enriched dependency. This means that the dependency graph for a semi-enriched set of WA tgds will only add edges oriented into positions associated with the new relational symbol. The set $\Sigma = \{R(x, y) \rightarrow \exists z\, R(x, z)\}$ shows that this is not the case for enrichment as $\Sigma \in \mathsf{WA}$ but $\widehat{\Sigma} \notin \mathsf{WA}$.

The slightly smaller class of sets of tgds with *stratified witness* (SW) was introduced by [15] around the same time as WA. An intermediate class, the *richly acyclic* tgds (RA) was introduced by [16] in a different context and it was later shown by [19] that $\mathsf{RA} \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. It can be easily verified that both classes SW and RA are closed under enrichment. The *safe dependencies* (SD), introduced by [17], and the *super-weakly acyclic* (sWA), introduced by [8], are both generalizations of the WA class, and both are close under semi-enrichment.

All of these classes have been proven to have PTIME membership tests, and have the following properties.

Theorem 6.5. *[15, 14, 17, 8, 19]*

1. $\mathsf{SW} \subset \mathsf{RA} \subset \mathsf{WA} \subset \mathsf{SD} \subset \mathsf{sWA}$.
2. $\mathsf{WA} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$, $\mathsf{RA} \subset \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$, and $\mathsf{sWA} \subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.

In order to complete the picture suggested by the previous theorem we need a few more results. Consider

$$\Sigma_3 \quad = \quad \{R(x, y) \rightarrow \exists z\, R(x, z)\}.$$

Clearly $\Sigma_3 \in \mathsf{WA}$. Let $I_0 = \{R(a, b)\}$, and consider a oblivious chase sequence $I_0, I_1, I_2, \ldots$. It is easy to see that for any I_n, where $n > 0$, there exists a (non-active) trigger $(\xi, \{x/a, y/z_n\})$, meaning that the oblivious chase will not terminate. Thus we have $\Sigma_3 \notin \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. On the other hand, for the set

$$\Sigma_4 \quad = \quad \{S(y), R(x, y) \rightarrow \exists z\, R(y, z)\},$$

we have $\Sigma_4 \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. Furthermore, $\Sigma_4 \notin \mathsf{WA}$, since the dependency graph of Σ_4 will have a generating self-loop on vertex $(R, 2)$. This gives us

Proposition 6.6. The classes WA and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ are incomparable wrt inclusion.

It was shown in [17] that $\mathsf{WA} \subset \mathsf{SD}$ and also that $\mathsf{SD} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$. We can now extend this result by showing that, similarly to the WA class, the following holds:

Proposition 6.7. The classes SD and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ are incomparable wrt inclusion.

Proof:
(Sketch) The proof consists of showing that $\Sigma_3 \in \mathsf{SD} \setminus \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$, and showing that $\Sigma_5 \in \mathsf{CT}^{\mathsf{obl}}_{\forall\forall} \setminus \mathsf{SD}$, where

$$\Sigma_5 \quad = \quad \{R(x,x) \rightarrow \exists y\, R(x,y)\}.$$

Details are omitted. □

From the semi-enrichment closure of the WA and SD classes and Proposition 6.4 we get the following result.

Proposition 6.8. $\mathsf{WA} \in \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$ and $\mathsf{SD} \in \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$. Furthermore, for any instance I and any $\Sigma \in \mathsf{SD}$, the semi-oblivious chase with Σ on I terminates in time polynomial in the number of tuples in I.

Note that the previous result follows directly also from a similar result for the class sWA presented by [8]. Still, as shown by the following proposition, the super-weakly acyclic class does not include the class of dependencies that ensures termination for the oblivious chase variation, nor does the inclusion hold in the other direction.

Proposition 6.9. sWA and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ are incomparable wrt inclusion.

Proof:
(Sketch) We exhibit the super-weakly acyclic set $\Sigma_3 = \{R(x,y) \rightarrow \exists z\, R(x,z)\}$. It is clear $\Sigma_3 \notin \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. For the converse, let

$$\Sigma_6 \quad = \quad \{S(x), R(x,y) \rightarrow \exists z\, R(y,z)\}.$$

Then $\Sigma_6 \notin \mathsf{sWA}$, on the other hand it can be observed that the oblivious chase with Σ_6 terminates on all instances. This is because tuples with new nulls cannot cause the dependency to fire, as these new nulls will never be present in relation S. □

Stratification based classes

Consider $\Sigma_7 = \{\xi_1, \xi_2\}$, where

$$\begin{aligned} \xi_1 &= R(x,x) \rightarrow \exists z\, S(x,z), \text{ and} \\ \xi_2 &= R(x,y), S(x,z) \rightarrow R(z,x). \end{aligned}$$

In the dependency graph of Σ_7 we will have the cycle $(R,1) \rightsquigarrow (S,2) \rightsquigarrow (R,1)$, and since $(S,2)$ is an existential position, the set Σ_7 is not weakly acyclic. However, it is easy to see that $\Sigma_7 \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$. It is also easily seen that if S is empty and R non-empty, then ξ_1 will "cause" ξ_2 to fire for every tuple in R. Let us denote this "causal" relationship by $\xi_1 \prec \xi_2$. On the other hand, there in no chase sequence in which a new null in $(S,2)$ can be propagated back to a tuple in R and fire a trigger based on ξ_1, thereby creating an infinite loop. We denote this with $\xi_2 \not\prec \xi_1$. In comparison, when chasing with

$$\Sigma_8 \quad = \quad \{R(x,y) \rightarrow \exists z\, R(z,x)\},$$

the new null z_n in (z_n, z_{n-1}) will propagate into tuple (z_{n+1}, z_n), in an infinite regress. If we denote the tgd in Σ_8 with ξ, we conclude that $\xi \prec \xi$. A formal definition of the $\prec$ relation is given in the Appendix.

The preceding observations led [6] to define the class of *stratified dependencies* by considering the *chase graph* of a set Σ, where the individual tgds in Σ are the vertices and there is an edge from ξ_1 to ξ_2 when $\xi_1 \prec \xi_2$. A set Σ is then said to be stratified if the vertex-set of every cycle in the chase graph forms a weakly acyclic set. The class of all sets of stratified tgds is denoted Str. In the previous example, $\Sigma_7 \in \mathsf{Str}$, and $\Sigma_8 \notin \mathsf{Str}$.

[17] observed that $\mathsf{Str} \not\subseteq \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ and that actually only $\mathsf{Str} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$, and came up with a corrected definition of $\prec$, which yielded the *corrected stratified* CStr of tgds[3], for which they showed

Theorem 6.10. *[17]*

$$\mathsf{CStr} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall},\ \mathsf{Str} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists},\ \textit{and}\ \mathsf{CStr} \subset\ \mathsf{Str}.$$

From the observation that the CStr class is closed under semi-enrichment and from Proposition 6.4 we have:

Proposition 6.11.

