Beyond Skolem Chase: A Study of Finite Chase under Standard Chase Variant

Arash Karimi[1], Heng Zhang[2], and Jia-Huai You[1]

[1]Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada,
[2]Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

Abstract. The chase procedure is an indispensable tool for several database applications, where its termination guarantees decidability of these tasks. Most previous studies have focused on the skolem chase variant and its termination. It is known that the standard chase variant is more powerful in termination analysis provided a database is given. But all-instance termination analysis presents a challenge, since the critical database and similar techniques do not work. The interest of this paper is on the following question: How to devise a framework in which various known decidable classes of finite skolem chase can all be properly extended by the technique of standard chase? To address this question, we develop a novel technique to characterize the activeness of all possible cycles of a certain length in the standard chase procedure, which leads to the formulation of a parameterized class called k-SAFE$(cycle)$, which guarantees all-instance, all-path termination, where the parameter takes a *cycle function* by which a concrete class of finite standard chase is instantiated. The approach applies to any class of finite skolem chase that is identified with a condition of acyclicity. More generally, we show how to extend the hierarchy of bounded rule sets under skolem chase without increasing the complexity of data and combined reasoning tasks.

Introduction

The advent of emerging applications of knowledge representation and ontological reasoning have been the motivation of recent studies on existential rule languages, known as *tuple-generating dependencies* (*TGDs* [1], also called *rules*) or Datalog$^{\pm}$ [2], which have been considered a powerful modeling language for applications in data exchange, data integration, ontological querying, and so on. A major advantage of this approach is that the formal semantics based on first-order logic facilitates reasoning in an application, where answering conjunctive queries over a database extended with a set of existential rules is a primary task, but unfortunately an undecidable one in general [3]. *Chase procedure* is a bottom-up algorithm that extends a given database by applying specified rules. If such a procedure terminates, given an input database D, a finite rule set Σ and a conjunctive query, we can answer the query against Σ and D by deciding whether the result of chase based on Σ and D entails the query.

Four variants of chase procedure have been considered in the literature, which are called *oblivious*, *semi-oblivious (skolem)*, *standard* (aka *restricted*) and *core* chase. With oblivious chase being a weaker version of skolem chase and core chase as a parallel, wrapping procedure of standard chase (which indeed captures all universal models),

the main interests have been on skolem [4] and standard chase [5]. In particular, given a rule set and a database, we know that whenever skolem chase terminates so does the standard chase, but the reverse does not hold in general. Thus, standard chase is more powerful in termination analysis.

In spite of existence of many notions of acyclicity in the literature, (cf. [6] for a comprehensive survey), there are natural examples from the real world ontologies that do not belong to any known class, but with terminating standard chase. However, finding any characterization to assure standard chase termination is a very challenging task, and in the last decade, to the best of our knowledge only one condition has been found [7], originally in the context of description logics. As shown in [8], the problem of checking all-instance termination of standard chase is not a recursively enumerable problem (i.e. in general, the set of all-instance terminating standard chase TGDs is not enumerable). The tools developed in this paper are the first of its kind to provide an effective enumeration algorithm to characterize all-instance terminating standard chase TGDs.

In this work, we provide a highly general theoretical framework to identify strict superclasses of all existing classes of finite chase that we are aware of.

The main contributions of this paper are as follows:

1. We introduce a novel characterization of derivations under standard chase. We show that, although there is no unique database for termination analysis under standard chase, there exist a collection of "critical databases" by which the non-termination behavior of a derivation sequence can be captured. This is shown in Theorem 1.
2. We define a hierarchy of decidable classes of finite standard chase, called k-SAFE(cycle), which, when given a definition of cycle, is instantiated to a concrete class of finite chase. This is achieved by our Theorem 3 based on which various acyclicity conditions under skolem chase can be generalized to introduce classes of finite chase beyond finite skolem chase.
3. We show that the entire hierarchy of δ-bounded rule sets under skolem chase [9] can be generalized by introducing δ-bounded sets under standard chase, and as shown in Theorem 8, this extension does not increase the reasoning complexity under skolem chase.

Preliminaries

We assume the following pairwise disjoint sets of symbols: Const a countably infinite set of constants, N a countably infinite set of (labeled) nulls, and V a countably infinite set of variables. A *schema* is a finite set $\mathcal{R}$ of relation (or predicate) symbols, where each $R \in \mathcal{R}$ is assigned a positive integer as its arity which is denoted by $arity(R)$. Let $\mathsf{Dom} = \mathsf{Const} \cup \mathsf{N}$. *Terms* are symbols in $\mathsf{Dom} \cup \mathsf{V}$. *Ground terms* are terms involving no variables. An atom is an expression of the form $R(\mathbf{t})$, where $\mathbf{t} \in (\mathsf{Const} \cup \mathsf{V})^{arity(R)}$ and R is a predicate symbol from $\mathcal{R}$. A *general instance* (or simply an *instance*) I is a set of atoms over the relational schema $\mathcal{R}$; $term(I)$ denotes the set of terms occurring in I. A *database* is a finite instance I where terms are constants from Const. Given two instances I and J over the same schema, a *homomorphism* $h : I \to J$ is a mapping on terms which is identity on constants and for every atom $R(\mathbf{t})$ of I we have that $R(h(\mathbf{t}))$ (which we may write alternatively by $h(R(\mathbf{t}))$) is an atom of J such that $h(I) \subseteq J$.

