Edinburgh Research Explorer

Chase Termination for Guarded Existential Rules

Citation for published version:
Calautti, M, Gottlob, G & Pieris, A 2015, Chase Termination for Guarded Existential Rules. in *Proceedings of the 9th Alberto Mendelzon International Workshop on Foundations of Data Management, Lima, Peru, May 6 - 8, 2015.*. CEUR Workshop Proceedings (CEUR-WS.org). <http://ceur-ws.org/Vol-1378/AMW_2015_paper_28.pdf>

Link:
[Link to publication record in Edinburgh Research Explorer](#)

Document Version:
Peer reviewed version

Published In:
Proceedings of the 9th Alberto Mendelzon International Workshop on Foundations of Data Management, Lima, Peru, May 6 - 8, 2015.

Chase Termination for Guarded Existential Rules

Marco Calautti[1], Georg Gottlob[2], and Andreas Pieris[3]

[1] DIMES, University of Calabria, Italy `calautti@dimes.unical.it`
[2] Department of Computer Science, University of Oxford, UK
`georg.gottlob@cs.ox.ac.uk`
[3] Institute of Information Systems, Vienna University of Technology, Austria
`pieris@dbai.tuwien.ac.at`

Abstract. The chase procedure is considered as one of the most fundamental algorithmic tools in database theory. It has been successfully applied to different database problems such as data exchange, and query answering and containment under constraints, to name a few. One of the central problems regarding the chase procedure is all-instance termination, that is, given a set of tuple-generating dependencies (TGDs) (a.k.a. existential rules), decide whether the chase under that set terminates, for every input database. It is well-known that this problem is undecidable, no matter which version of the chase we consider. The crucial question that comes up is whether existing restricted classes of TGDs, proposed in different contexts such as ontological reasoning, make the above problem decidable. In this work, we focus our attention on the oblivious and the semi-oblivious versions of the chase procedure, and we give a positive answer for classes of TGDs that are based on the notion of guardedness.

1 Introduction

The *chase procedure* (or simply chase) is considered as one of the most fundamental algorithmic tools in databases — it accepts as input a database D and a set Σ of constraints and, if it terminates (which is not guaranteed), its result is a finite instance D_Σ that enjoys two crucial properties:

1. D_Σ is a *model* of D and Σ, i.e., it contains D and satisfies the constraints of Σ; and
2. D_Σ is *universal*, i.e., it can be homomorphically embedded into every other model of D and Σ.

In other words, the chase is an algorithmic tool for computing universal models of D and Σ, which can be conceived as representatives of all the other models of D and Σ. This is precisely the reason for the ubiquity of the chase in database theory. Indeed, many key database problems can be solved by simply exhibiting a universal model.

A central class of constraints, which can be treated by the chase procedure and is of special interest for this work, are the well-known *tuple-generating dependencies (TGDs)* (a.k.a. *existential rules*) of the form $\forall \mathbf{X} \forall \mathbf{Y} (\varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} (\psi(\mathbf{Y}, \mathbf{Z})))$, where φ and ψ are conjunctions of atoms. Given a database D and a set Σ of TGDs, the chase adds new atoms to D (possibly involving nulls) until the final result satisfies Σ.

Example 1. Consider the database $D = \{person(Bob)\}$, and the TGD

$$\forall X (person(X) \rightarrow \exists Y \; hasFather(X, Y) \land person(Y)),$$

which asserts that each person has a father who is also a person. The database atom *triggers* the TGD, and the chase will add in D the atoms $hasFather(Bob, z_1)$ and $person(z_1)$ in order to satisfy it, where z_1 is a (labeled) null representing some unknown value. However, the new atom $person(z_1)$ triggers again the TGD, and the chase is forced to add the atoms $hasFather(z_1, z_2), person(z_2)$, where z_2 is a new null. The result of the chase is the instance

$$\{person(Bob), hasFather(Bob, z_1)\} \; \cup \; \bigcup_{i>0} \{person(z_i), hasFather(z_i, z_{i+1})\},$$

where $z_1, z_2, \ldots$ are nulls. ∎

As shown by the above example, the chase procedure may run forever, even for extremely simple databases and constraints. In the light of this fact, there has been a long line of research on identifying syntactic properties on TGDs such that, for every input database, the termination of the chase is guaranteed; see, e.g., [4, 8, 10, 12, 13] — this list is by no means exhaustive, and we refer to [9] for a comprehensive survey. With so much effort spent on identifying sufficient conditions for the termination of the chase procedure, the question that comes up is whether a sufficient condition that is also *necessary* exists. In other words, given a set Σ of TGDs, is it possible to determine whether, for every database D, the chase on D and Σ terminates? This interesting question has been recently addressed in [6], and unfortunately the answer is negative for all the versions of the chase that are usually used in database applications, namely the oblivious, semi-oblivious and restricted chase. In fact, the problem remains undecidable even if the database is known. This has been established in [4] for the restricted chase, and it was observed in [12] that the same proof shows undecidability also for the oblivious and the semi-oblivious chase.

