Non-Uniformly Terminating Chase: Size and Complexity

Marco Calautti
University of Trento
marco.calautti@unitn.it

Georg Gottlob
University of Oxford &
TU Wien
georg.gottlob@cs.ox.ac.uk

Andreas Pieris
University of Edinburgh &
University of Cyprus
apieris@inf.ed.ac.uk

ABSTRACT
The chase procedure, originally introduced for checking implication of database constraints, and later on used for computing data exchange solutions, has recently become a central algorithmic tool in rule-based ontological reasoning. In this context, a key problem is non-uniform chase termination: does the chase of a database w.r.t. a rule-based ontology terminate? And if this is the case, what is the size of the result of the chase? We focus on guarded tuple-generating dependencies (TGDs), which form a robust rule-based ontology language, and study the above central questions for the semi-oblivious version of the chase. One of our main findings is that non-uniform semi-oblivious chase termination for guarded TGDs is feasible in polynomial time w.r.t. the database, and the size of the result of the chase (whenever is finite) is linear w.r.t. the database. Towards our results concerning non-uniform chase termination, we show that basic techniques such as simplification and linearization, originally introduced in the context of ontological query answering, can be safely applied to the chase termination problem.

CCS CONCEPTS
• Information systems → Data management systems; • Theory of computation → Description logics.

KEYWORDS
semi-oblivious chase procedure; tuple-generating dependencies; guardedness; non-uniform termination

ACM Reference Format:
Marco Calautti, Georg Gottlob, and Andreas Pieris. 2022. Non-Uniformly Terminating Chase: Size and Complexity. In Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS ‘22), June 12–17, 2022, Philadelphia, PA, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3517804.3524146

1 INTRODUCTION

Nowadays we need to deal with data that is very large, heterogeneous, distributed in different sources, and incomplete. This makes the task of extracting information from such data by means of queries very complex. At the same time, we have very large amounts of knowledge about the application domain of the data in the form of ontologies. This gave rise to a research field, recently coined as knowledge-enriched data management [1], that lies at the intersection of data management and knowledge representation and reasoning. A major challenge for knowledge-enriched data management is to provide end users with flexible and integrated access to data by using the available knowledge about the underlying application domain. Ontology-based data access (OBDA) [23] has been proposed as a general paradigm for addressing the above challenge. The main algorithmic task underlying OBDA is querying knowledge-enriched data, that is, during the query answering process we also need to take into account the inferred knowledge. This problem is also known as ontological query answering.

Typically, the ontologies employed in data-intensive applications such as OBDA are modeled via description logics, in particular, members of the DL-Lite [11] and $\mathcal{EL}$ [3] families, mainly due to their good computational properties when it comes to ontological query answering. On the other hand, there is a consensus that rule-based ontologies, i.e., ontologies consisting of tuple-generating dependencies (TGDs) (a.k.a. existential rules), are also well-suited for data-intensive applications since they allow us to conveniently deal with higher-arity relations that appear in standard relational databases. In particular, linear and guarded TGDs strike a good balance between expressiveness and complexity that make them suitable ontology languages for data-intensive applications [10]. Interestingly, the main members of the DL-Lite family (modulo some easily handled features) are special cases of linear TGDs (in fact, simple linear TGDs, where variables are not repeated in rule-bodies), while the main members of the $\mathcal{EL}$ family are (up to a certain normal form) special cases of guarded TGDs.

A prominent tool for studying rule-based ontological query answering is the chase procedure (or simply chase). It takes as input a database $D$ and a rule-based ontology $\Sigma$, and, if it terminates, it computes a finite instance $D_\Sigma$ that is a universal model of $D$ and $\Sigma$, i.e., a model that can be homomorphically embedded into every other model of $D$ and $\Sigma$. This is why the chase is an important algorithmic tool for rule-based ontological query answering. And this is not only in theory. There are efficient implementations of the chase that allow us to solve ontological query answering by adopting a materialization-based approach. It has been recently observed that for RAM-based implementations the restricted (a.k.a. standard) version of the chase is the indicated tool [6, 20], while for RDBMS-based implementations the semi-oblivious chase is preferable [6]. For being able, though, to employ existing chase implementations, we need a guarantee that the chase terminates.

This brings us to the chase termination problem (which is clearly parameterized by the version of the chase) that comes in two different variants: uniform and non-uniform. The uniform variant takes as input a rule-based ontology $\Sigma$, and the question is whether the chase terminates for every database w.r.t. $\Sigma$. It is clear that whenever
uniform chase termination is guaranteed, we can solve ontological query answering via materialization no matter how the database looks like. But even if uniform chase termination is not guaranteed, we can still rely in some cases on materialization, depending on the input database. This reveals the relevance of the non-uniform variant of the chase termination problem, which takes as input a database $D$ and a rule-based ontology $\Sigma$, and asks if the chase of $D$ w.r.t. $\Sigma$ terminates. It is well-known that, no matter which version of the chase we consider, both uniform and non-uniform chase termination is undecidable when we consider arbitrary rule-based ontologies [12, 14]. On the other hand, once we focus on well-behaved classes of TGDs, we have several positive results concerning the uniform variant of the chase termination problem. The first such results were established for (simple) linear and guarded TGDs, and the semi-oblivious version of the chase [8]. There are also recent results for sticky TGDs (another well-behaved class that is inherently unguarded), and the semi-oblivious version of the chase [9]. The restricted chase has been recently studied with linear TGDs [16, 21], as well as with guarded and sticky TGDs [15].

Although the uniform variant of chase termination has been extensively studied in the literature, the non-uniform one has largely remained unexplored. In this work, we concentrate on the semi-oblivious chase and study the following questions: given a database $D$ and a (simple) linear or guarded rule-based ontology $\Sigma$:

1. What is the worst-case optimal size of the result of the chase of $D$ w.r.t. $\Sigma$ (whenever is finite)?
2. Can we decide whether the result of the chase of $D$ w.r.t. $\Sigma$ is finite, and if yes, what is the exact complexity?

Summary of Contributions. After illustrating the different nature of the non-uniformly terminating chase compared to the uniformly terminating one, we establish characterizations of non-uniform chase termination of the following form:

Main Characterizations. Given a database $D$, and a (simple) linear or guarded rule-based ontology $\Sigma$, the following are equivalent:

1. The result of the chase of $D$ w.r.t. $\Sigma$ is finite.
2. The size of the result of the chase of $D$ w.r.t. $\Sigma$ is bounded by $|D| \cdot f(\Sigma)$, for some computable function $f$ that assigns natural numbers to ontologies.
3. $\Sigma$ enjoys a syntactic property (relative to the database $D$) that relies on a non-uniform version of weak-acyclicity.

Interestingly, the result of the chase (whenever is finite) is linear w.r.t. the size of the given database. In the case of simple linear (resp., linear, guarded) rule-based ontologies, the function $f$ in the above characterizations is exponential (resp., double-exponential, triple-exponential) in the arity, and exponential (resp., exponential, double-exponential) in the number of predicates of the underlying schema. We further provide lower bounds on the size of the chase showing that the above upper bounds are worst-case optimal.

We then exploit the above characterizations (in fact, item (3)) to establish several complexity results for the problem in question that range from $AC_0$ to $2\text{EXPTIME}$. Among other results, we obtain that non-uniform semi-oblivious chase termination for linear (resp., guarded) rule-based ontologies is in $AC_0$ (resp., $\text{PTIME}$-complete) in data complexity, i.e., when the ontology is considered fixed.

Towards the above characterizations, we establish results of independent interest concerning the basic techniques of simplification and linearization, originally introduced in the context of ontological query answering. Simplification eliminates the repetition of variables in rule-bodies, while linearization converts guarded rule-based ontologies into linear ones. We show that both techniques can be applied in the context of chase termination in the sense that they preserve the finiteness of the chase, and, more importantly, the depth of the terms occurring in the result of the chase.

2 PRELIMINARIES

We consider the disjoint countably infinite sets $C$, $N$, and $V$ of constants, (labeled) nulls, and variables, respectively. We refer to constants, nulls and variables as terms. For an integer $n > 0$, we write $[n]$ for the set $\{1, \ldots, n\}$.

Relational Databases. A schema $S$ is a finite set of relation symbols (or predicates) with associated arity. We write $R/n$ to denote that $R$ has arity $n > 0$; we may also write $ar(R)$ for the integer $n$. A (predicate) position of $S$ is a pair $(R, i)$, where $R/n \in S$ and $i \in [n]$, that essentially identifies the $i$-th argument of $R$. We write $pos(S)$ for the set of positions of $S$, that is, the set $\{(R, i) | R/n \in S \text{ and } i \in [n]\}$. An atom over $S$ is an expression of the form $R(t)$, where $R/n \in S$ and $t$ is an $n$-tuple of terms. A fact is an atom whose arguments consist only of constants. For a variable $x$ in $t = (t_1, \ldots, t_n)$, let $pos(R(t), x) = \{(R, i) \mid t_i = x\}$. We write $var(R(t))$ for the set of variables in $t$. The notations $pos(\cdot, x)$ and $var(\cdot)$ extend to sets of atoms. An instance over $S$ is a (possibly infinite) set of atoms over $S$ with constants and nulls. A database over $S$ is a finite set of facts over $S$. The active domain of an instance $I$, denoted $\text{dom}(I)$, is the set of terms (constants and nulls) occurring in $I$. For a singleton instance $\{\alpha\}$, we simply write $\text{dom}(\alpha)$ instead of $\text{dom}(\{\alpha\})$.