A *tuple-generating dependency* (also called a *rule*) is a first-order sentence σ of the form: $\phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{x}, \mathbf{z})$, where $\mathbf{x}$ and $\mathbf{y}$ are sets of universally quantified variables (with the quantifier omitted) and ϕ and ψ are conjunctions of atoms constructed from relation symbols from $\mathcal{R}$, constants from Const, and variables from $\mathbf{x} \cup \mathbf{y}$ and $\mathbf{x} \cup \mathbf{z}$ respectively. The formula ϕ (resp. ψ) is called the *body* of σ, denoted $body(\sigma)$ (resp. the *head* of σ, denoted $head(\sigma)$).

Given a rule σ, a *skolem function* f_z^σ is introduced for each variable $z \in \mathbf{z}$ where $arity(f_z^\sigma) = |\mathbf{x}|$. Terms constructed from skolem functions and constants are called *skolem terms*. In this paper, we will regard skolem terms as a special class of nulls. *Ground terms* in this context are constants from N or skolem terms. The *functional transformation* of σ, denoted $sk(\sigma)$, is the formula obtained from σ by replacing each occurrence of $z_i \in \mathbf{z}$ with $f_{z_i}^\sigma(\mathbf{x})$. The *skolemized version* of a rule set Σ is the union of $sk(\sigma)$ for all rules $\sigma \in \Sigma$ and is denoted $sk(\Sigma)$.

We use $var_u(\sigma)$ and $var_{ex}(\sigma)$ to refer to the sets of universal and existential variables appearing in σ, and $var(\sigma)$ refers to the set of all variables appearing in σ. We further define: a *path* (based on Σ) is a nonempty (finite or infinite) sequence of rules from Σ; a *cycle* $C = (\sigma_1, ..., \sigma_n)$ $(n \geq 1)$ is a finite path whose first and last elements coincide (i.e., $\sigma_1 = \sigma_n$); a *k-cycle* $(k > 0)$ is a cycle in which at least one rule has $k+1$ occurrences and all other rules have $k + 1$ or less occurrences. Given a path P, with $\mathsf{Rules}(P)$, we denote the set of distinct rules used in P.

We assume all rules are *standardized apart* so that no variable is shared by more than one rule; given an instance I, $term(I)$ denotes the set of all terms occurring in I; and for a set or a sequence Q, the cardinality $|Q|$ is defined as usual.

Skolem and Standard Chase Variants

Chase is a fixpoint construction that accepts as input a database D and a finite set Σ of rules and adds atoms to D.

This paper is concerned with the skolem and standard chase variants. Below, let D be a database and Σ a rule set over the same schema.

We define skolem chase based on a breadth-first fixpoint construction as follows: we let $\mathsf{chase}_{sk}^0(D, \Sigma) = D$ and, for all $i > 0$, let $\mathsf{chase}_{sk}^i(D, \Sigma)$ be the union of $\mathsf{chase}_{sk}^{i-1}(D, \Sigma)$ and $h(head(sk(\sigma))$ for all rules $\sigma \in \Sigma$ and all homomorphisms h such that $h(body(\sigma)) \subseteq \mathsf{chase}_{sk}^{i-1}(D, \Sigma)$. Then, we let $\mathsf{chase}_{sk}(D, \Sigma)$ be the union of $\mathsf{chase}_{sk}^i(D, \Sigma)$ for all $i \geq 0$.

Due to order sensitive derivations, standard chase is defined over paths. In this paper, when we define a homomorphism $h : I \rightarrow J$, if I and J are clear from the context, we may define such a homomorphism as a mapping from variables to terms.