Although the chase termination problem is undecidable in general, the proof given in [6] does not show the undecidability of the problem for TGDs that enjoy some structural conditions, which in turn guarantee favorable model-theoretic properties. Such a key condition is *guardedness*, a well-accepted paradigm that gives rise to robust rule-based languages that capture important databases constraints and lightweight description logics. A TGD is guarded if it has an atom in the left-hand side that contains (or guards) all the universally quantified variables [2]. Guardedness guarantees the tree-likeness of the underlying models, and thus the decidability of central database problems. The question that comes up is whether guardedness has the same positive impact on chase termination.

We focus on the (semi-)oblivious versions of the chase, and we show that the problem of deciding the termination of the chase for guarded TGDs is decidable, and we establish precise complexity results. Surprisingly, the present work is to our knowledge the first one that establishes positive results for the (semi-)oblivious chase termination problem. For more details, we refer the reader to [1].

2 The Chase Termination Problem

The *TGD chase procedure* (or simply *chase*) takes as input an instance I and a set Σ of TGDs, and constructs a universal model of I and Σ. The chase works on I by applying the so-called *trigger* for a set of TGDs on I. The trigger for a set Σ of TGDs on an instance I is a pair (σ, h), where $\sigma = \varphi \to \psi \in \Sigma$ and h is a homomorphism that maps φ to I. An *application* of (σ, h) to I returns $J = I \cup h'(\psi)$, where $h' \supseteq h$ maps each existentially quantified variable in ψ to a new null value. Such a trigger application is written $I\langle\sigma, h\rangle J$. The choice of the type of the next trigger to be applied is crucial since it gives rise to different versions of the chase procedure. In this work, we focus our attention on the *oblivious* [2] and *semi-oblivious* [7, 12] chase.

A finite sequence $I_0, I_1, \dots, I_n$, where $n \geqslant 0$, is said to be a *terminating oblivious chase sequence* of I_0 w.r.t. a set Σ of TGDs if: (i) for each $0 \leqslant i < n$, there exists a trigger (σ, h) for Σ on I_i such that $I_i\langle\sigma, h\rangle I_{i+1}$; (ii) for each $0 \leqslant i < j < n$, assuming that $I_i\langle\sigma_i, h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j, h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j = \sigma$ implies $h_i \neq h_j$, i.e., h_i and h_j are different homomorphisms; and (iii) there is no trigger (σ, h) for Σ on I_n such that $(\sigma, h) \notin \{(\sigma_i, h_i)\}_{0 \leqslant i \leqslant n-1}$. In this case, the result of the chase is the (finite) instance I_n. An infinite sequence $I_0, I_1, \dots$ of instances is said to be a *non-terminating oblivious chase sequence* of I_0 w.r.t. Σ if: (i) for each $i \geqslant 0$, there exists a trigger (σ, h) for Σ on I_i such that $I_i\langle\sigma, h\rangle I_{i+1}$; (ii) for each $i, j > 0$ such that $i \neq j$, assuming that $I_i\langle\sigma_i, h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j, h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j = \sigma$ implies $h_i \neq h_j$; and (iii) for each $i \geqslant 0$, and for every trigger (σ, h) for Σ on I_i, there exists $j \geqslant i$ such that $I_j\langle\sigma, h\rangle I_{j+1}$; this is known as the fairness condition, and guarantees that all the triggers eventually will be applied. The result of the chase is defined as the infinite instance $\cup_{i \geqslant 0} I_i$.

The semi-oblivious chase is a refined version of the oblivious chase, which avoids the application of some superfluous triggers. Roughly speaking, given a TGD σ of the form $\varphi \to \psi$, for the semi-oblivious chase, two homomorphisms h and g that agree on the universally quantified variables of σ occurring in ψ are indistinguishable.

Henceforth, we write o-chase and so-chase for oblivious and semi-oblivious chase, respectively. A $\star$-chase sequence, where $\star \in \{o, so\}$, may be infinite.