1. $\mathsf{CStr} \subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.
2. CStr and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ are incomparable wrt inclusion.

Proof:
(Sketch) For the second part we have $\Sigma_3 \in \mathsf{CStr}$ and $\Sigma_3 \notin \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. For the converse consider the dependency set Σ_6 from the proof of Proposition 6.9. □

[33] further observed that the basic stratification definition also catches some false negatives. For this they considered the dependency set $\Sigma_9 = \{\xi_3, \xi_4\}$, where

$$\begin{aligned} \xi_3 &= S(x), E(x,y) \rightarrow E(y,x), \text{ and} \\ \xi_4 &= S(x), E(x,y) \rightarrow \exists z\, E(y,z), E(z,x). \end{aligned}$$

[3]For a definition of CStr, see the Appendix.

Here ξ_3 and ξ_4 belong to the same stratum according to the definition of CStr. Since new nulls in both $(E,1)$ and $(E,2)$ can be caused by $(E,1)$ and $(E,2)$, there will be generating self-loops on these vertices in the dependency graph. Hence $\Sigma_9 \notin \mathsf{CStr}$. On the other hand, it is easy to see that the number of new nulls that can be generated in the chase is bounded by the number of tuples in relation S in the initial instance. Consequently $\Sigma_9 \in \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

In order to avoid such false negatives, [33] gave an alternative definition of the $\prec$ relation and of the chase graph. Both of theses definitions are rather involved technically, see the Appendix. The new *inductively restricted* class, abbreviated IR, restricts each connected component in the modified chase graph to be in SD. In example above, $\Sigma_9 \in \mathsf{IR}$.

[33] also observed that IR only catches *binary* relationships $\xi_1 \prec \xi_2$. This could be generalized to a ternary relation $\prec(\xi_1, \xi_2, \xi_3)$, meaning that there exists a chase sequence such that firing ξ_1 will cause ξ_2 to fire, and this in turn causes ξ_3 to fire. This will eliminate those cases where ξ_1, ξ_2 and ξ_3 form a connected component in the (modified) chase graph, and yet there is no chase sequence that will fire ξ_1, ξ_2 and ξ_3 in this order. Thus the tree dependencies should not be in the same stratum.

Similarly to the ternary extension, the $\prec$ relation can be generalized to be k-ary. The resulting termination classes are denoted $\mathsf{T}[k]$. Thus $\mathsf{T}[2] = \mathsf{IR}$, and in general $\mathsf{T}[k] \subset \mathsf{T}[k+1]$ as introduced by [33]. The main property is

Theorem 6.12. *[17]*

$$\mathsf{CStr} \subset IR = \mathsf{T}[2] \subset \mathsf{T}[3] \subset \cdots \subset \mathsf{T}[k] \subset \cdots \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}.$$

To complete the picture, we have the following proposition based on the semi-oblivious closure for the T[k] hierarchy and Proposition 6.4.

Proposition 6.13.

1. $\mathsf{T}[k] \subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.
2. $\mathsf{T}[k]$ and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$ are incomparable wrt inclusion.

Before concluding this section need to mention that all the classes discussed here are closed under semi-enrichment, thus they ensure the termination for the less expensive semi-oblivious chase in a polynomial number of steps, in the size of the input instance. Also need to note that all the previous classes were extended by [18] using an innovative rewriting approach.

The Hasse diagram in Figure 2 summarizes all considered classes and their termination properties.

7. Complexity of stratification

As we noted in Section 6, all the acyclicity based classes have the property that testing whether a given set Σ belongs to it can be done in PTIME. The situation changes when we move to the stratified classes. [6] claimed that testing if $\xi_1 \prec \xi_2$ is in NP for a given ξ_1 and ξ_2, thus resulting in Str having a coNP membership problem. We shall see in Theorem 7.2 below that this cannot be the case, unless NP = coNP. We shall use the $\prec$ order as it is defined for the CStr class. The results also

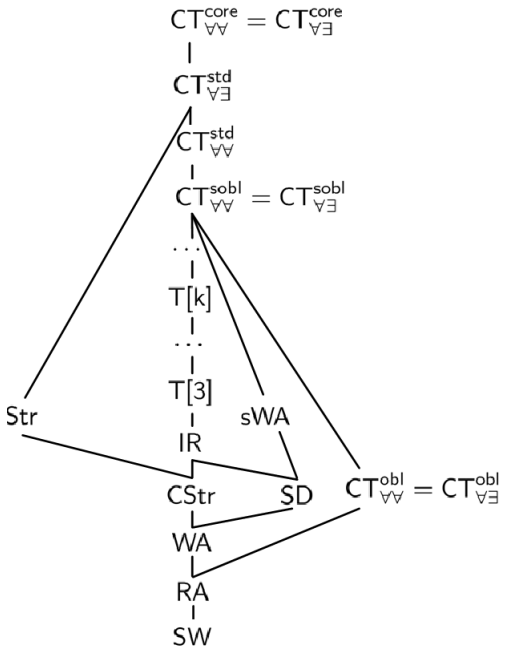


Figure 2. Sufficient classes.

hold for the Str class. First we need a formal definition. Given a tgd ξ and a mapping h from the universally quantified variables in ξ to $dom(I)$, by $h(\xi)$ we denote the formula obtained by replacing all universally quantified variables x with $h(x)$.

Definition 7.1. [17] Let ξ_1 and ξ_2 be tgds. Then ξ_1 *precedes* ξ_2, denoted $\xi_1 \prec \xi_2$, if there exists an instance I and homomorphisms h_1 and h_2 from the universal variables in ξ_1 and ξ_2 to $dom(I)$, such that:

(i) $I \models h_2(\xi_2)$, and

(ii) $I \xrightarrow{(\xi_1,h_1)} J$ using an oblivious chase step, and

(iii) $J \not\models h_2(\xi_2)$.

Note that the pair (ξ_1, h_1) in the previous definition denotes a trigger, not necessarily an active trigger, because the chase step considered is the oblivious one. Intuitively, the instance I in the definition is a witness to the "causal" relationship between ξ_1 and ξ_2 (via h_2), as $h_2(\xi_2)$ won't fire at I, but will fire once ξ_1 has been applied to I. The notion of stratum of Σ is as before, i.e. we build a chase graph consisting of a vertex for each tgd in Σ, and an edge from ξ_1 to ξ_2 if $\xi_1 \prec \xi_2$. Then ξ_1 and ξ_2 are in the same *stratum* when they both belong to the same cycle in the chase graph of Σ. A set Σ of tgds is said to be *C-stratified* (CStr) if all its strata are weakly acyclic [17].

Theorem 7.2.

1. Given two tgds ξ_1 and ξ_2, the problem of testing if $\xi_1 \prec \xi_2$ is coNP-hard.
2. Given a set of dependencies Σ, the problem of testing if $\Sigma \in \mathsf{CStr}$ is NP-hard.