Substitutions and Homomorphisms. A substitution from a set of terms $T$ to a set of terms $T'$ is a function $h : T \rightarrow T'$. Henceforth, we treat a substitution $h$ as the set of mappings $\{t \mapsto h(t) \mid t \in T\}$. The restriction of $h$ to a subset $S$ of $T$, denoted $h|_{S}$, is the substitution $\{t \mapsto h(t) \mid t \in S\}$. A homomorphism from a set of atoms $A$ to a set of atoms $B$ is a substitution $h$ from the set of terms in $A$ to the set of terms in $B$ such that $h$ is the identity on $C$, and $R(t_1, \ldots, t_n) \in A$ implies $h(R(t_1, \ldots, t_n)) = B(h(t_1), \ldots, h(t_n)) \in B$.

Tuple-Generating Dependencies. A tuple-generating dependency (TGD) $\sigma$ is a (constant-free) first-order sentence of the form $\forall \bar{x} \bar{y}\exists \bar{z} \psi(\bar{x}, \bar{y}, \bar{z})$, where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are tuples of variables of $V$, and $\psi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are non-empty conjunctions of atoms that mention only variables from $\bar{x} \cup \bar{y}$ and $\bar{x} \cup \bar{z}$, respectively. Note that, by abuse of notation, we may treat a tuple of variables as a set of variables. We write $\sigma$ as $\psi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for joining atoms. We refer to $\psi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ as the body and head of $\sigma$, denoted $\text{body}(\sigma)$ and $\text{head}(\sigma)$, respectively. The frontier of the TGD $\sigma$, denoted $\text{fr}(\sigma)$, is the set of variables $\bar{x}$, i.e., the variables that appear both in the body and the head of $\sigma$. The schema of a set $\Sigma$ of TGDs, denoted $\text{sch}(\Sigma)$, is the set of predicates occurring in $\Sigma$, and we write $ar(\Sigma)$ for the maximum arity over all those predicates. We assume, w.l.o.g., that no two TGDs of $\Sigma$ share a variable, and we let $|\Sigma| = \vert \text{atoms}(\Sigma) \vert = \vert \text{sch}(\Sigma) \vert \cdot ar(\Sigma)$, where $\text{atoms}(\Sigma)$ is the set of atoms occurring in the TGDs of $\Sigma$. An
instance $I$ satisfies a TGD $\sigma$ as the one above, written $I \models \sigma$, if whenever there exists a homomorphism $h$ from $\phi(x, y)$ to $I$, then there is $h' \supseteq h$ that is a homomorphism from $\psi(x, z)$ to $I$; we may treat a conjunction of atoms as a set of atoms. The instance $I$ satisfies a set $\Sigma$ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$.

**Guaranteed.** A TGD $\sigma$ is called guarded if there exists an atom in body($\sigma$) that contains (or “guards”) all the variables in body(\sigma). Conventionally, the leftmost such an atom in body(\sigma) is the guard of $\sigma$, denoted guard($\sigma$). The class of guarded TGDs, denoted $G$, is defined as the family of all finite sets of guarded TGDs. A TGD is called linear if it has only one atom in its body, and the corresponding class is denoted $L$. We further call a linear TGD simple if no variable occurs more than once in its body-atom, and the corresponding class is denoted $S$. It is clear that $S \subseteq G \subseteq L$.

### 3 THE SEMI-OBLIVIOUS CHASE PROCEDURE

The semi-oblivious chase procedure (or simply chase) takes as input a database $D$ and a set $\Sigma$ of TGDs, and constructs an instance that contains $D$ and satisfies $\Sigma$. Central notions in this context are those of trigger, active trigger, and trigger application.

**Definition 3.1.** Given a set $\Sigma$ of TGDs, and an instance $I$, a trigger for $\Sigma$ on $I$ is a pair $(h, s)$, where $h \in \Sigma$ and $s$ is a homomorphism from body($h$) to $I$. The result of $(h, s)$, denoted result($h, s$), is the set $\mu(h, s) = \{x \mid \var(body(h)) \rightarrow C \cup \Sigma\}$ defined as follows:

$$
\mu(h, s) = \begin{cases} 
\{h(x)\} & \text{if } x \in \text{fr}(s) \\
\bot & \text{otherwise}
\end{cases}
$$

where $\bot = \bot_{\sigma(h, s)}$ is a null value from $\mathbb{N}$. The trigger $(h, s)$ is active if result($h, s$) $\nsubseteq I$. The application of $(h, s)$ to $I$ returns the instance $J = I \cup \text{result}(h, s)$, and is denoted as $I(h, s)$. 

Observe that in the definition of result($h, s$) above each existentially quantified variable $x$ of head($h$) is mapped by $\mu$ to a null value of $\mathbb{N}$ whose name is uniquely determined by the trigger $(h, s)$ and $x$ itself. This means that, given a trigger $(h, s)$, we can unambiguously write down the set of atoms result($h, s$).

The main idea of the chase is, starting from a database $D$, to exhaustively apply active triggers for the given set $\Sigma$ of TGDs on the instance constructed so far. This is formalized via the notion of (semi-oblivious) chase derivation, which can be finite or infinite.

**Definition 3.2.** Consider a database $D$, and a set $\Sigma$ of TGDs.

- A finite sequence $(I_i)_{0 \leq i \leq n}$ of instances, with $D = I_0$ and $n \geq 0$, is a chase derivation of $D$ w.r.t. $\Sigma$ if, for each $i \in \{0, \ldots, n-1\}$, there is a trigger $(h, s)$ for $\Sigma$ on $I_i$ with $I_i(h, s)I_{i+1}$, and there is no active trigger for $\Sigma$ on $I_n$. The result of such a chase derivation is the instance $I_n$.
- An infinite sequence $(I_i)_{i \geq 0}$ of instances, with $D = I_0$, is a chase derivation of $D$ w.r.t. $\Sigma$ if, for each $i \geq 0$, there is an active trigger $(h, s)$ for $\Sigma$ on $I_i$ such that $I_i(h, s)I_{i+1}$. Moreover, $(I_i)_{i \geq 0}$ is fair if, for each $i \geq 0$, and for every active trigger $(h, s)$ for $\Sigma$ on $I_i$, there exists $j > i$ such that $(h, s)$ is not an active trigger for $\Sigma$ on $I_j$. The result of such a chase derivation is the instance $\bigcup_{i \geq 0} I_i$.

A chase derivation is valid if it is finite, or infinite and fair.

Let us stress that infinite but unfair chase derivations are not considered as valid ones since they do not serve the main purpose of the chase, that is, to build an instance that satisfies the given set of TGDs. Indeed, given the set $\Sigma$ consisting of the TGDs

$$
\sigma = R(x, y) \rightarrow \exists z R(y, z) \quad \sigma' = R(x, y) \rightarrow P(x, y),
$$

the result of the unfair chase derivation of $D = \{R(a, b)\}$ w.r.t. $\Sigma$ that involves only triggers of the form $(\sigma, \cdot)$, i.e., only the TGD $\sigma$ is used, does not satisfy $\sigma'$, and thus, it does not satisfy $\Sigma$.

**Non-Uniform Chase Termination.** A valid chase derivation may be infinite even for very simple settings: it is easy to see that the only chase derivation of the database $D = \{R(a, b)\}$ w.r.t. the singleton set of TGDs $\Sigma = \{R(x, y) \rightarrow \exists z R(y, z)\}$ is infinite. The questions that come up are, given a database $D$ and a set $\Sigma$ of TGDs:

1. (1) What is the worst-case optimal size (that is, the cardinality) of the result of a finite chase derivation $D$ w.r.t. $\Sigma$?
2. (2) Can we decide whether all or some valid chase derivations of $D$ w.r.t. $\Sigma$ are finite?