Example 2. Let $D = \{p(a, b)\}$, and $\Sigma = \{\forall X \forall Y (p(X, Y) \to \exists Z (p(Y, Z)))\}$. There exists only one $\star$-chase sequence of D w.r.t. Σ, where $\star \in \{o, so\}$, which is non-terminating, i.e., $I_0, I_1, \dots$ with

$$I_0 = \{p(a, b)\} \quad I_1 = \{p(a, b), p(b, z_1)\} \quad I_i = I_{i-1} \cup \{p(z_{i-1}, z_i)\}, \text{ for } i \geqslant 2,$$

where $z_1, z_2, \dots$ are nulls of $\mathbf{N}$. ■

For a set of TGDs, a key question is whether all or some $\star$-chase sequences are terminating on *all* databases. Before formalizing the above decision problems, let us recall the following key classes of TGDs:

$$\mathsf{CT}^\star_\forall = \{\Sigma \mid \forall D, \text{ all } \star\text{-chase sequences of } D \text{ w.r.t. } \Sigma \text{ are terminating}\}$$
$$\mathsf{CT}^\star_\exists = \{\Sigma \mid \forall D, \text{ there exists a terminating } \star\text{-chase sequence of } D \text{ w.r.t. } \Sigma\}.$$

The decision problems tackled in this work are as follows: for $q \in \{\forall, \exists\}$:

q-**Sequence $\star$-Chase Termination:**
    Instance: A set Σ of TGDs.
    Question: Does $\Sigma \in \mathsf{CT}_q^\star$?

We recall that $\mathsf{CT}_\forall^\mathsf{o} = \mathsf{CT}_\exists^\mathsf{o} \subset \mathsf{CT}_\forall^\mathsf{so} = \mathsf{CT}_\exists^\mathsf{so}$ [7]. This implies that the preceding decision problems coincide for the (semi-)oblivious chase. Henceforth, we refer to the $\star$-chase termination problem, and we write $\mathsf{CT}^\star$ for $\mathsf{CT}_\forall^\star$ and $\mathsf{CT}_\exists^\star$, where $\star \in \{\mathsf{o}, \mathsf{so}\}$.

3   The Complexity of Chase Termination

We focus on the class of guarded TGDs [2], and two key subclasses of it, namely simple linear and linear TGDs [3], and we investigate the complexity of the (semi-)oblivious chase termination problem. Recall that linear TGDs are TGDs with just one atom in the body, while simple linear TGDs forbid the repetition of variables in the body. Notice that, despite their simplicity, simple linear TGDs are powerful enough for capturing prominent database dependencies, and in particular inclusion dependencies, as well as key description logics such as DL-Lite. In the sequel, we denote by G the class of guarded TGDs, which is defined as the family of all possible sets of guarded TGDs. Analogously, we denote by SL and L the classes of simple linear and linear TGDs, respectively; clearly, $\mathsf{SL} \subset \mathsf{L} \subset \mathsf{G}$. Let us first consider the less expressive classes.

3.1   Linearity

By exploiting syntactic conditions that ensure the termination of each (semi-)oblivious chase sequence on all databases, we syntactically characterize the classes $(\mathsf{CT}^\star \cap \mathsf{SL})$ and $(\mathsf{CT}^\star \cap \mathsf{L})$, where $\star \in \{\mathsf{o}, \mathsf{so}\}$. We rely on weak-acyclicity [5] and rich-acyclicity [11]. Both weak- and rich-acyclicity are defined by posing an acyclicity condition on a graph, which encodes how terms are propagated among the positions of the underlying schema during the chase. In fact, weak-acyclicity forbids the existence of dangerous cycles (which involve the generation of new null values) in the dependency graph [5], while rich-acyclicity pose the same condition on the so-called extended dependency graph [11]. Let WA and RA be the classes of weakly- and richly-acyclic TGDs, respectively; notice that $\mathsf{RA} \subset \mathsf{WA}$. For simple linear TGDs we show that:

Theorem 1. $(\mathsf{CT}^\mathsf{o} \cap \mathsf{SL}) = (\mathsf{RA} \cap \mathsf{SL})$ *and* $(\mathsf{CT}^\mathsf{so} \cap \mathsf{SL}) = (\mathsf{WA} \cap \mathsf{SL})$.

In simple words, the above theorem states that, given a set $\Sigma \in \mathsf{SL}$: $\Sigma \in \mathsf{CT}^\mathsf{o}$ iff Σ is richly-acyclic, and $\Sigma \in \mathsf{CT}^\mathsf{so}$ iff Σ is weakly-acyclic. This result is established by showing that a dangerous cycle in the extended dependency graph (resp., dependency graph) necessarily gives rise to a non-terminating o-chase (resp., so-chase) sequence.