Definition 1. Let $P = (\sigma_1, \sigma_2, ...)$ be a path, where $\sigma_i \in \Sigma$ for all $i \geq 1$, and $H = (h_1, h_2, ...)$ homomorphisms such that $|P| = |H|$. A *standard chase path based on* Σ, denoted $\mathsf{chase\_p}_{std}(D, P, H)$, expresses a sequence of chase steps that satisfy the following conditions: for all $i \geq 1$,

(i) $h_i(body(\sigma_i)) \subseteq D_{i-1}$, with $h_i : var_u(\sigma_i) \rightarrow term(D_{i-1})$

(ii) $h_i^+(head(\sigma_i)) \not\subseteq D_{i-1}$ for all *extensions* h_i^+ of h_i,[1] such that $h_i^+ : var(\sigma_i) \rightarrow term(D_{i-1})$

where $D_0 = D$ and $D_i = D_{i-1} \cup h_i(head(sk(\sigma_i)))$. We also use $\mathsf{chase\_p}_{std}(D, P, H)$ to denote the union of D_i, $\forall i \geq 0$.

By abuse of notation, we may say "$\mathsf{chase\_p}_{std}(D, P, H)$ is a standard chase path" to mean that the sequence of chase steps on the path P using H, for the given database D, constitutes a standard chase path.

We say that a rule set Σ *has finite standard chase*, or *is terminating under standard chase*, if there are only finite standard chase paths based on Σ, for all databases.

The classes of rule sets whose chase terminates on all paths (all possible derivation sequences of chase steps) independent of given databases (thus all instances) is denoted by $CT^{\triangle}_{\forall\forall}$, where $\triangle \in \{sk, std\}$ (sk for skolem chase and std for standard chase).

In skolem chase, the applied rule σ and the related homomorphism h form a *trigger* (σ, h), w.r.t. the conclusion set generated so far. If such a skolem chase step is also a standard chase step, (σ, h) is called an *active trigger* [10].

A conjunctive query (CQ) Q is a formula of the form $Q(\mathbf{x}) := \exists \mathbf{y} \Phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables; $\Phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms with variables in $\mathbf{x} \cup \mathbf{y}$. A boolean conjunctive query (BCQ) is a CQ of the form $Q()$.

It is well-known that, for all boolean conjunctive queries (BCQs) q, $D \cup \Sigma \models q$ (under the semantics of first-order logic) if and only if $\mathsf{chase}_{sk}(D, \Sigma) \models q$ and if and only if $\bigcup_{\forall P,H} \mathsf{chase\_p}_{std}(D, P, H) \models q$ [11].

To illustrate the practical relevance of the standard chase, let us consider modeling a secure communication protocol where two different signal types can be transmitted: *type A* (or inter-zone) and *type B* (intra-zone). Let us consider a scenario where a transmitter from one zone requests to establish a secure communication with a receiver from another zone in this network. There are an unknown number of *trusted servers*. Before a successful communication between two users can occur, following a handshake protocol, the transmitter must send a type A signal to a trusted server in the same zone and receive an acknowledgment back. Then, that trusted server sends a type B signal to a trusted server in the receiver zone.

Below, we use existential rules to model the described communication (the modeling here does not include the actual process of transmitting signals).

Example 1. Consider the rule set Σ_1 below, where $S(x, y)$ denotes a request to send a type A signal from x to y and $D(x, y)$ q request to send a type B signal from x to y. Note that u and v are trusted servers in the zone where x and y are located, respectively.

$$D(x, y) \rightarrow \exists u\, S(x, u), S(u, x)$$
$$D(x, y), S(x, z), S(z, x) \rightarrow \exists v\, D(z, v)$$

Let $sk(\Sigma_1)$ be

$$\sigma_1 : D(x, y) \rightarrow S(x, s_1(x)), S(s_1(x), x)$$
$$\sigma_2 : D(x, y), S(x, z), S(z, x) \rightarrow D(z, s_2(z))$$

[1] An *extension* h_i^+ of h_i, denoted $h_i^+ \supseteq h_i$, assigns in addition to mapping h_i ground terms to existential variables.

where s_1 and s_2 are function symbols. With the database $D_0 = \{D(t, r)\}$, after applying σ_1 and σ_2, we have: $D_0 \cup \{S(t, s_1(t)), S(s_1(t), t), D(s_1(t), s_2(s_1(t)))\}$.
Clearly, the path (σ_1, σ_2) leads to a standard chase path, but $(\sigma_1, \sigma_2, \sigma_1)$ does not, since the trigger for applying the last rule in the path is not active - the head can be satisfied by already derived atoms $S(s_1(t), t)$ and $S(t, s_1(t))$.

Now let us consider another rule set $\Sigma_2 = \{r_1, r_2\}$, where,

$$r_1 : D(x, y) \to \exists u\, B(u), S(x, u), S(u, x)$$
$$r_2 : D(x, y), S(x, z), S(z, x), B(z) \to \exists v\, B(v), D(z, v)$$

With the same input database D_0 , we can find out that any non-empty prefix of $P = (r_1, r_2, r_1, r_2, r_1)$ leads to a standard chase path except P itself. Note that P is a 2-cycle.