To properly formalize the above questions, we need to recall some central classes of sets of TGDs, parameterized by a database $D$:

$$
\text{CT}_{D} = \{ \Sigma \mid \text{every valid chase derivation of } D \text{ w.r.t. } \Sigma \text{ is finite} \}
$$

and

$$
\text{CT}_{D}^\exists = \{ \Sigma \mid \text{there exists a valid chase derivation of } D \text{ w.r.t. } \Sigma \text{ that is finite} \}.
$$

It is well-known that, for every database $D$, $\text{CT}_{D}^\exists = \text{CT}_{D}^\exists D$ [19]; henceforth, we simply write $\text{CT}_{D}^\exists$ for $\text{CT}_{D}^\exists D$ and $\text{CT}_{D}^\exists$. Furthermore, it is known that for every database $D$ and set $\Sigma$ of TGDs, two distinct valid chase derivations of $D$ w.r.t. $\Sigma$ have always the same result, which we denote by chase($D, \Sigma$). Consequently,

$$
\text{CT}_{D} = \{ \Sigma \mid \text{the instance chase($D, \Sigma$) is finite} \}.
$$

Therefore, question (1) above essentially asks for a worst-case optimal upper bound on the cardinality of the instance chase($D, \Sigma$), assuming that $\Sigma \in \text{CT}_{D}$, that is, chase($D, \Sigma$) is finite. On the other hand, question (2) above corresponds to the following decision problem, parameterized by a class $C$ of sets of TGDs:

**Problem:** $\text{ChTrm}(C)$

**Input:** A database $D$ and a set $\Sigma \in C$.

**Question:** Is it the case that $\Sigma \in \text{CT}_{D}$?

**Our Goal.** With TGD being the class of arbitrary sets of TGDs, we know that $\text{ChTrm}(TGD)$ is undecidable. This was shown in [12] for the restricted chase, but it was observed in [22] that the same proof applies to the semi-oblivious version of the chase. In view of the fact that the set of TGDs employed in the undecidability proof of [12] is far from being guarded, we are interested in studying the above questions for guarded TGDs, and subclasses thereof. In other words, we are focussing on the following questions, for $C \in \{\text{SL}, L, G\}$:

- Given a database $D$, and a set of TGDs $\Sigma \in C \cap \text{CT}_{D}$, what is the complexity of $\text{ChTrm}(C)$?
- Is $\text{ChTrm}(C)$ decidable, and if yes, what is the complexity?
The rest of the paper is devoted to providing answers to the above research questions. To this end, we are going to establish, for each class $C \in \{SL, L, G\}$, a characterization of non-uniform chase termination of the following form, which is of independent interest:

**Target Characterization.** Consider a database $D$, and a set $\Sigma \in C$ of TGDs. The following are equivalent:

1. $\Sigma \in CT_D$.
2. $|\text{chase}(D, \Sigma)| \leq |D| \cdot f_C(\Sigma)$, for some computable function $f_C : C \to \mathbb{N}$ (as usual, $\mathbb{N}$ denotes the set of natural numbers).
3. $\Sigma$ enjoys a syntactic property (relative to the database $D$) that relies on a non-uniform version of weak-acyclicity.

Notice that, for all the classes of TGDs in question, the size of the chase instance is linear w.r.t. the size of the given database. We further complement the above characterization with a lower bound showing that the provided upper bound is worst-case optimal.

Once we have the above results in place, for each class $C \in \{SL, L, G\}$ of TGDs, we will immediately get an answer concerning the worst-case optimal size of the chase instance. Concerning the decidability of ChTrm($C$), it is clear that item (2) of the above characterization will provide a simple decision procedure: given a database $D$ and a set $\Sigma \in C$, simply construct the instance chase($D, \Sigma$), and if $|\text{chase}(D, \Sigma)|$ exceeds the value $|D| \cdot f_C(\Sigma)$, then reject; otherwise, accept. However, in most of the cases, this naive approach will not provide worst-case optimal complexity upper bounds. Towards optimal complexity bounds, we are going to exploit item (3) of the above characterization. Indeed, as we shall see, in all the cases, the procedure of checking whether the syntactic property in question holds provides optimal complexity bounds for ChTrm($C$).

### 4 UNIFORMLY VS. NON-UNIFORMLY TERMINATING CHASE

The first step of our analysis towards the characterizations described above, will be to provide a generic upper bound on the size of the instance chase($D, \Sigma$) (whenever is finite), for a database $D$ and a set $\Sigma$ of guarded TGDs, of the form $|D| \cdot k$, where $k$ depends only on $\Sigma$ and the depth of the terms occurring in chase($D, \Sigma$) (the notion of depth is defined below). We will then prove, for each class $C \in \{SL, L, G\}$ of TGDs, a database-independent upper bound on the depth of the terms that occur in a chase instance. By combining the above bounds, we will eventually get the desired upper bound on the size of the form $|D| \cdot f_{\text{chase}}(\Sigma)$ on the size of chase($D, \Sigma$) (whenever is finite), for a database $D$ and a set of TGDs $\Sigma \in C$.

At this point, one may think that the above bounds can be obtained by merely adapting existing results for arbitrary (not necessarily guarded) TGDs concerning the uniformly terminating chase. Therefore, before we proceed with our analysis, we would like to stress the different nature of the non-uniformly terminating chase.

**Bounded Chase Size.** We know from [22] that, if we focus on arbitrary sets of TGDs that ensure the uniform termination of the chase, then we can bound the size of the chase instance via a uniform function over the input database. More precisely, with

$$\text{CT} = \{ \Sigma \in \text{TGD} \mid \text{for every database } D, \text{chase}(D, \Sigma) \text{ is finite} \}$$

the following result is implicit in [22]:

**Theorem 4.1.** For a set $\Sigma \in \text{CT}$, there is a computable function $f_{\Sigma}$ from databases to $\mathbb{N}$ s.t., for every database $D$, $|\text{chase}(D, \Sigma)| \leq f_{\Sigma}(D)$.

Let us stress that the function $f_{\Sigma}$ provided by Theorem 4.1 is actually a polynomial, which in turn implies that, for a set $\Sigma \in \text{CT}$, $|\text{chase}(D, \Sigma)|$ is polynomial w.r.t. the size of $D$, for every database $D$. One may think that a version of Theorem 4.1 for the non-uniform case, where guardedness does not play any crucial role, can be easily obtained by adapting the proof of Theorem 4.1. In other words, one may expect a result of the following form: for a set of (not necessarily guarded) TGDs $\Sigma$, there is a computable function $f_\Sigma$ from databases to $\mathbb{N}$ such that, for every database $D$ with $\Sigma \in CT_D$, $|\text{chase}(D, \Sigma)| \leq f_\Sigma(D)$. We proceed to show that this is not true.

**Proposition 4.2.** There exists a set $\Sigma$ of TGDs such that the following holds: for every computable function $f_\Sigma$ from databases to $\mathbb{N}$, there exists a database $D$ with $\Sigma \in CT_D$ such that $|\text{chase}(D, \Sigma)| > f_\Sigma(D)$.

To show the above proposition, we first strengthen the undecidability result for ChTrm($\Sigma$) from [12] by showing that it remains undecidable even in data complexity. More precisely, we show that there exists a set $\Sigma^*$ of TGDs such that the problem

| PROBLEM: | ChTrm($\Sigma^*$) |
|----------|------------------|
| INPUT:   | A database $D$. |
| QUESTION:| Is it the case that $\Sigma^* \in CT_D$? |

is undecidable. The proof is by a reduction from the halting problem; in fact, we adapt the reduction given in [12] so that only the database depends on the Turing machine, while the set of TGDs is fixed. Now, by contradiction, assume that there is a computable function $f_{\Sigma^*}$ from databases to $\mathbb{N}$ such that, for every database $D$ with $\Sigma^* \in CT_D$, $|\text{chase}(D, \Sigma^*)| \leq f_{\Sigma^*}(D)$. This implies that, for every database $D$, $\Sigma^* \in CT_D$ iff $|\text{chase}(D, \Sigma^*)| \leq f_{\Sigma^*}(D)$. Since $f_{\Sigma^*}$ is computable, we conclude that ChTrm($\Sigma^*$) is decidable, which is a contradiction.

**Bounded Term Depth.** As said above, a key notion that will play a crucial role in our analysis is the depth of a term occurring in a chase instance. The formal definition follows.

**Definition 4.3.** Consider a database $D$, and a set $\Sigma$ of TGDs. For a term $t \in \text{dom}(\text{chase}(D, \Sigma))$, the depth of $t$, denoted depth$(t)$, is inductively defined as follows:

- if $t$ is a constant, then depth$(t) = 0$, and
- if $t$ is a null value of the form $\bot_{\sigma, k}$, then depth$(t) = 1 + \max\{|\text{depth}(h(x)) | x \in \text{fr}(\sigma)\} \cup \{0\}$.

We write $\text{maxdepth}(D, \Sigma)$ for the maximal depth over all terms of $\text{dom}(\text{chase}(D, \Sigma))$, that is, $\max_{t \in \text{dom}(\text{chase}(D, \Sigma))}\{\text{depth}(t)\}$; in case chase($D, \Sigma$) is infinite, then $\text{maxdepth}(D, \Sigma) = \infty$.

An interesting result in the case of uniformly terminating chase, which provides a database-independent bound on the depth of terms in a chase instance, has been recently established in [7]:

**Theorem 4.4.** For every set $\Sigma \in \text{CT}$, there exists an integer $k_{\Sigma} \geq 0$ such that, for every database $D$, $\text{maxdepth}(D, \Sigma) \leq k_{\Sigma}$.

Again, one may be tempted to think that a version of Theorem 4.4 for the non-uniform case, where guardedness does not play any crucial role, can be obtained by adapting the proof of Theorem 4.4. It is not difficult to show though that this is not the case.
Proposition 4.5. There is a set $\Sigma$ of TGDs, and family of databases $(D_n)_{n \geq 1}$ with $n = |D_n|$ and $\Sigma \in \text{CT}_{D_n}$ s.t. $\maxdepth(D_n, \Sigma) = n - 1$.