Proof:
For part 1 of the theorem we will use a reduction from the graph 3-colorability problem that is known to be NP-complete. It is also well known that a graph G is 3-colorable iff there is a homomorphism from G to K_3, where K_3 is the complete graph with 3 vertices. We provide a reduction $G \mapsto \{\xi_1, \xi_2\}$, such that G is not 3-colorable if and only if $\xi_1 \prec \xi_2$.

We identify a graph $G = (V, E)$, where $|V| = n$ and $|E| = m$ with the sequence

$$G(x_1, \ldots, x_n) = E(x_{i_1}, y_{i_1}), \ldots, E(x_{i_m}, y_{i_m}),$$

and treat the elements in V as variables. Similarly, we identify the graph K_3 with the sequence:

$$K_3(z_1, z_2, z_3) = (E(z_1, z_2), E(z_2, z_1), E(z_1, z_3), E(z_3, z_1), E(z_2, z_3), E(z_3, z_2))$$

where z_1, z_2, and z_3 are variables. With these notations, given a graph $G = (V, E)$, we construct tgds ξ_1 and ξ_2 as follows:

$$\begin{aligned} \xi_1 = &\quad R(z) \rightarrow \exists z_1, z_2, z_3\ K_3(z_1, z_2, z_3), \text{ and} \\ \xi_2 = &\ E(x, y) \rightarrow \exists x_1, \ldots, x_n\ G(x_1, \ldots, x_n). \end{aligned}$$

Clearly the reduction is polynomial in the size of G. We will now show that $\xi_1 \prec \xi_2$ iff G is not 3-colorable.

First, suppose that $\xi_1 \prec \xi_2$. Then there exists an instance I and homomorphisms h_1 and h_2, such that $I \models h_2(\xi_2)$. Consider J, where $I \xrightarrow{(\xi_1, h_1)} J$. Thus R^I had to contain at least one tuple, and E^I had to be empty, because otherwise the monotonicity property of the chase would imply that that $J \models h_2(\xi_2)$.

On the other hand, we have $I \xrightarrow{(\xi_1, h_1')} J$, where the instance $J = I \cup \{K_3(h_1'(z_1), h_1'(z_2), h_1'(z_3))\}$, and h_1' is a distinct extension of h_1. Since $E^I = \emptyset$, and we assumed that $J \not\models h_2(\xi_2)$, it follows that there is no homomorphism from G into J, i.e. there is no homomorphism from $G(h_2'(x_1), \ldots, h_2'(x_n))$ to $K_3(h_1'(z_1), h_1'(z_2), h_1'(z_3))$, where h_2' is a distinct extension of h_2. Therefore the graph G is not 3-colorable.

For the other direction, let us suppose that graph G is not 3-colorable. This means that there is no homomorphism from G into K_3. Consider now $I = \{R(a)\}$, homomorphism $h_1 = \{z/a\}$, and homomorphism $h_2 = \{x/h_1'(z_1), y/h_1'(z_2)\}$. It is easy to verify that I, h_1 and h_2 satisfy the three conditions for $\xi_1 \prec \xi_2$.

For part 2 of the theorem, consider the set $\Sigma = \{\xi_1, \xi_2\}$ defined as follows:

$$\begin{aligned} \xi_1 &= R(z_1, v) \rightarrow \exists z_2, z_3, w\ K_3(z_1, z_2, z_3), R(z_2, w), R(z_3, w), S(w), \text{ and} \\ \xi_2 &= E(x, y) \rightarrow \exists x_1, \ldots, x_n, v\ G(x_1, \ldots, x_n), R(x, v). \end{aligned}$$

It is straightforward to verify that $\Sigma \notin \mathsf{WA}$ and that $\xi_2 \prec \xi_1$. Similarly to the proof of part 1, it can be shown that $\xi_1 \prec \xi_2$ iff the graph G is not 3-colorable. From this follows that $\Sigma \in \mathsf{CStr}$ iff there is no cycle in the chase graph iff $\xi_1 \not\prec \xi_2$ iff G is 3-colorable. □

Note that the reduction in the previous proof can be used to show that the problem "$\Sigma \in \mathsf{Str}$?" is NP-hard. Similar result can be also obtained for the IR class and also for the *local stratification* based classes introduced by [20]. The obvious upper bound for the problem $\xi_1 \prec \xi_2$ is given by:

Proposition 7.3. Given dependencies ξ_1 and ξ_2, the problem of determining whether $\xi_1 \prec \xi_2$ is in Σ_2^{p}.

Proof:
From [6] we know that if $\xi_1 \prec \xi_2$ there is an instance I satisfying Definition 7.1, such that size of I is bounded by a polynomial in the size of $\{\xi_1, \xi_2\}$. Thus, we can guess instance I, homomorphisms h_1 and h_2 in NP time. Next, with a NP oracle we can check if $I \models h_2(\xi_2)$ and $J \not\models h_2(\xi_2)$, where $I \xrightarrow{(\xi_1, h_1)} J$. □

We shall see that the upper bound of the proposition actually can be lowered to Δ_2^{p}. For this we need the following characterization theorem.

Theorem 7.4. Let $\xi_1 = \alpha_1 \rightarrow \beta_1$ and $\xi_2 = \alpha_2 \rightarrow \beta_2$ be tgds. Then, $\xi_1 \prec \xi_2$ if and only if there is an atom t, and (partial) mappings h_1 and h_2 on Vars, such that the following hold.

(a) $t \in \beta_1$,

(b) $h_1(t) \in h_2(\alpha_2)$,

(c) $h_1(t) \notin h_1(\alpha_1)$, and

(d) There is no idempotent homomorphism from $h_2(\beta_2)$ to $h_2(\alpha_2) \cup h_1(\alpha_1) \cup h_1(\beta_1)$.

Proof:
We first prove the "only if" direction. For this, suppose that $\xi_1 \prec \xi_2$, that is, there exists an instance I and mappings g_1 and g_2, such that conditions (i) – (iii) of Definition 7.1 are fulfilled.

From conditions (ii) and (iii) we have that $g_1(\alpha_1) \subseteq I$ and $g_1'(\beta_1) \not\subseteq I$, for any distinct extension g_1' of g_1.

Now, consider $h_1 = g_1$ and $h_2 = g_2$. Let t be an atom from β_1 such that $h_1'(t) \in h_1'(\beta_1) \cap h_2(\alpha_2)$ and $h_1'(t) \notin h_1(\alpha_1)$, for an extension h_1' of h_1. Such an atom t must exists, since otherwise it will be that $h_1'(\beta_1) \cap h_2(\alpha_2) \subseteq h_1(\alpha_1)$, which is not possible because of conditions (i) and (iii) (note that $h_1 = g_1$). It is now easy to see that t, h_1' and h_2 satisfy conditions $(a), (b)$, and (c) of the theorem. It remains to show that condition (d) also is satisfied. By construction we have $J = I \cup h_1'(\beta_1)$. It now follows that $I \cup h_1'(\beta_1) \not\models h_2(\xi_2)$. Because $h_1(\alpha_1) \subseteq I$, condition (d) is indeed satisfied.