Finite Standard Chase by Activeness

Since skolem chase is based on triggers and standard chase applies active triggers, standard chase provides a more powerful tool for termination analysis for a given database. However, this does not extend to the case of all-instance termination, as the critical database technique [12] for skolem chase does not apply to standard chase. Recall that a critical database is one that contains a ground atom for each relation symbol filled with exactly one fresh constant.

Example 2. With $\Sigma = \{E(x_1, x_2) \to \exists z E(x_2, z)\}$ and the critical database $D^c = \{E(*, *)\}$, where $*$ is a fresh constant, the skolem chase does not terminate w.r.t. D^c, which is a faithful simulation of the termination behavior of Σ under skolem chase. But the standard chase of Σ and D^c terminates in zero step, as no active triggers exist. However, the standard chase of Σ and database $\{E(a, b)\}$ does not terminate.

The above example is not at all a surprise, as the complexity of membership checking in the class of rule sets that have finite standard chase, $CT^{std}_{\forall\forall}$, is coRE-complete [8], which implies that in general, there exists no effectively computable (finite) set of databases which can simulate termination w.r.t. all input databases [2]. What we can do is to identify an RE subset of $CT^{std}_{\forall\forall}$, whose size can grow asymptotically and, at the same time, whose finite subsets can be checked for standard chase termination.

One natural consideration draws attention to the notion of cycles since, firstly, cycles are recursively enumerable with increasing lengths, and secondly, it is these cycles that may cause non-termination. Thus, checking cycles for safety yields an RE set, which can be extended by checking more nested cycles. However, a challenging question is *which databases to be checked against*. In the following, we tackle this question.

Capturing Activeness by Witness Databases

Given a path, our goal is to simulate derivations under standard chase with an arbitrary database, by derivations with a collection of databases. If this simulation captures all

[2] Even if there exists such finite databases, there is no computable bound on their size.

derivations in the path with any database, by running this simulation we can determine whether the given rule set is terminating for some finite set of pre-specified paths, for all databases.

We only need to consider the type of paths that potentially lead to cyclic applications of chase in the sense that the last rule of the path depends on the first, possibly through some intermediate ones in between.

Example 3. Consider the rule set $\Sigma = \{\sigma\}$ where

$$\sigma : \ T(x, y), P(x, y) \rightarrow \exists z T(y, z)$$

With $D_0 = \{T(a, b), P(a, b)\}$, we have: $\mathsf{chase}_{\Delta}(D_0, \Sigma) = D_0 \cup \{T(b, f(b))\}$, where f is a skolem function and $\Delta = \{std, sk\}$. It can be seen that after the first application of σ, there does not exist any more trigger and the skolem chase of Σ and D_0 terminates.

Note that σ in Example 3 depends on itself based on the classic notion of unification. So, with the aim of ruling out similar examples, we need to consider a dependency notion under which the cycle $P = (\sigma, \sigma)$ in the above example is not identified as a dangerous cycle. For this purpose, we consider the notion of rule dependencies as introduced in [13], which is based on *piece unification* [14].

Definition 2. (Piece unifier) Given a pair of rules (σ_1, σ_2), a *piece unifier* of $body(\sigma_2)$ and $head(\sigma_1)$ is a unifying substitution θ of $var(B) \cup var(H)$ where $B \subseteq body(\sigma_2)$ and $H \subseteq head(\sigma_1)$ which satisfies the following conditions:

(a) $\theta(B) = \theta(H)$, and
(b) variables in $var_{ex}(H)$ are not unified with variables that occur in both B and $body(\sigma_2) \setminus B$.

Condition (a) gives a sufficient condition for rule dependency, but it may be an overestimate, which is constrained by condition (b). It can be easily observed that in Example 3, condition (a) holds (for $B = \{T(x, y)\}$ and $H = \{T(y, z)\}$ where $\theta = \{x/y, y/z\}$) but condition (b) does not (since $var_{ex}(H) = \{z\}$ and z is unified with y which occurs in B and $body(\sigma) \setminus B = \{P(x, y)\}$). Therefore, based on Def. 2, no piece unifier of $body(\sigma)$ and $head(\sigma)$ exists.

Definition 3. Given two rules σ_1 and σ_2, we say that σ_2 *depends on* σ_1, if there is a piece unifier of $body(\sigma_2)$ with $head(\sigma_1)$, and we say that σ_2 *depends on* σ_1 *w.r.t.* θ, if θ is such a piece unifier.

Terminology: Given a vector $V = (v_1, ..., v_n)$ where $n \geq 2$, a *projection of* V *preserving end points*, denoted $V' = (v'_1, ..., v'_m)$, is a projection of V (as defined in usual way), with the additional requirement that the end points are preserved, i.e., $v'_1 = v_1$ and $v'_m = v_n$. By abuse of terminology, V' above will simply be called a *projection* of V.