To prove the above result, consider the family $(D_n)_{n \geq 1}$, where

$$D_n = \{ P(a_1, b, b), R(a_1, a_2), R(a_2, a_3), \ldots, R(a_{n-1}, a_n) \},$$

and the singleton set of TGDs $\Sigma$ consisting of

$$\sigma = R(x, y), P(x, z, v) \rightarrow \exists \psi P(y, w, z).$$

It is clear that $\Sigma \in \text{CT}_{D_n}$, and that $\maxdepth(D_n, \Sigma) = n - 1$. Note that this does not contradict Theorem 4.4 since $\Sigma \notin \text{CT}$; indeed, $\text{chase}(D, \Sigma)$, where $D = \{ P(a, a, a), R(a, a) \}$, is infinite.

Propositions 4.2 and 4.5 illustrate the different nature of the non-uniformly terminating chase compared to the uniformly terminating one. As we shall see, our analysis reveals that there are versions of Theorems 4.1 and 4.4 for the non-uniform case, providing that we concentrate on guarded TGDs, which rely on techniques and proofs different than the ones used for Theorems 4.1 and 4.4.

5 Bounding the Chase Size

We proceed to provide a generic upper bound on the size of the chase instance $\text{chase}(D, \Sigma)$ (whenever it is finite), for some database $D$ and set $\Sigma \in G$ of TGDs, based on $\maxdepth(D, \Sigma)$. To this end, we first need to recall some auxiliary notions.

Guarded Chase Forest and Trees. Consider a valid chase derivation $\delta = (i_1)_{i \geq 0}$ of $D$ w.r.t. $\Sigma$ with $i_1(\sigma_1, h_1)_{i+1}$ for each $i \geq 0$, which means that $I_{i+1} = I_i \cup \text{result}(\sigma_i, h_i)$. The guarded chase forest of $\delta$, denoted $\text{gforest}(\delta)$, is the directed graph $(V, E)$, where $V = \bigcup_{i \geq 0} I_i$, and $(\alpha, \beta) \in E$ iff there is $i \geq 0$ such that $\alpha = h_i(\text{guard}(\sigma_i))$ and $\beta \in I_{i+1} \setminus I_i$. It is easy to verify that $\text{gforest}(\delta)$ is indeed a forest consisting of directed trees rooted at the atoms of $D$. For an atom $\alpha \in D$, let $\text{gtree}(\delta, \alpha)$ be the tree of $\text{gforest}(\delta)$ rooted at $\alpha$. In what follows, by abuse of notation, we may treat $\text{gforest}(\delta)$ and $\text{gtree}(\delta, \alpha)$ as the sets of their nodes, namely as sets of atoms.

The Generic Bound. The notion of depth defined for terms can be transferred to atoms: given an atom $\alpha = R(t_1, \ldots, t_n) \in \text{chase}(D, \Sigma)$, the depth of $\alpha$, denoted $\text{depth}(\alpha)$, is defined as $\max_{i \in \{0, \ldots, n\}} \{ \text{depth}(t_i) \}$. We show a lemma that provides an upper bound on the number of atoms of a certain depth occurring in $\text{gtree}(\delta, \alpha)$. We define the set

$$\text{gtree}^i(\delta, \alpha) = \{ \beta \in \text{gtree}(\delta, \alpha) \mid \text{depth}(\beta) = i \},$$

that is, the set of atoms of $\text{gtree}(\delta, \alpha)$ of depth $i \geq 0$. We can then show the following key technical lemma:

Lemma 5.1. Consider a database $D$, and a set $\Sigma \in G$ of TGDs. Let $\delta$ be a valid chase derivation of $D$ w.r.t. $\Sigma$. For each $\alpha \in D$ and $i \geq 0$,

$$|\text{gtree}^i(\delta, \alpha)| \leq |\Sigma|^2 \cdot |\text{ar}(\Sigma)|^{(d+1)}.$$

The proof of the above result is by induction on the depth $i \geq 0$. We can now show the following result.

Proposition 5.2. Consider a database $D$, and a set $\Sigma \in G \cap \text{CT}_D$ of TGDs. With $d = \maxdepth(D, \Sigma)$, it holds that

$$|\text{chase}(D, \Sigma)| \leq |D| \cdot (d+1) \cdot |\Sigma|^2 \cdot |\text{ar}(\Sigma)|^{(d+1)}.$$

To prove the above result, consider a valid chase derivation $\delta$ of $D$ w.r.t. $\Sigma$. It is straightforward to see that, for an atom $\alpha \in D$,

$$\text{gtree}(\delta, \alpha) = \bigcup_{d=0}^{d} \text{gtree}^i(\delta, \alpha).$$

Therefore, by Lemma 5.1, $|\text{gtree}(\delta, \alpha)| \leq (d+1) \cdot |\Sigma|^2 \cdot |\text{ar}(\Sigma)|^{(d+1)}$.

Hence, $|\text{chase}(D, \Sigma)| \leq |D| \cdot (d+1) \cdot |\Sigma|^2 \cdot |\text{ar}(\Sigma)|^{(d+1)}$, as needed.

The way that Proposition 5.2 is stated gives the impression that the provided upper bound on $|\text{chase}(D, \Sigma)|$ is at least exponential w.r.t. $D$, since the depth of a null may depend on $D$, which is of course undesirable. Interestingly, as we shall see in the next sections, the depth of a term is actually bounded by a constant that depends only on the set of TGDs, and thus, $|\text{chase}(D, \Sigma)|$ is always linear in $|D|$, for all the classes of TGDs considered in this work. In particular, we will see that, for each class of TGDs $C \subseteq \{ \text{SL}, \text{L}, \text{G} \}$, given a database $D$ and a set $\Sigma \in C \cap \text{CT}_D$, $\maxdepth(D, \Sigma) \leq \text{dc}(\Sigma)$, where the function $\text{dc} : C \rightarrow \mathbb{N}$ is defined as follows:

$$\text{dc}(\Sigma) = |\text{sch}(\Sigma)| \cdot |\text{ar}(\Sigma)|$$

$$\text{dc}(\Sigma) = |\text{sch}(\Sigma)| \cdot |\text{ar}(\Sigma)|^{(d+1)}$$

$$\text{dc}(\Sigma) = |\text{sch}(\Sigma)| \cdot |\text{ar}(\Sigma)|^{(d+1)} \cdot 2 \cdot |\text{sch}(\Sigma)| \cdot |\text{ar}(\Sigma)|^{(d+1)}.$$

6 Simple Linear TGDs

We now concentrate on the class of simple linear TGDs, the easier class to analyse among the classes of TGDs considered in this work, and provide a characterization of non-uniform chase termination as the one described in Section 3, together with a matching lower bound for the size of the chase instance. We then exploit this characterization to pinpoint the complexity of $\text{ChTrm}(\text{SL})$.

We know from [8] that in the case of simple linear TGDs, uniform chase termination can be characterized via weak-acyclicity. Recall that weak-acyclicity was introduced in [13] as the main formalism for data exchange purposes, which guarantees finiteness of the chase for every input database. Unsurprisingly, we employ a non-uniform variant of weak-acyclicity, that is, a database-dependent variant of weak-acyclicity, for characterizing non-uniform chase termination. Let us clarify, however, that since we want to provide a worst-case optimal upper bound on the size of the chase instance, we need to rely on a more refined analysis than the one performed in [8]; the size of the chase instance was not considered in [8].

Non-Uniform Weak-Acylicity. We first need to recall the notion of the dependency graph of a set $\Sigma$ of TGDs, defined as a directed multigraph $\text{dg}(\Sigma) = (N, E)$, where $N = \text{pos}(\text{sch}(\Sigma))$, and $E$ contains only the following edges. For each TGD $\sigma \in \Sigma$ with $\text{head}(\sigma) = \{ \alpha_1, \ldots, \alpha_k \}$, for each $x \in \text{fr}(\sigma)$, and for each position $\pi \in \text{pos}(\text{body}(\sigma), x)$:

- For each $i \in [k]$, and for each $\pi' \in \text{pos}(\alpha_i, x)$, there exists a normal edge $(\pi, \pi') \in E$.
- For each existentially quantified variable $z$ in $\sigma$, $i \in [k]$, and $\pi' \in \text{pos}(\alpha_i, z)$, there is a special edge $(\pi, \pi') \in E$.
We further need to define when a predicate is reachable from another predicate. Given predicates \( R, P \in \sigma(\Sigma) \), we write \( R \rightarrow P \) if \( R = P \), or there exists a TGD \( \sigma \in \Sigma \) such that \( R \) occurs in body(\( \sigma \)) and \( P \) occurs in head(\( \sigma \)). We say that \( P \) is reachable from \( R \) (w.r.t. \( \Sigma \)), denoted \( R \rightarrow^{\Sigma} P \), if (i) \( R \rightarrow P \), or (ii) there exists \( T \in \sigma(\Sigma) \) such that \( R \rightarrow^{\Sigma} T \) and \( T \rightarrow^{\Sigma} P \). Given a database \( D \), we say that a (not necessarily simple, and possibly cyclic) path \( C \) in \( \sigma(\Sigma) \) is \( D \)-supported if there is \( R(i) \in D \) and a node \((P, i)\) in \( C \) with \( R \rightarrow^{\Sigma} P \).