For the "if" direction of the theorem, suppose that there exists an atom t and homomorphisms h_1 and h_2, such that conditions $(a), (b), (c)$ and (d) holds. Let $g_1 = h_1$, $g_2 = h_2$ and let $I = (h_1(\alpha_1) \cup$

$h_2(\alpha_2)) \setminus h_1'(t)$, for a distinct extension h_1' of h_1. Because $h_1'(t) \notin I$ and $h_1'(t) \in h_2(\alpha_2)$, it follows that $h_2(\alpha_2) \not\subseteq I$. Thus we have $I \models h_2(\xi_2)$, proving point (i) of Definition 7.1. On the other hand, because point (c) of the theorem is assumed, it follows that $h_1(\alpha_1) \subseteq I$, from which we get $I \xrightarrow{(\xi_1, h_1)} J$, where $J = I \cup h_1'(\beta_1)$, proving points (i) and (ii) from Definition 7.1. Since $I \cup h_1'(\beta_1) = h_1(\alpha_1) \cup h_2(\alpha_2) \cup h_1'(\beta_1)$, and point (d) holds, we get $J \not\models h_2(\xi_2)$, thus showing that condition (iii) of Definition 7.1 is also satisfied.[4] □

With this characterization result we can now tighten the Σ_2^p upper bound of Proposition 7.3 as follows:

Theorem 7.5. Given dependencies ξ_1 and ξ_2, the problem of determining whether $\xi_1 \prec \xi_2$ is in Δ_2^{p}.

Proof:
For this proof we will use the characterization Theorem 7.4, and the observation that $\Delta_2^{\mathsf{p}} = \mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{coNP}}$. Consider the following PTIME algorithm that enumerates all possible h_1, h_2 and t:

for all $t \in \beta_1$
  for all $(h_1, h_2) \in mgu(t, \alpha_2)$
    if $h_1(t) \notin h_1(\alpha_1)$ **return** t, h_1, h_2

In the algorithm, $mgu(t, \alpha_2)$ denotes all pairs (h_1, h_2) such that there exists an atom $t' \in \alpha_2$, with $h_1(t) = h_2(t')$, and there is no (g_1, g_2) and f different from the identity mappings, such that $h_1 = g_1 \circ f$ and $h_2 = g_2 \circ f$.

Using the values returned by previous algorithm and with a coNP oracle we can test if point (d) holds. Thus, the problem is in Δ_2^{p}. □

Armed with these results we can now state the upper-bound for the complexity of the CStr membership problem.

Theorem 7.6. Let Σ be a set of tgds. Then the problem of testing if $\Sigma \in \mathsf{CStr}$ is in Π_2^{p}.

Proof:
(Sketch) To prove that Σ is not in CStr guess a set of tuples $(\xi_1, t^1, h_1^1, h_2^1), \ldots, (\xi_k, t^k, h_1^k, h_2^k)$, where $\xi_1, \ldots, \xi_k$ are tgds in Σ, $t_1, \ldots, t_k$ are atoms, and h_1^i,h_2^i are homomorphisms, for $i \in \{1, \ldots, k\}$. Then, using an NP oracle check that $\xi_i \prec \xi_{i+1}$, for $i \in \{1, \ldots, k-1\}$, and that $\xi_k \prec \xi_1$, using the characterization Theorem 7.4 with t^i, h_1^i, h_2^i and t^k, h_1^k, h_2^k respectively. And then check in PTIME if the set of dependencies $\{\xi_1, \ldots, \xi_k\}$ is not weakly acyclic. Thus, the complexity is in Π_2^{p}. □

[4] It is easy to note that by adding the extra condition "(e) *there is no idempotent homomorphism from* β_1 *to* α_1" in the previous theorem we obtain a characterization of the stratification order associated with the Str class.

We note that using the obvious upper-bound Σ_2^{p} for testing if $\xi_1 \prec \xi_2$, the membership problem for the class CStr would be in Π_3^{p}. As mentioned the same results apply also for the class Str. Even if the complexity bounds for testing if $\xi_1 \prec \xi_2$ are not tight, it can be noted that a coNP upper bound would not lower the Π_2^{p} upper bound of the membership problem for CStr.

The complexities of the various restricted classes is shown in the diagram below. The diagram also shows the termination classes $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$, $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$, $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ and $\mathsf{CT}^{\mathsf{core}}_{\forall\forall}$.

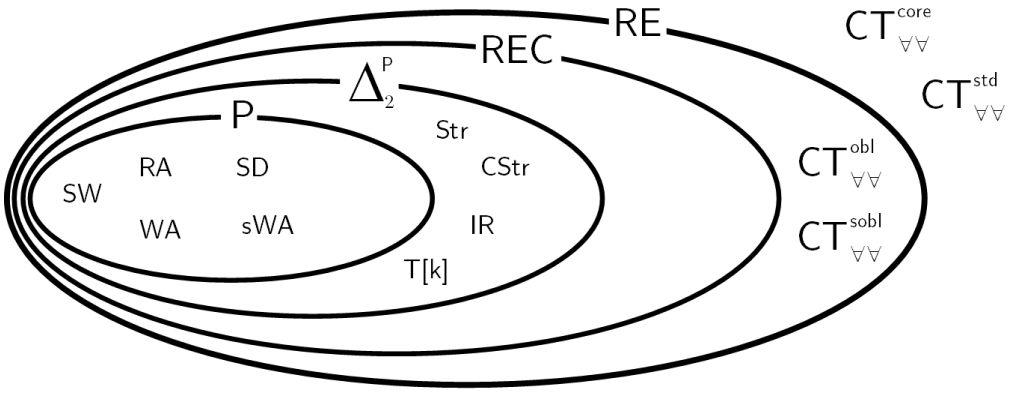


Figure 3. Membership complexity for restricted dependency classes.

8. Conclusions

We have undertaken a systematization of the somewhat heterogeneous area of the chase. Our analysis produced a taxonomy of the various chase versions and their termination properties, showing that the main sufficient classes that guarantee termination for the standard chase also ensures termination for the complexity-wise less expensive semi-oblivious chase. Even if the standard chase procedure in general captures more sets of dependencies that ensure the chase termination than the semi-oblivious chase, we argue that for most practical constraints the semi-oblivious chase is a better choice. We have also proved that the membership problem for the classes $\mathsf{CT}^{\mathsf{core}}_{\forall\forall}$, $\mathsf{CT}^{\mathsf{core}}_{\forall\exists}$ and $\mathsf{CT}^{\mathsf{std}}_{\forall\exists}$ are Π_2^0-complete, and in case we also at least one denial constraint the same holds for $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$, $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$ and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$. Still, it is yet unknown if the membership problem for $\mathsf{CT}^{\mathsf{std}}_{\forall\forall}$ remains Π_2^0-complete without denial constraints. Gogacz and Marcinkowski [13] conjecture that the problem actually is coRE-complete, and show that this is true for the classes $\mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$ and $\mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$, without using denial constraints.