Definition 4. Let Σ be a rule set, D a database, and P a path $(\sigma_1, ..., \sigma_n)$ with $n \geq 2$. A vector of homomorphisms $H = (h_1, ..., h_n)$ is called *chained* for P if there is a projection $H' = (h'_1, ..., h'_m)$ of H and the corresponding path projection $P' = (\sigma'_1, ..., \sigma'_m)$ of P such that for all $1 \leq i < m$, σ'_{i+1} depends on σ'_i w.r.t. h'_i.

With the aim of capturing condition (ii) of Def. 1 based on the notions that we introduced, in what follows, we define the notion of activeness.

Definition 5. (Activeness) Let Σ be a rule set and D a database. A path $P = (\sigma_1, ..., \sigma_n)$ is said to be *active* w.r.t. D, if there exists a chained vector of homomorphisms $H = (h_1, ..., h_n)$ for P such that $\mathsf{chase\_p}_{std}(D, P, H)$ is a standard chase path for D, P, H based on Σ. In this case, $H = (h_1, ..., h_n)$ is called a *witness* of the activeness of P w.r.t. D.

Intuitively, the definition says that a path P is active w.r.t. a database D if there is an active trigger for each rule application in the path, hence it is a standard chase path, and the last rule application depends on some earlier ones and eventually on the first on the path. In other words, if P is not active w.r.t. D we then know that P is either "terminating" w.r.t. D, or for each vector of homomorphisms $H = (h_1, ..., h_n)$ such that $\mathsf{chase\_p}_{std}(D, P, H)$ is a standard chase path, H is not chained for P; thus it does not pose as a "dangerous cycle" even in the case $\sigma_1 = \sigma_n$.

Let $H = (h_1, ..., h_n)$ be a vector of homomorphisms such that $\mathsf{chase\_p}_{std}(D, P, H)$ is a standard chase path. To determine if H is chained for P, $\mathcal{O}(n^2)$ checks are needed,[3] where each $h_i(body(\sigma_i))$ is checked against every $h_j(head(\sigma_j))$, for all $j < i$, to determine if σ_i depends on σ_j w.r.t. $h_i \cup h_j$.

We are now ready to address the issue of which databases to be checked against for termination analysis. First, let us define a mapping $e_i : \mathsf{V} \cup \mathsf{Const} \to \langle \mathsf{V}, i \rangle \cup \mathsf{Const}$, for each $i \in \mathbb{N}$, where constants in Const are mapped to themselves and each variable $v \in \mathsf{V}$ is mapped to $\langle v, i \rangle$.

Definition 6. Given a path $P = (\sigma_1, \sigma_2, ...)$, we define

$$D_P = \{e_i(body(\sigma_i)) : 1 \leq i < |P| + 1\}. \tag{1}$$

A pair $\langle x, i \rangle$ in D_P is intended to name a *fresh constant*, based on the derivation step in P in which the variable x occurs. In this way, all these fresh constants are distinct. Note that D_P is a set of atoms over these new constants (and the constants already appearing in rules) and is independent of any input database. Let us call these pairs *indexed constants* and use short hand v^i for $\langle v, i \rangle$.

Intuitively, atoms in D_P are intended to simulate those in a derivation sequence where body atoms in an applied rule are satisfied by atoms from a given database. Since in general we don't know which body atoms are satisfied by a given database in each step of a derivation sequence, we need to test activeness using each subset of D_P. This process is finite whenever P is.

Note that due to the structure of D_P, application of each rule to D_P may lead to a new trigger and therefore, without the notion of chained property, every path can be trivially active.

Example 4. Consider the rule set Σ of Example 3 and a path $P = (\sigma, \sigma)$. We have: $D_P = \{T(x^1, y^1), P(x^1, y^1), T(x^2, y^2), P(x^2, y^2)\}$. We see that P is *not* active w.r.t. D_P, as there does not exist any chained vector of homomorphisms $H = (h_1, h_2)$ for P.

[3] Note that the problem of piece unifiability is known to be NP-complete.

Theorem 1. *Let Σ be a rule set, and $P = (\sigma_1, ..., \sigma_n)$ a path. P is active w.r.t. some database if and only if P is active w.r.t. some database $D^* \subseteq D_P$.*

Definition 7. A k-cycle $P = (\sigma_1, ..., \sigma_n)$ is *safe* if for all databases D, P is not active w.r.t. D. A rule set Σ is said to be *k-safe* if all k-cycles of Σ are safe.

It then follows from Theorem 1, we have

Corollary 2. *Let Σ be a rule set. For any k-cycle $P = (\sigma_1, ..., \sigma_n)$, if for all subsets D^* of D_P, P is not active w.r.t. D^*, then P is safe.*

E.g., it can be verified that the rule set Σ_1 in Example 1 is k-safe for any $k \geq 1$.