**Definition 6.1.** Consider a database \( D \), and a set \( \Sigma \) of TGDs. We say that \( \Sigma \) is weakly-acyclic w.r.t. \( D \), or simply \( D \)-weakly-acyclic, if there is no \( D \)-supported cycle in \( \sigma(\Sigma) \) with a special edge. □

The key properties of non-uniform weak-acyclicity that are crucial for our analysis are summarized by the following two technical lemmas. The first one, which holds for arbitrary TGDs, provides a database-independent upper bound on the depth of terms occurring in chase(\( D, \Sigma \)) via \( d_{SL} : SL \rightarrow \mathbb{N} \) defined in Section 5; recall that \( d_{SL}(\Sigma) = |sch(\Sigma)| \cdot |rel(\Sigma)| \).

**Lemma 6.2.** Consider a database \( D \), and a set \( \Sigma \) of TGDs that is \( D \)-weakly-acyclic. It holds that maxdepth(\( D, \Sigma \)) \( \leq d_{SL}(\Sigma) \).

What is more interesting is the fact that whenever there exists a bound on the depth of the terms occurring in a chase instance chase(\( D, \Sigma \)), for a database \( D \) and a set \( \Sigma \in SL \), then \( \Sigma \) is necessarily \( D \)-weakly-acyclic. For this property, simple linearity is crucial.

**Lemma 6.3.** Consider a database \( D \), and a set \( \Sigma \in SL \). If there is \( k \geq 0 \) such that maxdepth(\( D, \Sigma \)) \( \leq k \), then \( \Sigma \) is \( D \)-weakly-acyclic.

As discussed above, (uniform) weak-acyclicity has been used in [8] to characterize uniform chase termination in the case of simple linear TGDs. However, [8] did not establish properties analogous to Lemmas 6.2 and 6.3 for (uniform) weak-acyclicity since estimating the size of the chase was not part of the investigation.

**Characterizing Non-Uniform Termination.** We are now ready to establish the desired characterization of non-uniform chase termination in the case of simple linear TGDs. Let \( f_{SL} \) be the function from \( SL \) to the natural numbers \( \mathbb{N} \) defined as follows:

\[
    f_{SL}(\Sigma) = (d_{SL}(\Sigma) + 1) \cdot (|\Sigma|^2 \cdot |rel(\Sigma)|^{d_{SL}(\Sigma) + 1}).
\]

We can then show the following result:

**Theorem 6.4.** Consider a database \( D \), and a set \( \Sigma \in SL \) of TGDs. The following are equivalent:

1. \( \Sigma \in CT_D \).
2. \( |\text{chase}(D, \Sigma)| \leq |D| \cdot f_{SL}(\Sigma) \).
3. \( \Sigma \) is \( D \)-weakly-acyclic.

Proof. The interesting directions are (1) \( \Rightarrow \) (3) and (3) \( \Rightarrow \) (2); (2) \( \Rightarrow \) (1) is trivial. For (1) \( \Rightarrow \) (3), observe that if \( \Sigma \in CT_D \), which means that chase(\( D, \Sigma \)) is finite, then there is \( k \geq 0 \) that bounds maxdepth(\( D, \Sigma \)), and (3) follows by Lemma 6.3. The statement (3) \( \Rightarrow \) (2) follows from Proposition 5.2 and Lemma 6.2. □

We complement the above characterization with a result that essentially states the following: the size of the chase instance is unavoidably exponential in the arity and the number of predicates of the underlying schema.

**Theorem 6.5.** There exists a family of databases \( \{D_\ell\}_{\ell \geq 0} \) with \( \ell = |D_\ell| \), and a family of sets of TGDs \( \{\Sigma, n, m \in SL \cap CT_{D_\ell}\}_{n, m \geq 0} \) with \( n = |sch(\Sigma)| - 1 \) and \( m = ar(\Sigma, n, m) \), such that

\[
    |\text{chase}(D_\ell, \Sigma, n, m)| \geq \ell \cdot m^{n-1}.
\]

**Complexity.** We now study the complexity of ChTrm(SL). Observe that the naive approach, which relies on item (2) of Theorem 6.4, that explicitly constructs the chase instance, shows that ChTrm(SL) is in ExpTime (even if we bound the arity, we still get ExpTime), and in PTime in data complexity, i.e., when the set of TGDs is considered fixed. Notice also that Theorem 6.5 tells us that this is the best that we can achieve via the naive approach since there is no way to lower the upper bound on the size of the chase instance provided by Theorem 6.4. It turns out that the exact complexity of ChTrm(SL) is significantly lower than what we get via the naive approach.

**Theorem 6.6.** ChTrm(SL) is NL-complete, even for schemas with unary and binary predicates, and in \( AC_0 \) in data complexity.

The finer procedures that lead to Theorem 6.6 rely on item (3) of Theorem 6.4. In particular, we know that \( \Sigma \in CT_D \) if \( \Sigma \) is \( D \)-weakly-acyclic. Thus, it suffices to show that, given a database \( D \) and a set \( \Sigma \in SL \) of TGDs, the problem of deciding whether \( \Sigma \) is not \( D \)-weakly-acyclic is in NL in general, and in \( AC_0 \) when \( \Sigma \) is fixed. The former is shown via a nondeterministic algorithm that performs a reachability check on the dependency graph of \( \Sigma \), while the latter is shown by constructing a union of conjunctive queries \( Q_\Sigma \) (which depends only on \( \Sigma \)) such that \( \Sigma \) is not \( D \)-weakly-acyclic if \( D \) satisfies \( Q_\Sigma \).

The NL-hardness of ChTrm(SL), even for schemas with unary and binary predicates, is inherited from [8], where uniform chase termination is studied. In particular, [8] shows that, given a set \( \Sigma \in SL \) such that \( sch(\Sigma) \) consists of unary and binary predicates, the following problem is NL-hard: with \( D_\Sigma = \{P(c) | P/1 \in sch(\Sigma)\} \cup \{R(c, c) | R/2 \in sch(\Sigma)\} \), is it the case that chase(\( D_\Sigma, \Sigma \)) is finite?

7 **LINEAR TGDs**

We now concentrate on the class of linear TGDs, and provide a characterization of non-uniform chase termination as the one described in Section 3, together with a matching lower bound for the size of the chase instance. We then exploit this characterization to pinpoint the complexity of ChTrm(L). In contrast to simple linear TGDs, non-uniform weak-acyclicity is not powerful enough for characterizing the finiteness of the chase instance.

**Example 7.1.** Consider the database \( D = \{R(a, b)\} \), and the singleton set \( \Sigma \) consisting of the (non-simple) linear TGD

\[
    R(x, x) \rightarrow \exists x R(z, x).
\]

It is easy to see that there is no trigger for \( \Sigma \) on \( D \). This means that \( \text{chase}(D, \Sigma) = D \) is finite, whereas \( \Sigma \) is not \( D \)-weakly-acyclic. □

In [8], an extended version of weak-acyclicity, called critical-weak-acyclicity, has been proposed for characterizing uniform chase termination in the case of linear TGDs. Therefore, one could employ a non-uniform version of critical-weak-acyclicity for characterizing non-uniform chase termination. However, due to the involved definition of critical-weak-acyclicity, it is not clear how a database-independent bound on the depth of the terms occurring in a chase.
instance can be established. Hence, to obtain a result analogous to Theorem 6.4, we exploit a technique, called simplification, that converts linear TGDs into simple linear TGDs. This is a rather folklore technique in the context of ontological query answering, but this is the first time that it is being applied in the context of chase termination. Thus, our main technical task will be to show that simplification preserves the finiteness of the chase, and, more importantly, the depth of the terms occurring in a chase instance. This is a non-trivial task since the chase instance of the simplified version of a set \( \Sigma \) of linear TGDs is structurally different than the chase instance of \( \Sigma \) in the sense that there is no immediate correspondence between their terms and atoms.

**Simplification.** Let \( \bar{t} = (t_1, \ldots, t_n) \) be a tuple of (not necessarily distinct) terms. We write \( \text{unique}(\bar{t}) \) for the tuple obtained from \( \bar{t} \) by keeping only the first occurrence of each term in \( \bar{t} \). For example, if \( \bar{t} = (x, y, x, z, y) \), then \( \text{unique}(\bar{t}) = (x, y, z) \). For each \( i \in [n] \), the identifier of \( t_i \) in \( \bar{t} \), denoted \( \text{id}_j(t_i) \), is the integer that identifies the position of \( t_i \) at which \( t_i \) appears. We write \( \text{id}(\bar{t}) \) for the tuple \( (\text{id}_1(t_1), \ldots, \text{id}_j(t_n)) \). For example, if \( \bar{t} = (x, y, x, z, y) \), then \( \text{id}(\bar{t}) = (1, 2, 1, 3, 2) \). For an atom \( \alpha = \text{R}(\bar{t}) \), the simplification of \( \alpha \), denoted \( \text{simple}(\alpha) \), is the atom \( \text{R}_\text{id}(\text{unique}(\bar{t})) \). We can naturally refer to the simplification of a set of atoms. For a tuple of variables \( \bar{x} = (x_1, \ldots, x_n) \), a specialization of \( \bar{x} \) is a function \( f \) from \( \bar{x} \) to \( \bar{x} \) such that \( f(x_1) = x_1 \) and \( f(x_i) \in \{f(x_1), \ldots, f(x_{i-1}), x_i\} \), for each \( i \in \{2, \ldots, n\} \). We write \( f(\bar{x}) \) for \( (f(x_1), \ldots, f(x_n)) \). We are now ready to define the simplification of linear TGDs.