Finally we have analyzed the complexity of the membership problem for the class of stratified sets of dependencies. Our bounds for this class are not tight, and it remains an open problem to determine the complexity exactly.

References

[1] Beeri C, Vardi MY. The Implication Problem for Data Dependencies. In: ICALP. Volume 115 of Lecture Notes in Computer Science. 1981 pp. 73–85. URL https://doi.org/10.1007/3-540-10843-2_7.

[2] Mendelzon AO. Database States and Their Tableaux. In: XP2 Workshop on Relational Database Theory. 1981. doi:DOI: 10.1145/329.318579.

[3] Imielinski T, Jr WL. Incomplete Information in Relational Databases. *J. ACM*, 1984;**31**(4):761–791. doi:10.1145/1634.1886.

[4] Aho AV, Beeri C, Ullman JD. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 1979;**4**(3):297–314. doi:http://doi.acm.org/10.1145/320083.320091.

[5] Fagin R. Horn clauses and database dependencies. *J. ACM*, 1982;**29**(4):952–985. doi:10.1145/322344.322347.

[6] Deutsch A, Nash A, Remmel JB. The chase revisited. In: PODS. 2008 pp. 149–158.

[7] Calì A, Gottlob G, Kifer M. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In: KR. 2008 pp. 70–80.

[8] Marnette B. Generalized schema-mappings: from termination to tractability. In: PODS. 2009 pp. 13–22. doi:10.1145/1559795.1559799.

[9] Grahne G, Onet A. Anatomy of the chase. *CoRR*, 2013;**abs/1303.6682**. URL http://arxiv.org/abs/1303.6682.

[10] Gogacz T, Marcinkowski J. All-Instances Termination of Chase is Undecidable. In: Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II. 2014 pp. 293–304. URL https://doi.org/10.1007/978-3-662-43951-7_25.

[11] Hernich A, Kupke C, Lukasiewicz T, Gottlob G. Well-founded semantics for extended datalog and ontological reasoning. In: PODS. 2013 pp. 225–236. doi:10.1145/2463664.2465229.

[12] Magka D, Krötzsch M, Horrocks I. Computing Stable Models for Nonmonotonic Existential Rules. In: IJCAI. 2013 pp. 1031–1038. ISBN:978-1-57735-633-2.

[13] Gogacz T, Marcinkowski J. Termination of oblivious chase is undecidable [Technical Report], 2014.

[14] Fagin R, Kolaitis PG, Miller RJ, Popa L. Data Exchange: Semantics and Query Answering. In: ICDT. 2003 pp. 207–224. URL https://doi.org/10.1007/3-540-36285-1_14.

[15] Deutsch A, Tannen V. Reformulation of XML Queries and Constraints. In: ICDT. 2003 pp. 225–241.

[16] Hernich A, Schweikardt N. CWA-solutions for data exchange settings with target dependencies. In: PODS. 2007 pp. 113–122. doi:10.1145/1265530.1265547.

[17] Meier M, Schmidt M, Lausen G. On Chase Termination Beyond Stratification. *PVLDB*, 2009;**2**(1):970–981. doi:10.14778/1687627.1687737.

[18] Spezzano F, Greco S. Chase Termination: A Constraints Rewriting Approach. *PVLDB*, 2010;**3**(1-2):93–104. doi:10.14778/1920841.1920858.

[19] Grahne G, Onet A. On Conditional Chase Termination. In: AMW vol. 11, 2011.

[20] Greco S, Spezzano F, Trubitsyna I. Stratification Criteria and Rewriting Techniques for Checking Chase Termination. *PVLDB*, 2011;**4**(11):1158–1168.

[21] Abiteboul S, Hull R, Vianu V. Foundations of Databases. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[22] Papadimitriou CH. Computational complexity. Addison-Wesley, 1994. ISBN: 978-0-201-53082-7.

[23] Rogers H. Theory of recursive functions and effective computability (Reprint from 1967). MIT Press, 1987. ISBN: 978-0-262-68052-3. URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=3182.

[24] Enderton HB. A mathematical introduction to logic. Academic Press, 1972. ISBN: 978-0-12-238450-9.

[25] Onet A. The Chase Procedure and its Applications in Data Exchange. In: Kolaitis PG, Lenzerini M, Schweikardt N (eds.), Data Exchange, Integration, and Streams, volume 5 of *Dagstuhl Follow-Ups*, pp. 1–37. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. ISBN: 978-3-939897-61-3, 2013. doi:10.4230/DFU.Vol5.10452.1. URL http://dx.doi.org/10.4230/DFU.Vol5.10452.1.

[26] Chandra AK, Merlin PM. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In: STOC. 1977 pp. 77–90. doi:10.1145/800105.803397.

[27] Rutenberg V. Complexity of generalized graph coloring. In: Proceedings of the 12th symposium on Mathematical foundations of computer science 1986. Springer-Verlag New York, Inc., New York, NY, USA. ISBN: 0387-16783-8, 1986 pp. 537–581. URL http://dl.acm.org/citation.cfm?id=22416.22478.

[28] Stockmeyer LJ. The Polynomial-Time Hierarchy. *Theor. Comput. Sci.*, 1976;**3**(1):1–22. URL https://doi.org/10.1016/0304-3975(76)90061-X.

[29] Fagin R, Kolaitis PG, Popa L. Data exchange: getting to the core. In: PODS. 2003 pp. 90–101.

[30] Gottlob G, Nash A. Data exchange: computing cores in polynomial time. In: PODS. 2006 pp. 40–49. doi:10.1145/1142351.1142358.

[31] Hernich A. Computing universal models under guarded TGDs. In: ICDT. 2012 pp. 222–235. ISBN: 978-1-4503-0791-8. doi:10.1145/2274576.2274600.

[32] Book RV, Otto F. String-rewriting systems. Texts and monographs in computer science. Springer, 1993. ISBN: 978-3-540-97965-4.

[33] Meier M, Schmidt M, Lausen G. On Chase Termination Beyond Stratification [Technical Report and Erratum]. Website, 2009. URL http://arxiv.org/abs/0906.4228.

A. Termination classes

This appendix includes the full definitions of the dependency classes considered in Section 6.

A.1. Dependencies with stratified-witness (SW)

Definition A.1. *[14, 7]* For a database schema $\mathbf{R}$, define a *position* in $\mathbf{R}$ to be a pair (R, k), where R is a relation symbol from $\mathbf{R}$ and $1 \leq k \leq arity(R)$.