Let us now focus on (non-simple) linear TGDs. It is possible to show, by exhibiting a counterexample, that a dangerous cycle does not necessarily correspond to an infinite chase derivation. Thus, rich- and weak-acyclicity are not powerful enough for syntactically characterize the fragment of linear TGDs that guarantees the termination of the oblivious and semi-oblivious chase, respectively. Interestingly, it is possible to extend rich- and weak-acyclicity, focussing on linear TGDs, in such a way that the

above key property holds. The obtained formalisms are dubbed *critical-rich-acyclicity* and *critical-weak-acyclicity*, and the corresponding classes are denoted as LCriticalRA and LCriticalWA, respectively. We show that:

Theorem 2. $(CT^o \cap L) = LCriticalRA$ *and* $(CT^{so} \cap L) = LCriticalWA$.

The above syntactic characterizations, apart from being interesting in their own right, allow us to obtain optimal upper bounds for the $\star$-chase termination problem for $(S)L$ — we simply need to analyze the complexity of deciding whether a set of (simple) linear TGDs enjoys the above acyclicity-based conditions, which can be formulated as a reachability problem on a graph. In particular, we obtain the following results:

Theorem 3. *Consider a set Σ of TGDs. The problem of deciding whether $\Sigma \in CT^\star$, where $\star \in \{o, so\}$, is*

1. NL-*complete, even for unary and binary predicates, if $\Sigma \in SL$; and*
2. PSPACE-*complete, and* NL-*complete for predicates of bounded arity, if $\Sigma \in L$.*

For the hardness results, a generic technique, called the *looping operator*, is proposed, which allows us to obtain lower bounds for the chase termination problem in a uniform way. In fact, the goal of the looping operator is to provide a generic reduction from propositional atom entailment to the complement of chase termination.

3.2 Guardedness

We proceed to investigate the (semi-)oblivious chase termination problem for guarded TGDs. Although there is no way (at least no obvious one) to syntactically characterize the classes $(CT^\star \cap G)$, where $\star \in \{o, so\}$, via rich- and weak-acyclicity, as we did for (simple) linear TGDs, it is possible to show that the problem of recognizing the above classes is decidable. For technical reasons, we focus on *standard databases*, that is, databases that have two constants, let say 0 and 1, that are available via the unary predicates $0(\cdot)$ and $1(\cdot)$, respectively. In particular, we show the following:

Theorem 4. *Consider a set $\Sigma \in G$. The problem of deciding whether $\Sigma \in CT^\star$, where $\star \in \{o, so\}$, focussing on standard databases, is 2*EXPTIME-*complete, and* EXPTIME-*complete for predicates of bounded arity.*

The upper bounds are obtained by exhibiting an alternating algorithm that runs in exponential space, in general, and in polynomial space in case of predicates of bounded arity. The lower bounds are obtained by reductions from the acceptance problem of alternating exponential (resp., polynomial) space clocked Turing machines, i.e., Turing machines equipped with a counter. These reductions are obtained by modifying significantly existing reductions for the problem of propositional atom entailment under guarded TGDs, and then exploiting the looping operator mentioned above. The fact that the database is standard, is crucial for establishing the above lower bounds; the upper bounds hold even for non-standard databases.

4 Future Work

Our next step is to perform similar analysis focussing on the restricted version of the chase. We already have some preliminary positive results. In particular, if we focus on single-head linear TGDs, where each predicate appears in the head of at most one TGD, then we can syntactically characterize, via a careful extension of weak-acyclicity, the fragment that guarantees the termination of the restricted chase, and obtain a polynomial time upper bound. We are currently working towards the full settlement of the problem.

References

1. Calautti, M., Gottlob, G., Pieris, A.: Chase termination for guarded existential rules. In: PODS (2015), to appear
2. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. 48, 115–174 (2013)
3. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
4. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: PODS. pp. 149–158 (2008)
5. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
6. Gogacz, T., Marcinkowski, J.: All-instances termination of chase is undecidable. In: ICALP. pp. 293–304 (2014)
7. Grahne, G., Onet, A.: Anatomy of the chase. CoRR abs/1303.6682 (2013), http://arxiv.org/abs/1303.6682
8. Grau, B.C., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity conditions and their application to query answering in description logics. In: KR (2012)
9. Greco, S., Molinaro, C., Spezzano, F.: Incomplete Data and Data Dependencies in Relational Databases. Morgan & Claypool Publishers (2012)
10. Greco, S., Spezzano, F., Trubitsyna, I.: Stratification criteria and rewriting techniques for checking chase termination. PVLDB 4(11), 1158–1168 (2011)
11. Hernich, A., Schweikardt, N.: CWA-solutions for data exchange settings with target dependencies. In: PODS. pp. 113–122 (2007)
12. Marnette, B.: Generalized schema-mappings: From termination to tractability. In: PODS. pp. 13–22 (2009)
13. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. PVLDB 2(1), 970–981 (2009)