**Definition 7.2.** Consider a linear TGD \( \sigma \) of the form

\[
R(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{y}, \bar{z}),
\]

where \( \bar{y} \subseteq \bar{x} \), and a specialization \( f \) of \( \bar{x} \). The simplification of \( \sigma \) induced by \( f \) is the simple linear TGD

\[
\text{simple}(R(f(\bar{x}))) \rightarrow \exists \bar{y} \text{simple}(\psi(f(\bar{y}), \bar{z})).
\]

We write \( \text{simple}(\sigma) \) for the set of all simplifications of \( \sigma \) induced by some specialization of \( \bar{x} \). For a set \( \Sigma \subseteq L \) of TGDs, the simplification of \( \Sigma \) is defined as the set

\[
\text{simple}(\Sigma) = \bigcup_{\sigma \in \Sigma} \text{simple}(\sigma)
\]

consisting only of simple linear TGDs.

The next result, which is one of our main technical results, shows that the technique of simplification can be safely applied in the context of chase termination in the sense that it preserves the finiteness of the chase, as well as the maximal depth over all terms in a chase instance. The key ingredient underlying the proof of this result is a delicate correspondence between the terms and atoms in the instance \( \text{chase}(D, \Sigma) \) for some database \( D \) and set \( \Sigma \) of linear TGD, with those in the instance \( \text{chase}(\text{simple}(D), \text{simple}(\Sigma)) \).

**Proposition 7.3.** Consider a database \( D \), and a set \( \Sigma \subseteq L \). Then:

1. \( \Sigma \in \text{CT}_D \) if and only if \( \text{simple}(\Sigma) \in \text{CT}_{\text{simple}(D)} \).
2. \( \text{maxdepth}(D, \Sigma) = \text{maxdepth}(\text{simple}(D), \text{simple}(\Sigma)) \).

\( ^1 \)It is easy to see that (2) implies (1), but we explicitly state (1) since it is interesting in its own right, and it is also explicitly used in the proof of Theorem 7.5.

**Characterizing Non-Uniform Termination.** We are now ready to establish a result for linear TGDs analogous to Theorem 6.4 via simplification. To this end, we first provide a database-independent upper bound on the depth of terms occurring in a chase instance via \( d_{\text{f}} : L \rightarrow \mathbb{N} \); recall that \( d_{\text{f}}(\Sigma) = |\text{sch}(\Sigma)| \cdot \text{ar}(\Sigma)^{\text{maxdepth}(\Sigma) + 1} \).

**Lemma 7.4.** Consider a database \( D \), and a set \( \Sigma \subseteq L \) with \( \text{simple}(\Sigma) \) being \( \text{simple}(D) \)-weakly-acyclic. Then \( \text{maxdepth}(D, \Sigma) \leq d_{\text{f}}(\Sigma) \).

**Proof.** By Lemma 6.2, and item (2) of Proposition 7.3,

\[
\text{maxdepth}(D, \Sigma) \leq d_{\text{f}}(\text{simple}(\Sigma)) = |\text{sch}(\text{simple}(\Sigma))| \cdot \text{ar}(\text{simple}(\Sigma)).
\]

It is easy to see that \( |\text{sch}(\text{simple}(\Sigma))| \leq |\text{sch}(\Sigma)| \cdot \text{ar}(\Sigma)^{\text{maxdepth}(\Sigma) + 1} \), and that \( \text{ar}(\text{simple}(\Sigma)) \leq \text{ar}(\Sigma) \). This implies that

\[
\text{maxdepth}(D, \Sigma) \leq |\text{sch}(\Sigma)| \cdot \text{ar}(\Sigma)^{\text{maxdepth}(\Sigma) + 1}
\]

and the claim follows by the definition of the function \( d_{\text{f}} \). \( \square \)

Let \( f_L \) be the function from \( L \rightarrow \mathbb{N} \) defined as follows:

\[
f_L(\Sigma) = (d_{\text{f}}(\Sigma) + 1) \cdot |\Sigma|^{|2 \cdot \text{ar}(\Sigma) + d_{\text{f}}(\Sigma) + 1}.
\]

By exploiting the generic bound provided by Proposition 5.2, the direction \( (1) \Rightarrow (3) \) of Theorem 6.4, item (1) of Proposition 7.3, and Lemma 7.4, it is an easy task to show the following:

**Theorem 7.5.** Consider a database \( D \), and a set \( \Sigma \subseteq L \) of TGDs.

The following are equivalent:

1. \( \Sigma \in \text{CT}_D \).
2. \( |\text{chase}(D, \Sigma)| \leq |D| \cdot f_L(\Sigma). \)
3. \( \text{simple}(\Sigma) \) is \( \text{simple}(D) \)-weakly-acyclic.

We complement the above characterization with a result that states the following: the size of the chase instance is unavoidably double-exponential in the arity, and exponential in the number of predicates of the underlying schema.

**Theorem 7.6.** There exists a family of databases \( \{D_\ell\}_{\ell \geq 0} \) with \( \ell = |D_\ell| \), and a family of sets of TGDs \( \{\Sigma_{n,m} \in L \cap \text{CT}_D\}_{n,m \geq 0} \) with \( n = |\text{sch}(\Sigma_{n,m})| - 1 \) and \( m = \text{ar}(\Sigma_{n,m}) - 3 \), such that

\[
|\text{chase}(D_\ell, \Sigma_{n,m})| \geq \ell \cdot 2^n(2^m-1).
\]

**Complexity.** We proceed with the complexity of \( \text{ChTrm}(L) \). We first note that the naive procedure obtained from item (2) of Theorem 7.5 shows that \( \text{ChTrm}(L) \) is in \( 2\text{ExpTime} \), in \( \text{ExpTime} \) if we bound the arity, and in \( \text{PTime} \) in data complexity. This is provably the best that we can get from the naive approach according to Theorem 7.6. However, the exact complexity of \( \text{ChTrm}(L) \) is lower.

**Theorem 7.7.** \( \text{ChTrm}(L) \) is \text{PSPACE}-complete, \text{NL}-complete for schemas of bounded arity, and in \( \text{AC}_0 \) in data complexity.

The finer procedures that lead to Theorem 7.7 rely on item (3) of Theorem 7.5. In particular, we know that \( \Sigma \in \text{CT}_D \) iff \( \text{simple}(\Sigma) \) is \( \text{simple}(D) \)-weakly-acyclic. Thus, it suffices to show that, given a database \( D \) and a set \( \Sigma \subseteq L \) of TGDs, the problem of deciding whether \( \text{simple}(\Sigma) \) is not \( \text{simple}(D) \)-weakly-acyclic is in \( \text{PSPACE} \) in general, in \( \text{NL} \) in the case of bounded arity, and in \( \text{AC}_0 \) when \( \Sigma \) is fixed. The \( \text{PSPACE} \) and \( \text{NL} \) upper bounds are shown via a non-deterministic algorithm that performs a reachability check on the dependency graph of \( \text{simple}(\Sigma) \), but without explicitly constructing
it. The $AC_0$ upper bound is shown by constructing a union of conjunctive queries $Q_{\alpha}$ (which depends only on $\Sigma$) such that $\text{simple}(\Sigma)$ is not simple$(D)$-weakly-acyclic if $D$ satisfies $Q_{\alpha}$. The hardness results are inherited from [8], where it is shown that the problem is hard for the database consisting of all the atoms that can be formed using one constant and the predicates of the underlying schema.

8 GUARDED TGDS

In this final section, we focus on the class of guarded TGDS, and provide a characterization of non-uniform chase termination as the one described in Section 3, together with a matching lower bound for the size of the chase instance. We then exploit this characterization to pinpoint the complexity of ChTrm($G$). Note that the work [8], where the uniform termination problem for guarded TGDS has been studied, does not provide any syntactic characterization, which can then be used to devise a decision procedure, but it rather relies on a sophisticated alternating algorithm. Towards our characterization, we combine a technique known as linearization [17], which converts a set of guarded TGDS into a set of linear TGDS, with simplification used in the previous section. Linearization is a very useful technique that has found several applications in the context of query answering [2, 4, 5, 17, 18]. This is the first time, however, that it is used in the context of chase termination. Hence, as for simplification, our main technical task will be to show that linearization preserves the finiteness of the chase, and, more importantly, the depth of the terms occurring in a chase instance.

**Linearization.** We proceed to recall the technique of linearization. In particular, we are going to explain how we convert a database $D$ and a set $\Sigma \in G$ into a database $\text{lin}(D)$ and a set $\text{lin}(\Sigma)$ of linear TGDS, respectively, such that a property analogical to the one provided by Proposition 7.3 for simplification holds. Before defining $\text{lin}(D)$ and $\text{lin}(\Sigma)$, we need to introduce some auxiliary notions.