Definition A.2. *[15]* Let Σ be a set of tgds over schema $\mathbf{R}$. The *chase flow graph* associated with Σ is a directed edge-labeled graph $G_\Sigma^F = (V, E)$, such that each vertex in V represents a position in $\mathbf{R}$ and $((R,i),(S,j)) \in E$ if there exists a tgd $\xi \in \Sigma$ of the form $\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \beta(\bar{x}, \bar{z})$, a variable $u \in \bar{x} \cup \bar{y}$ occurring in position (R, i) in α and a variable $v \in \bar{x} \cup \bar{z}$ that occurs in position (S, j) in β. In case $v \in \bar{z}$, the edge is labeled as *existential*, otherwise the label is considered *universal*.

Definition A.3. *[15]* A set Σ of tgds has *stratified-witness* if G_Σ^F has no cycles through an existential edge. The class of all sets of dependencies with stratified-witness is denoted SW.

As an example consider a schema with two binary relations R and S. The positions in this schema are $\{(R,1),(R,2),(S,1),(S,2)\}$. Let Σ_1 contain the following tgd:

$$\xi_{11} \quad : \quad S(x,y) \rightarrow \exists z\ R(x,z)$$

and let Σ_2 be:

$$\begin{aligned} \xi_{21} \quad &: \quad S(x,y) \rightarrow \exists z\ R(x,z) \\ \xi_{22} \quad &: \quad R(x,y) \rightarrow S(x,x) \end{aligned}$$

It is easy to observe that $\Sigma_1 \in \mathsf{SW}$. On the other hand, $\Sigma_2 \notin \mathsf{SW}$, since the flow graph of Σ_2 has the cycle given by the existential edges $((S,2),(R,2))$ and universal edge $((R,2),(S,2))$. The following theorem by [15] guarantees standard chase termination for all sets of dependencies in SW.

Theorem A.4. *[15]* $\mathsf{SW} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

A.2. Rich acyclicity (RA)

Definition A.5. *[16]* Let Σ be a set of tgds over schema $\mathbf{R}$. The *extended-dependency graph* associated with Σ is a directed edge-labeled graph $G_\Sigma^E = (V, E)$, such that each vertex represents a position in $\mathbf{R}$ and $((R,i),(S,j)) \in E$, if there exists a tgd $\xi \in \Sigma$ of the form $\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \beta(\bar{x}, \bar{z})$, and if one of the following holds:

1. $x \in \bar{x}$ and x occurs in α on position (R, i) and in β on position (S, j). In this case the edge is labeled as universal;
2. $x \in \bar{x} \cup \bar{y}$ and x occurs in α on position (R, i), and a variable $z \in \bar{z}$ occurs in β on position (S, j). In this case the edge is labeled as existential.

Definition A.6. [16] A set of tgds Σ is said to be *richly acyclic* G_Σ^E does not contain a cycle going through an existential edge. The class of all richly acyclic tgd sets is denoted RA.

As an example consider database schema $\mathbf{R} = \{S, R\}$, with $arity(S) = 1$ and $arity(R) = 2$. The set $\{(S,1),(R,1),(R,2)\}$ represents all positions in $\mathbf{R}$. Let Σ_1 contain the following dependency:

$$\xi_1 \quad = \quad S(x) \rightarrow \exists y\, R(x,y)$$

and let Σ_2 contain:

$$\xi_2 \quad = \quad R(x,y) \rightarrow \exists z\, R(x,z).$$

It is easy to see that $\Sigma_1 \in \mathsf{RA}$ as there are only outgoing edges from $(S,1)$ in the extended dependency graph. On the other hand, $\Sigma_2 \notin \mathsf{RA}$ because there is a an existential self-loop on node $(R,2)$ in the extended dependency graph. Note that the problem of testing if $\Sigma \in \mathsf{RA}$ is polynomial in size of Σ.

Theorem A.7. *[19]* $\mathsf{RA} \subset \mathsf{CT}^{\mathsf{obl}}_{\forall\forall}$

A.3. Weak acyclicity (WA)

Definition A.8. *[14]* Let Σ be a set of tgds over schema $\mathbf{R}$. The *dependency graph* associated with Σ is a directed edge-labeled graph $G_\Sigma = (V, E)$, such that the set of vertices V represents the positions in $\mathbf{R}$. There is an edge $((R,i),(S,j)) \in E$, if there exists a dependency $\xi \in \Sigma$ of the form $\alpha(\bar{x},\bar{y}) \rightarrow \exists \bar{z}\ \beta(\bar{x},\bar{z})$, and a variable $x \in \bar{x}$ such that x occurs in position (R,i) in α and one of the following holds:

1. x occurs in β in position (S,j). In this case the edge is labeled as universal;
2. there exists variable $z \in \bar{z}$ which occurs in position (S,j) in β. In this case the edge is labeled as existential.

Definition A.9. *[14]* A set of tgds Σ is said to be *weakly acyclic* if G_Σ does not have any cycle going through an existential edge. The class of all weakly acyclic sets of tgds is denoted WA.

Note that the problem of testing if $\Sigma \in \mathsf{WA}$ is polynomial in size of Σ.

As an example consider Σ_1 containing the same dependencies as in the example used for the RA, Σ_1:

$$\xi_1 \quad = \quad R(x,y) \rightarrow \exists z\, R(x,z)$$

and let Σ_2 containing a slight variation of the dependency from Σ_1:

$$\xi_2 \quad = \quad R(x,y) \rightarrow \exists z\, R(y,z).$$

In this case $\Sigma_1 \in \mathsf{WA}$ as the dependency graph does not contain any cycles (note that compared with the extended dependency graph the existential self loop on node $(R,2)$ is not part of the dependency graph). On the other hand, Σ_2 is not weakly acyclic as it has an existential self loop on node $(R,2)$.

Theorem A.10. *[14]* $\mathsf{WA} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

A.4. Safe dependencies (SD)

Definition A.11. *[7]* The set of *affected positions* $\mathit{aff}(\Sigma)$ for a set of tgds Σ is defined as follows. For all positions (R, i) that occur in the head of some tgd $\xi \in \Sigma$, then

1. if an existential variable appears in position (R, i) in ξ, then $(R, i) \in \mathit{aff}(\Sigma)$;
2. if universally quantified variable x appears in position (R, i) in the head and x appears only in affected positions in the body, then $(R, i) \in \mathit{aff}(\Sigma)$.

Intuitively, the affected positions are those where new null values can occur during the chase process. For example, the set of affected positions associated with the set of dependencies $\Sigma = \{R(x, y, z), S(y) \rightarrow \exists w\ R(y, w, x)\}$ is $\mathit{aff}(\Sigma) = \{(R, 2)\}$.