We now introduce a hierarchy of classes of finite standard chase based on the notion of k-safety. The idea is to introduce a parameter of *cycle function* to generalize various notions of acyclicity in the literature.

Definition 8. Let Σ be a rule set and $\mathcal{C}$ the set of all finite cycles based on Σ. A *cycle function* is a mapping $cycle_\Sigma : \mathcal{C} \to \{T, F\}$.

Let $cycle$ be a binary function from rule sets and cycles of which $cycle_\Sigma$ is the projection function on its first parameter, i.e. $cycle(\Sigma, C) = cycle_\Sigma(C)$. By abuse of terminology, the function $cycle$, which in addition takes a rule set as a parameter, is also called a cycle function.

Definition 9. (k-SAFE(cycle) rule sets) A rule set Σ is said to be k-SAFE($cycle$), or belong to k-SAFE($cycle$) (under cycle function $cycle$), if for every k-cycle C which is mapped to T under $cycle_\Sigma$, C is safe.

In the following, we show how to define a cycle function from any arbitrary acyclicity condition of finite skolem chase such as weak acyclicity (WA) [11], super-weak acyclicity (SWA) [12], etc.

Definition 10. Let Π be an arbitrary condition of acyclicity of finite skolem chase. A cycle function Φ_Π is defined as follows: For each rule set Σ and each cycle C based on Σ if the condition for checking whether Π holds for rules in $\mathsf{Rules}(C)$ [4], then Φ_Π maps (Σ, C) to F; otherwise Φ_Π maps (Σ, C) to T.

Example 5. Considering the rule set Σ_1 from Example 1, and assuming $\Pi = \text{WA}$ in Def. 10, Φ_Π maps all cycles C such that $\mathsf{Rules}(C) \cap \{\sigma_1, \sigma_2\} \neq \{\sigma_1, \sigma_2\}$ to F and the rest of the cycles to T.

The following theorem is easy to show.

Theorem 3. *Let Π be a class of finite skolem chase that is defined by a condition of acyclicity and Φ_Π be the corresponding cycle function as defined above. Then, for all $k \geq 1$, (i) k-SAFE(Φ_Π) $\supset \Pi$, and (ii) k-SAFE(Φ_Π) $\subseteq CT^{std}_{\forall\forall}$.*

Example 6. It is not difficult to check that the rule set Σ_1 in Example 1 is 2-SAFE(Φ_{WA}), where Φ_{WA} is the corresponding cycle function based on the condition of acyclicity for WA. Note that Φ_{WA} maps 1-cycles of the form (σ_i, σ_i), where $i = 1, 2$, to F and all other 1-cycles to T [5]. Similarly, the rule set Σ_2 in the same example is 2-SAFE(Φ_{WA}).

[4] Recall that $\mathsf{Rules}(C)$ is the set of distinct rules in C.

[5] Note that e.g, for 1-cycles of the form (σ_2, σ_2) or $(\sigma_2, \sigma_1, \sigma_1, \sigma_2)$, there is no vector of homomorphisms satisfying the chained property.

Extension of Bounded Rule Sets

In [9], a family of existential rule languages with finite (skolem) chase based on the notion of δ-*boundedness* has been introduced and the data and combined complexities of reasoning with those languages for specific bound functions (k-*exponentially bounded functions*). Our aim in this section is to provide an effective method to extend bounded languages to introduce all-instance terminating standard chase classes. Before we state our results, let us introduce some notions.

A *bound function* is a function from positive integers to positive integers. A rule set Σ is called δ-*bounded under skolem chase* for some bound function δ, if for all databases D, $ht(\mathsf{chase}_{sk}(D, \Sigma)) \leq \delta(\|\Sigma\|)$, where $\|\Sigma\|$ is the number of symbols occurring in Σ, and $ht(A)$ is the maximum height (maximum nesting depth) of skolem terms that have at least one occurrence in A, and ∞ otherwise. Let us denote by δ-$\mathcal{B}^{sk}$ the class of δ-*bounded rule sets under skolem chase*.

Lemma 4. *For any given rule set Σ and bound function δ, there exists a cycle function $\mathcal{F}_{\delta,\Sigma} : \mathbf{C} \rightarrow \{T, F\}$, where $\mathbf{C}$ is set of all finite cycles from Σ with the following property: $\mathcal{F}_{\delta,\Sigma}$ maps cycles C from Σ of length greater than $\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}$ to F and the rest of cycles from Σ to T. Note that this property provides a legitimate cycle function based on Def. 8.*

We call the cycle function $\mathcal{F}_{\delta,\Sigma}$ constructed in the above lemma, δ-*cycle function w.r.t.* Σ. We are ready to define the standard chase counterpart of δ-bounded rule sets.