For an atom $a \in \text{chase}(D, \Sigma)$, the type of $a$ w.r.t. $D$ and $\Sigma$, denoted $\text{type}_{D,\Sigma}(a)$, is the set of atoms $\{\beta \in \text{chase}(D, \Sigma) \mid \text{dom}(\beta) \subseteq \text{dom}(a)\}$. The completion of $D$ w.r.t. $\Sigma$, denoted complete($D$, $\Sigma$), is defined as the set of atoms $\{a \in \text{chase}(D, \Sigma) \mid \text{dom}(a) \subseteq \text{dom}(D)\}$.

For a schema $S$, and a set of terms $V$, we write base($S$, $V$) for the set of atoms that can be formed using predicates from $S$ and terms from $V$, that is, the set of atoms $\{R(\bar{v}) \mid R \in S \text{ and } \bar{v} \in \text{var}(R)\}$. A $\Sigma$-type $\tau$ is a pair $(a, T)$, where $a$ is an atom $R(\bar{t}) = R(t_1, \ldots, t_n)$, with $R \in \text{sch}(\Sigma)$, $t_1, t_2, \ldots, t_n \subseteq \text{max}_{\{i \mid i \notin \{1, \ldots, n\}}(\bar{t}) + 1$ for $i \in \{2, \ldots, n\}$, and $T \subseteq \text{base}(\text{sch}(\Sigma), \text{dom}(a)) \setminus \{a\}$. We write guard$(\tau)$ for the atom $a$, atoms$(\tau)$ for the set $T \cup \{a\}$, and ar$(\tau)$ for the integer ar(guard$(\tau)$), that is, the arity of $\tau$. Intuitively, $\tau$ encodes the shape of a guard and its type. Given a tuple $\bar{u} = (u_1, \ldots, u_m)$ that is isomorphic to $\bar{t} = (t_1, \ldots, t_n)$, i.e., $n = m$ and $u_i = u_j$ iff $t_i = t_j$ for each $i, j \in [n]$, the instantiation of $\tau$ with $\bar{u}$, denoted $\tau(\bar{u})$, is the set of atoms obtained from atom ($\tau$) after replacing $t_i$ with $u_i$. Finally, for $\alpha = R(\bar{t})$, we write $\Sigma$-types$(\alpha)$ for the set of all $\Sigma$-types $\tau$ such that guard($\tau$) = $R(\bar{u})$, and $I$ is isomorphic to $\bar{u}$.

We are now ready to define the linearization of $D$ as the database

$$\text{lin}(D) = \left\{ \{\tau(\bar{u}) \mid \tau(\bar{u}) = \text{type}_{D,\Sigma}(R(\bar{t})) \}, \right\},$$

where $\tau$ is a new predicate not occurring in $\text{sch}(\Sigma)$. Here is a simple example that illustrates the above definition.

**Example 8.1.** Consider the database

$$D = \{R(a, a, b, c)\}$$

and the set $\Sigma$ consisting of the guarded TGDS

$$\sigma = P(x, y, x, u, w), S(x, u) \rightarrow \exists z_1 \exists z_2 R(y, u, x, z_1), T(z_1, z_2, x)$$

and

$$\sigma' = R(x, x, y, z) \rightarrow Q(x, z).$$

It is clear that the $\Sigma$-type

$$\tau = (R(1, 1, 2, 3), \{Q(1, 3)\})$$

belongs to $\Sigma$-types$(R(a, a, b, c))$ since it holds that $(a, a, b, c)$ and $(1, 1, 2, 3)$ are isomorphic. Moreover, it is clear that

$$\tau(a, a, b, c) = \{R(a, a, b, c), Q(a, c)\} = \text{type}_{D,\Sigma}(R(a, a, b, c)).$$

We conclude that $\text{lin}(D) = \{\{\tau(\bar{u}) \mid \text{type}_{D,\Sigma}(R(a, a, b, c))\}.$

We now proceed with the linearization of $\Sigma$. Consider a TGD $\sigma \in \Sigma$ of the form

$$\phi(\bar{x}, \bar{y}) \rightarrow \exists z_1 \ldots \exists z_n R_1(\bar{u}_1), \ldots, R_m(\bar{u}_m)$$

with guard$(\sigma) = R(\bar{u})$, and $(\bar{x} \cup \{z_1 \ldots z_n\})$ are the variables occurring in head$(\sigma)$. For a $\Sigma$-type $\tau$ such that there exists a homomorphism $h$ from $\varphi(\bar{x}, \bar{y})$ to atoms$(\tau)$ and $h(\bar{u}) = \text{guard}(\tau)$, the linearization of $\sigma$ induced by $\tau$ and $h$ is

$$\{\tau(\bar{u}) \rightarrow \exists z_1 \ldots \exists z_n [\tau_1(\bar{u}_1), \ldots, \tau_m(\bar{u}_m)],$$

where, for each $i \in [m]$, $\tau_i$ is defined as follows. Let $f$ be the function from the variables in head$(\sigma)$ to $\mathbb{N}$ defined as

$$f(t) = \begin{cases} h(t) & \text{if } t \in \bar{x} \\ \text{ar}(\Sigma) + i & \text{if } t = z_i. \end{cases}$$

With $\alpha_i = R_i(f(\bar{u}_i))$, $\tau_i = (\alpha_i, T_i)$, where $T_i$ is

$$\{\beta \in \text{complete}(I, \Sigma) \mid \text{dom}(\beta) \subseteq \text{dom}(\alpha_i)\} \setminus \{\alpha_i\}$$

with

$$I = \{\alpha_1, \ldots, \alpha_m\} \cup \text{atoms}(\tau).$$

Note that $\tau_i$ is not a proper $\Sigma$-type since the integers in $\alpha_i$ do not appear in the right order. Let $\rho$ be the renaming function that renames the integers in $\alpha_i$ in order to appear in increasing order starting from 1 (e.g., $\rho(R(2, 2, 4, 1)) = R(1, 1, 2, 3)$; the formal definition is omitted since it is clear what the function $\rho$ does). We then define $\tau_i$ as the pair $(\rho(\alpha_i), \rho(T_i))$. We write $\text{lin}(\sigma)$ for the set of all linearizations of $\sigma$, induced by some $\Sigma$-type $\tau$, and homomorphism $h$ from $\varphi(\bar{x}, \bar{y})$ to atoms$(\tau)$ such that $h(\bar{u}) = \text{guard}(\tau)$. Finally, the linearization of $\Sigma$ is defined as the (finite) set of TGDS

$$\text{lin}(\Sigma) = \bigcup_{\sigma \in \Sigma} \text{lin}(\sigma).$$

An example that illustrates the above definition follows.

**Example 8.2.** Let $\Sigma$ be the set of TGDS given in Example 8.1. Consider the $\Sigma$-type

$$\tau = (P(1, 2, 1, 2, 3), \{Q(1, 2), Q(1, 4)\}).$$

It is easy to verify that

$$h = \{x \mapsto 1, y \mapsto 2, u \mapsto 2, w \mapsto 3\}$$

376
is a homomorphism from body(σ) to
atoms(τ) = \{P(1, 2, 1, 2, 3), S(1, 2), S(1, 1)\},
and h(P(x, y, u, w)) = P(1, 2, 1, 2, 3). The linearization of σ
induced by τ and h is the linear TGD
\[ \tau(x, y, u, w) \rightarrow \exists z_1, \exists z_2 [\tau_1(u, y, x, z_1), [\tau_2(z_1, z_2, x)], \]
where
\[ \tau_1 = (R(1, 1, 2, 3), \{S(2, 1), S(2, 2), Q(1, 3)\}), \]
\[ \tau_2 = (T(1, 2, 3), \emptyset). \]

The next result, which is another key technical result of this work,
shows that the technique of linearization can be safely applied in
the context of chase termination in the sense that it preserves the
finiteness of the chase, as well as the maximal depth over all terms in
a chase instance. It actually shows a property analogous to the one
established in Proposition 7.3 for simplification, and the key ingredient
of its proof is an intricate correspondence between the terms and
atoms in the instance chase(\(D, \Sigma\)), for some database D and set \(\Sigma\)
of guarded TGDs, with those in the instance chase(\(\text{lin}(D), \text{lin}(\Sigma)\)).

**Proposition 8.3.** Consider a database D, and a set \(\Sigma \in G\). Then\(^2\)
(1) \(\Sigma \in \text{CT}_{\Sigma}\) if and only if \(\text{lin}(\Sigma) \in \text{CT}_{\text{lin}(D)}\).
(2) \(\text{maxdepth}(D, \Sigma) = \text{maxdepth}(\text{lin}(D), \text{lin}(\Sigma))\).