Definition A.12. *[17]* The *propagation graph* for a set of tgds Σ is a directed edge labeled graph $G^P_\Sigma = (\mathit{aff}(\Sigma), E)$. An edge $((R, i), (S, j))$ belongs to E if there exists a dependency $\xi \in \Sigma$ of the form $\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \beta(\bar{x}, \bar{z})$, a variable x that occurs in α at position (R, i), such that x occurs only in affected positions in α and one of the following holds:

1. x appears in β at affected position (S, j). In this case the edge is labeled as universal;
2. there exists variable $z \in \bar{z}$ which occurs at position (S, j) in β. In this case the edge is labeled existential.

Definition A.13. *[17]* A set of tgds Σ is called *safe* if its propagation graph G^P_Σ does not have a cycle going through an existential edge. The class of all safe sets of tgds is denoted SD.

Consider for example $\Sigma = \{R(x, y, z), S(y) \rightarrow \exists w\ R(y, w, x)\}$. It is easy to see that $\Sigma \notin \mathsf{WA}$, but because the only affected position in Σ is $(R, 2)$ and there are no edges going in or out from this position in G^P_Σ, it follows that $\Sigma \in \mathsf{SD}$.

We note that the problem of testing if $\Sigma \in \mathsf{SD}$, for a given Σ, is polynomial in size of Σ.

Theorem A.14. *[17]* $\mathsf{SD} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

A.5. Super-weak acyclicity (sWA)

Let Σ be a set of tgds such that no two tgds share common variable names. First Skolemize Σ by replacing each existential variable y in $\xi \in \Sigma$ by a Skolem function $f_y(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the universal variables that occur both in the body and head of ξ. The Skolemized Σ can then be viewed as a logic program P_Σ.

Definition A.15. *[8]* A *place* for a logic program P_Σ is a pair (A, i), where A is an atom in P_Σ and $1 \leq i \leq \mathit{arity}(A)$.

Definition A.16. *[8]* Let $\xi \in \Sigma$ be a tgd, and y an existential variable in ξ. The set of *output places* for y in ξ, denoted $\mathit{Out}(\xi, y)$, is the set of places in the head of $P_{\{\xi\}}$ that contains the Skolem term $f_y(\ldots)$.

Definition A.17. *[8]* Given a tgd $\xi \in \Sigma$, and x a universal variable in ξ. The set of *input places* for x in ξ, denoted $\mathit{In}(\xi, x)$, is the set of places in the body of $P_{\{\xi\}}$ where x occurs.

In the following definition a *substitution* is a function from variables to variables and constants. Substitutions are extended to atoms containing function terms in the natural way.

Definition A.18. *[8]* Places (A, i) and (B, j) are unifiable, denoted $(A, i) \sim (B, j)$, if $i = j$ and there exists substitutions θ and θ' such that $\theta(A) = \theta'(B)$.

Given two sets of places Q and Q', the relationship $Q \sqsubseteq Q'$ means that for all $q \in Q$ there exists $q' \in Q'$, such that $q \sim q'$.

Let S be a set of atoms and x a variable. Then $\Gamma_x(S)$ denotes the set of all places where x occurs in some atom in S.

Let Σ be a set of tgds and Q a set of places. Then $\mathit{Move}(\Sigma, Q)$ denotes the smallest set of places such that $Q \sqsubseteq \mathit{Move}(\Sigma, Q)$, and for every rule $\xi \in P_\Sigma$ of the form $\alpha \to \beta$, and for every variable x, if $\Gamma_x(\alpha) \sqsubseteq \mathit{Move}(\Sigma, Q)$, then $\Gamma_x(\beta) \sqsubseteq \mathit{Move}(\Sigma, Q)$.

Definition A.19. *[8]* Let Σ be a set of tgds and let $\xi, \xi' \in \Sigma$. Then ξ is said to trigger ξ', denoted $\xi \sim_\Sigma \xi'$, if there exists an existential variable y in ξ, and a universal variable x that appears both in the head and body of ξ', such that $\mathit{In}(\xi', x) \sqsubseteq \mathit{Move}(\Sigma, \mathit{Out}(\xi, y))$. A set of constraints Σ is said to be *super-weakly acyclic* iff the trigger relation $\sim_\Sigma$ is acyclic. The set of all super-weakly acyclic sets of tgds is denoted sWA.

Theorem A.20. *[8]* $\mathsf{sWA} \subset \mathsf{CT}^{\mathsf{sobl}}_{\forall\forall}$.

[8] also shows that testing if a set of tgd Σ is super-weakly acyclic is polynomial.

A.6. Stratification (Str, CStr)

Let ξ be a tgd $\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z})$, and $\bar{a} = \bar{a}_1\bar{a}_2$ a sequence cf constants of same length as $\bar{x}\bar{y}$. Then $\xi(\bar{a})$ denotes the tgd $\alpha(\bar{a}_1, \bar{a}_2) \to \exists \bar{z}\ \beta(\bar{a}_1, \bar{z})$.

Definition A.21. *[6]* Let ξ_1 and ξ_2 be tgds. We write $\xi_1 \prec \xi_2$, if there are instances I and J, and a sequence $\bar{a}$ with values from $dom(J)$, such that:

1. $I \models \xi_2(\bar{a})$, and
2. there exists an active trigger (ξ_1, h), such that $I \xrightarrow{(\xi_1,h)} J$, and
3. $J \not\models \xi_2(\bar{a})$.

As an example consider $\Sigma = \{\xi_1, \xi_2\}$, where:

$$\begin{aligned} \xi_1 &= R(x, y) \to S(x), \text{ and} \\ \xi_2 &= S(x) \to R(x, x). \end{aligned}$$

With the instance $I = \{R(a,b)\}$ and the sequence $\bar{a} = (a)$ we have that $I \models \xi_2(a)$; and for the homomorphism $h = \{x/a, y/b\}$ we have $I \xrightarrow{(\xi_1,h)} J$, where $J = \{R(a,b), S(a)\}$. Because $J \not\models \xi_2(a)$, it follows that $\xi_1 \prec \xi_2$. On the other hand, $\xi_2 \not\prec \xi_1$ because for any instance I and sequence of constants $\bar{b}$ such that $I \models \xi_1(\bar{b})$ and $I \xrightarrow{(\xi_2,h)} J$, for some active trigger (ξ_2, h), it follows that $J \models \xi_2(\bar{b})$.

Given a set of tgds Σ, the *chase graph* associated with Σ is a directed graph $G_\Sigma^C = (V, E)$, where $V = \Sigma$, and $(\xi_1, \xi_2) \in E$ iff $\xi_1 \prec \xi_2$.

Definition A.22. *[6]* A set of tgds Σ is said to be *stratified* if the set of dependencies in every simple cycle in G_Σ^C is weakly acyclic. The set of all stratified tgd sets is denoted Str.

Theorem A.23. [17] $\mathsf{Str} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\exists}$.

Next is the definition of the class CStr.