Definition 11. Given a bound function δ, a rule set Σ is called δ-*bounded under the standard chase variant* if, for all databases D, Σ is in k-$\mathsf{SAFE}(cycle)$, where $k = \|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}$ and $cycle$ is a δ-cycle function w.r.t. Σ.

Denote the set of all δ-bounded rule sets under the standard chase variant by δ-$\mathcal{B}^{std}$. The next result states the relationship between δ-$\mathcal{B}^{std}$ and δ-$\mathcal{B}^{sk}$.

Theorem 5. *For any bound function δ, δ-$\mathcal{B}^{std} \supset \delta$-$\mathcal{B}^{sk}$.*

Membership Complexity

We consider the membership problem in δ-$\mathcal{B}^{std}$ languages: given a rule set Σ and a bound function δ, whether Σ is δ-bounded under the standard chase variant.

Proposition 6. *Let δ be a bound function computable in $\mathsf{DTIME}(T(n))$[6] for some function $T(n)$. Then, it is in*

$$\mathsf{DTIME}(\mathsf{C}_\delta + \|\Sigma\|^{\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}} \times |\mathcal{P}(D_c)| \times \mathsf{C}_c \times \Pi_2^{\mathsf{P}}[7])$$

to check if a rule set Σ is δ-bounded under the standard chase variant, where $\mathsf{C}_\delta = T(log\|\Sigma\|)^{\mathcal{O}(1)}$, $|\mathcal{P}(D_c)| = 2^{b(\Sigma)\times\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}}$ in which $b(\Sigma)$ is the maximum number of body atoms for rules in Σ, and $\mathsf{C}_c = \|\Sigma\|^{2\mathcal{O}(\delta(\|\Sigma\|))} \times 2^{\|\Sigma\|^{\mathcal{O}(\delta(\|\Sigma\|))}}$.

[6] The set of all decision problems solvable in $T(n)$ using a deterministic Turing machine.

[7] The class of problems solvable in coNP using an NP oracle.

Now consider what we call *exponential tower functions*:

$$\mathsf{exp}_\kappa(n) = \begin{cases} n & \kappa = 0 \\ 2^{\mathsf{exp}_{\kappa-1}(n)} & \kappa > 0 \end{cases}$$

Then, we have the following corollary from Prop. 6:

Corollary 7. *Checking if a rule ontology is* exp_κ*-bounded under standard chase variant is in* $(\kappa + 2)$-EXPTIME.

The corollary implies that, for any exponential tower function δ, the extra computation for checking δ-boundedness under standard chase stays within the same complexity upper bound for δ-boundedness under skolem chase, as reported in [9].

Remark 1. For membership checking, switching from skolem to standard chase variant does not increase the complexity for exp_κ-bounded rule sets. However, it should be clear that for some syntactic classes of finite skolem chase which are known to belong to exp_0-bounded rule sets (e.g. WA, JA, SWA, etc.), the penalty for going beyond finite skolem chase is the higher cost of membership checking.

Data and Combined Complexity:

Since the size of the tree generated by standard chase has the same upper bound as that generated by skolem chase, it is not difficult to show that the complexity upper bound for checking $\Sigma \cup D \models q$ for exp_κ-bounded rule sets is the same for the standard and skolem chase variants. It then follows from [9] that we have

Theorem 8. *The problem of Boolean conjunctive answering for* exp_κ*-bounded rule ontologies under standard chase variant is* $(\kappa + 2)$-EXPTIME*-complete for combined complexity and* PTIME*-complete for data complexity.*

Remark 2. Based on Theorem 8, it can be observed that the entire hierarchy of δ-bounded rule sets under skolem chase introduced in [9] can be extended with no increase in combined and data complexity of reasoning. This holds even for individual syntactic classes of finite skolem chase (such as WA, JA, SWA, etc.).

Related Work and Discussion

Many sufficient conditions have been discovered for termination of skolem and oblivious chase based on different notions of acyclicity based on graphs derived from the given rule set. Typically, such a graph is constructed based on some notion of dependencies in the rule set, e.g., position dependencies and rule dependencies. These classes include *WA* (*weak acyclicity*) [11], *STR* (*stratification*) [15], *safety* [16], *SWA* (*super-weak acyclicity*) [4], *aGRD* (*acyclicity of graph of rule dependencies*) [17], and *JA* (*joint acyclicity*) [18].

To the best of our knowledge, [7] is the only work that have studied termination of the standard chase towards positive results. Their results are only in the context of *Horn-$\mathcal{SRIQ}$ description logics* and not applicable to existential rules. In [19] a notion

called *well of positivity* has been devised which is an extension of the critical instance technique and has been applied in undecidability results. For their purpose, they only need one extra constant in addition to the well of positivity to show undecidability of the standard chase for all instances on all paths. However, in general, there is no bound for the number of constants needed to witness non-termination of the standard chase.