Definition A.24. *[17]* Let ξ_1 and ξ_2 be tgds. We write $\xi_1 \prec_\mathsf{c} \xi_2$, if there are instances I and J, and sequence $\bar{a}$ with values from $dom(J)$, such that:

1. $I \models \xi_2(\bar{a})$, and
2. there exists (not necessarily active) trigger (ξ_1, h), such that $I \xrightarrow{(\xi_1,h)} J$ in an oblivious chase step, and
3. $J \not\models \xi_2(\bar{a})$.

Given a set of tgds Σ, the *c-chase graph* associated with Σ is a directed graph $G_\Sigma^{CC} = (V, E)$, where $V = \Sigma$ and $(\xi_1, \xi_2) \in E$ iff $\xi_1 \prec_\mathsf{c} \xi_2$. A set of tgds Σ is said to be *c-stratified* if the set of dependencies in every simple cycle in G_Σ^{CC} is weakly acyclic. The set of all c-stratified tgd sets is denoted CStr.

Theorem A.25. *[17]* $\mathsf{CStr} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

A.7. Inductively restricted dependencies (IR) and the T-hierarchy

The definition below is taken from the erratum (http://arxiv.org/abs/0906.4228) and not from [17], where the presented condition, as mentioned in the erratum, does not guarantee the standard-chase termination for all sequences on all instances.

Let Σ be a set of tgds, I an instance and N a set of nulls. The set of all positions (R, i), such that there is a tuple in I that contains a null from N in position (R, i), is denoted *nullpos*(N, I).

Definition A.26. *[17]* Let Σ be a set of tgds and P a set of positions. Let $\xi_1, \xi_2 \in \Sigma$. Then $\xi_1 \prec_P \xi_2$ if there are instances I, J and sequence $\bar{a}$ of values from $dom(J)$, such that:

1. $I \models \xi_2(\bar{a})$, and

2. there exists (not necessarily active) trigger (ξ_1, h), such that $I \xrightarrow{(\xi_1,h)} J$ in an oblivious chase step, and
3. $J \not\models \xi_2(\bar{a})$, and
4. there is an null x in the head of $\xi_2(\bar{a})$, such that $\mathit{nullpos}(\{x\}, I) \subseteq P$.

As an example, consider Σ containing a single tgd $\xi = R(x,y) \rightarrow \exists z\ R(y,z)$. Note that $\Sigma \notin \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$. It is easy to see that with instances $I = \{R(a,b)\}$, $J = \{R(a,b), R(b,X)\}$, and sequence $\bar{a} = (b,x)$, conditions 1,2 and 3 from the previous definition are fulfilled. For the 4$^{\text{th}}$ condition, note that $\xi(\bar{a})$ represents the formula $R(a,x) \rightarrow \exists z R(x,z)$. Thus, x occurs in the head of $\xi(\bar{a})$. On the other hand, $\mathit{nullpos}(\{X\}, I) = \varnothing$ and instance I does not contain any nulls, hence for any set P, $\mathit{nullpos}(\{X\}, I) \subseteq P$. Consequently $\xi \prec_P \xi$, for any set of positions P.

For a tgd ξ, with $\mathit{vars}_\forall(\xi)$ we denote the set of all universally quantified variables in ξ and with $\mathit{vars}_\exists(\xi)$ the set of all existentially quantified variables in ξ.

Definition A.27. *[17]* Let P be a set of positions and ξ a tgd. Then $\mathit{affcl}(\xi, P)$ denotes the set of positions (R,i) from the head of ξ, such that one of the following holds:

1. for all $x \in vara_\forall(\xi)$, where x occurs in (R,i), the variable x occurs in the body of ξ only in positions from P, or
2. position (R,i) contains a variable $x \in vars_\exists(\xi)$.

For the previous example we have $\mathit{affcl}(\xi, P) = \{(R,1),(R,2)\}$, where $P = \{(R,2)\}$. Given a set of dependencies Σ, the set of all positions in Σ is written as $\mathit{positions}(\Sigma)$.

Definition A.28. *[17]* A *2-restriction system* is a pair (G_Σ, P), where G_Σ is a directed graph (Σ, E) and $P \subseteq \mathit{positions}(\Sigma)$ such that:

1. for all $(\xi_1, \xi_2) \in E$, $\mathit{affcl}(\xi_1, P) \cap \mathit{position}(\Sigma) \subseteq P$ and $\mathit{affcl}(\xi_2, P) \cap \mathit{positions}(\Sigma) \subseteq P$, and
2. for all $\xi_1 \prec_P \xi_2$, $(\xi_1, \xi_2) \in P$.

A 2-restriction system is *minimal* if it is obtained from $((\Sigma, \varnothing), \varnothing)$, that is the graph over Σ without any edge and the set of position empty, by a repeated application of constraints 1 and 2, from the previous definition, such that P is extended only by those positions that are required to satisfy condition 1. Let us denote by $part(\Sigma, 2)$ the set that contains the sets of all strongly connected components in a minimal 2-restriction system.

Returning to our example the minimal 2-restriction system is computed as follows. Consider pair $((\{\xi\}, \varnothing), \varnothing)$. Previously we showed that $\xi \prec_P \xi$, for any set of positions P, by particularization we have $\xi \prec_\varnothing \xi$. Thus, we add edge (ξ, ξ) to E. Using condition 1 from Definition A.28 we have $affcl(\xi, \varnothing) = \{(R,2)\}$. That is we add position $(R,2)$ to P. By repeating this process once again with $P = \{(R,2)\}$, we add to P the position $(R,1)$ too. Hence, the minimal 2-restriction system is $((\Sigma, \{(\xi,\xi)\}, \{(R,1),(R,2)\}\})$. The only connected component in this restriction system is $\{\xi\}$. [17] provide a simple algorithm that computes the set $part(\Sigma, 2)$.

Definition A.29. *[17]* A set Σ of tgds is called *inductively restricted* iff every $\Sigma' \in part(\Sigma, 2)$ is in SD. The set of all inductively restricted tgd sets is denoted by IR.

Theorem A.30. *[17]* $\mathsf{IR} \subset \mathsf{CT}^{\mathsf{std}}_{\forall\forall}$.

[17] observed that the inductive restriction criterion can be extended to form a hierarchy of classes that ensure the standard-chase termination on all branches for all instances. Intuitively, the lowest level of this hierarchy, noted $T[2]$, is the class of inductively restricted dependencies. Level $T[k]$, $k > 2$ is obtained by extending the binary relation $prec_P$ to a k-ary relation $prec_{k,P}$. Intuitively, $prec_{k,P}(\xi_1, \ldots, \xi_k)$ means that there exists a standard-chase sequence such that firing ξ_1 will also cause ξ_2 to fire. This in turn will cause ξ_3 to fire and so on until ξ_k. Based on this new relation, the set $part(\Sigma, k)$ is computed similarly to $part(\Sigma, 2)$. The algorithm that computes $part(\Sigma, k)$ was introduced by [17]. For all $k \leq 2$, it is shown that $T[k] \subset T[k+1]$.