Under reasonable assumptions, one can find syntactic conditions that guarantee termination of the standard chase for all instances and all paths. As an example, consider the following rule $\Sigma = \{\sigma\}$, where:

$$\sigma : E(x, y) \rightarrow \exists u_1 \dots u_k E(y, u_1), E(u_1, u_2), \dots, E(u_k, y)$$

for some $k \geq 1$. It can be observed that starting from any database D, $\mathsf{chase}_{std}(D, \Sigma)$ terminates in maximum $|I_E|+1$ steps, where I_E is the database D restricted to the tuple over predicate E. Intuitively, the reason that $\Sigma \in CT^{std}_{\forall\forall}$ is the observation that predicate E (for which $E[i]$, for some i, is the placeholder of (potentially) infinite terms), when seen as a graph in which it is interpreted as the edge relation, forms a cycle in $head(\sigma)$, where σ belongs to the following cycle $C = (\sigma, \sigma)$ that is mapped to T under any condition of (skolem) acyclicity.

In [20] it is shown that for (weakly-)guarded and (simple-)linear TGDs, the problem of deciding (non-)termination of oblivious/skolem chase is decidable. This characterization leads to finding upper bounds for the length of cycles that should be considered in order to decide (non-)termination w.r.t. the skolem chase. Theorem 1 provides tools to extend their results to all-instance, all-path standard chase variant. Note that they only consider single-head TGDs [8]. So, their results cannot be trivially extended to the standard variant of chase for (arbitrary) multi-head TGDs. Further results in this direction are outside of the focus of this paper and is the subject of the future work.

Conclusions

In this work we introduced a technique to characterize finite standard chase, which can be applied to extend any class of finite skolem chase identified by a condition of acyclicity. Then, we showed how to apply our techniques to extend δ-bounded rule sets. Our complexity analyses showed that this extension does not increase the complexities of membership checking, nor the complexity of combined and data reasoning tasks for δ-bounded rule sets under standard chase compared to skolem chase. However, comparing with some subclasses of the exp_0-bounded language, membership checking for finite standard chase incurs higher cost.

We will next investigate conditions for subclasses with a reduction of cost for membership testing. One idea is to find syntactic conditions under which triggers to a rule are necessarily active, and the other is on symmetric conditions under which triggers of certain kind cannot be active. We anticipate that position graphs could be useful in these analyses.

[8] The standard transformation of multi-head TGDs to single-head TGDs preserves oblivious/skolem chase termination, but does not preserve standard chase termination.

References

1. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. Journal of the ACM (JACM) **31**(4) (1984) 718–741
2. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: Proc. LICS-2010. (2010) 228–242
3. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP-1981. (1981) 73–85
4. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: Proc. PODS-2009. (2009) 13–22
5. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. In: Database TheoryICDT 2003. Springer (2003) 207–224
6. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. Journal of Artificial Intelligence Research (2013) 741–808
7. Carral, D., Feier, C., Hitzler, P.: A practical acyclicity notion for query answering over horn-$\mathcal{SRIQ}$ ontologies. In: International Semantic Web Conference, Springer (2016) 70–85
8. Grahne, G., Onet, A.: The data-exchange chase under the microscope. CoRR **abs/1407.2279** (2014)
9. Zhang, H., Zhang, Y., You, J.H.: Existential rule languages with finite chase: complexity and expressiveness. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press (2015) 1678–1684
10. Onet, A.: The chase procedure and its applications in data exchange. Dagstuhl Follow-Ups **5** (2013)
11. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science **336**(1) (2005) 89–124
12. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM (2009) 13–22
13. Baget, J.F.: Improving the forward chaining algorithm for conceptual graphs rules. KR **4** (2004) 407–414
14. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, Ê.: Extending decidable cases for rules with existential variables. In: IJCAI. Volume 9. (2009) 677–682
15. Deutsch, A., Nash, A., Remmel, J.: The chase revisited. In: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM (2008) 149–158
16. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. Proceedings of the VLDB Endowment **2**(1) (2009) 970–981
17. Baget, J.F., Mugnier, M.L., Thomazo, M.: Towards farsighted dependencies for existential rules. In: Web Reasoning and Rule Systems. Springer (2011) 30–45
18. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Twenty-Second International Joint Conference on Artificial Intelligence. (2011)
19. Gogacz, T., Marcinkowski, J.: All–instances termination of chase is undecidable. In: Automata, Languages, and Programming. Springer (2014) 293–304
20. Calautti, M., Gottlob, G., Pieris, A.: Chase termination for guarded existential rules. In: Proceedings of the 34th ACM Symposium on Principles of Database Systems, ACM (2015) 91–103