**Characterizing Non-Uniform Termination.** We proceed to es-
\[ \text{ctablish a result for guarded TGDs analogous to Theorems 6.4 \text{ and 6.5}} \]
\[ \text{of its proof is an intricate correspondence between the terms and}
\[ \text{atoms in the instance chase}(D, \Sigma), \text{for some database D and set } \Sigma \text{ of}
\[ \text{guarded TGDs, with those in the instance chase}(\text{lin}(D), \text{lin}(\Sigma)).} \]

**Lemma 8.4.** Consider a database D, and \(\Sigma \in G\) with \(\text{gsimple}(\Sigma)\)
being \(\text{gsimple}(D)\)-weakly-acyclic. Then \(\text{maxdepth}(D, \Sigma) \leq \text{dc}(\Sigma)\).

**Proof.** By Lemma 6.2, and item (2) of Propositions 7.3 and 8.3,
\[ \text{maxdepth}(D, \Sigma) \leq \text{dc}(\text{gsimple}(\Sigma)) \]
\[ = \text{sch}(\text{gsimple}(\Sigma)) \cdot \text{ar}(\text{gsimple}(\Sigma)). \]

It is not difficult to show that
\[ \text{sch}(\text{gsimple}(\Sigma)) \leq \text{sch}(\text{lin}(\Sigma)) \cdot \text{ar}(\text{lin}(\Sigma)) \text{ar}^2(\text{lin}(\Sigma)). \]

We further know that
\[ \text{sch}(\text{lin}(\Sigma)) \leq \text{sch}(\Sigma) \cdot \text{ar}(\Sigma) \text{ar}^2(\Sigma) \cdot \text{sch}(\Sigma) \cdot \text{ar}(\Sigma) \text{ar}^3(\Sigma). \]

Since \(\text{ar}(\text{gsimple}(\Sigma)) \leq \text{ar}(\text{lin}(\Sigma)) = \text{ar}(\Sigma),\) we can conclude that
\(\text{maxdepth}(D, \Sigma) \leq \text{dc}(\Sigma),\) and the claim follows.

Let \(f_\Sigma\) be the function from G to \(\mathbb{N}\) defined as follows:
\[ f_\Sigma(\Sigma) = (d_\Sigma(\Sigma) + 1) \cdot |\Sigma|^2 \cdot \text{ar}^2(\Sigma) \cdot \text{dc}(\Sigma) + 1. \]

By using the generic bound provided by Proposition 5.2, the di-
rection (1) \(\Rightarrow\) (3) of Theorem 7.5, item (1) of Proposition 8.3, and
Lemma 8.4, it is an easy task to show the following:

**Theorem 8.5.** Consider a database D, and a set \(\Sigma \in G\) of TGDs.
The following are equivalent:
(1) \(\Sigma \in \text{CT}_D\).
(2) \(\text{sch}(D, \Sigma) \leq |D| \cdot f_\Sigma(\Sigma).\)
(3) \(\text{gsimple}(\Sigma)\) is \(\text{gsimple}(D)\)-weakly-acyclic.

We complement the above with a result that states the following:
the size of the chase instance is unavoidably triple-exponential in
the arity, and double-exponential in the number of predicates of
the schema. The formal statement is as follows:

**Theorem 8.6.** There exists a family of databases \(\{D_\ell\}_{\ell > 0}\)
with \(\ell = |D_\ell|\), and a family of sets of TGDs \(\{\Sigma_{n, m}\} \subseteq \text{G} \cap \text{CT}_{D_\ell}\}_{n, m \geq 0}\) with
\[ n = \frac{\text{sch}(\Sigma_{n, m})}{4} - 3 \quad \text{and} \quad m = \frac{\text{ar}(\Sigma_{n, m})}{2} - 2, \] such that
\[ |\text{sch}(D_\ell, \Sigma_{n, m})| \geq \ell \cdot 2^n \left(\frac{4^n}{4} \right). \]

**Complexity.** We proceed with the complexity of \(\text{ChTrm}(G)\). We first note that the naive procedure obtained from item (2) of Theorem 8.5 shows that \(\text{ChTrm}(G)\) is in \(3\text{EXPTime}\), in \(2\text{EXPTime}\) if we bound the arity, and in \(\text{PTIME}\) in data complexity. This is provably the best that we can get from the naive approach according to Theorem 8.6. However, apart from the case of data complexity, the exact complexity of \(\text{ChTrm}(G)\) is significantly lower.

**Theorem 8.7.** \(\text{ChTrm}(G)\) is \(2\text{EXPTime}\)-complete, \(\text{EXPTime}\)-
complete for bounded arity, and \(\text{PTIME}\)-complete in data complexity.

The above result is shown by explicitly constructing \(\text{gsimple}(D)\)
and \(\text{gsimple}(\Sigma),\) on input D and \(\Sigma,\) and then relying on the procedure
for \(\text{ChTrm}(\Sigma).\) The \(2\text{EXPTime}/\text{EXPTime}\) lower bounds are inherited from [8], where uniform chase termination is studied. The \(\text{PTIME}\)-hardness in data complexity is shown by exploiting a technique from [8], known as \(\text{looping operator},\) which allows us to transfer hardness results for ontological query answering to our problem.

**9 CONCLUSIONS**

The results of this work provide a rather complete picture con-
cerning the size and complexity of the non-uniformly terminating
semi-oblivious chase for guarded TGDs, and subclasses thereof.
Note that our results can be adapted to the oblivious chase, a rel-
axation of the semi-oblivious chase. The fact that for linear TGDs
the problem can be solved by evaluating an UCQ, which depends only
on the TGDs, over the given database is particularly interesting as
it allows us to exploit standard database systems. We are currently
working on an experimental evaluation of the obtained procedures.
An interesting direction is to perform a similar analysis for the
restricted version of the chase, which we expect to be even more
challenging. The latter has been confirmed for the uniform version
of the chase termination problem [15].

**Acknowledgments.** We thank the referees for their feedback.
Georg Gottlob was supported by his Royal Society Research Pro-
fessorship, grant “RAISON DATA” RP/R1/201074. Andreas Pieris
was supported by the EPSRC grant EP/S003800/1 “EQUID”.

**REFERENCES**

[1] Serge Abiteboul, Marcelo Arenas, Pablo Barceló, Meghyn Bienvenu, Diego Cal-
vane, Claire David, Richard Hull, Ekke Hüllemeyer, Benny Kimelfeld, Leonid
Libkin, Wim Martens, Tova Milo, Filip Murlak, Frank Neven, Magdalena Ortiz,
Thomas Schwentick, Julia Stoyanovich, Jianwen Su, Dan Suciu, Victor Vianu, and Ke Yi. 2016. Research Directions for Principles of Data Management (Abridged). SIGMOD Rec. 45, 4 (2016), 5–17.

[2] Antoine Amarilli and Michael Benedikt. 2018. When Can We Answer Queries Using Result-Bounded Data Interfaces?. In PODS. 281–293.

[3] Franz Baader, Sebastian Brandt, and Carsten Lutz. 2005. Pushing the EL Envelope. In IJCAI. 364–369.

[4] Jean-François Baget, Meghyn Bienvenu, Marie-Laure Mugnier, and Michael Thomazo. 2017. Answering Conjunctive Regular Path Queries over Guarded Existential Rules. In IJCAI. 793–799.

[5] Pablo Barceló, Victor Dalmau, Cristina Feier, Carsten Lutz, and Andreas Pieris. 2020. The Limits of Efficiency for Open- and Closed-World Query Evaluation Under Guarded TGDs. In PODS. 259–270.

[6] Michael Benedikt, George Konstantinidis, GianSalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. 2017. Benchmarking the Chase. In PODS. 37–52.

[7] Pierre Bourhis, Michel Leclère, Marie-Laure Mugnier, Sophie Tison, Federico Ulliana, and Lily Gallois. 2019. Oblivious and Semi-Oblivious Boundedness for Existential Rules. In IJCAI. 1581–1587.

[8] Marco Calautti, Georg Gottlob, and Andreas Pieris. 2015. Chase Termination for Guarded Existential Rules. In PODS. 91–103.

[9] Marco Calautti and Andreas Pieris. 2019. Oblivious Chase Termination: The Sticky Case. In ICDT. 17:1–17:18.

[10] Andrea Calì, Georg Gottlob, and Michael Kifer. 2013. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. J. Artif. Intell. Res. 48 (2013), 115–174.

[11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J. Autom. Reasoning 39, 3 (2007), 385–429.

[12] Alin Deutsch, Alan Nash, and Jeff B. Remmel. 2008. The Chase Revisited. In PODS. 149–158.

[13] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. Theor. Comput. Sci. 336, 1 (2005), 89–124.

[14] Tomasz Gogacz and Jerry Marcinkowski. 2014. All-Instances Termination of Chase is Undecidable. In ICALP. 293–304.

[15] Tomasz Gogacz, Jerry Marcinkowski, and Andreas Pieris. 2020. All-Instances Restricted Chase Termination. In PODS. 245–258.

[16] Tomasz Gogacz, Jerry Marcinkowski, and Andreas Pieris. 2020. All-Instances Restricted Chase Termination for Linear TGDs. Künstliche Intell. 34, 4 (2020), 465–473.

[17] Georg Gottlob, Marco Manna, and Andreas Pieris. 2014. Polynomial Combined Rewritings for Existential Rules. In KR.

[18] Georg Gottlob, Marco Manna, and Andreas Pieris. 2020. Multi-head Guarded Existential Rules Over Fixed Signatures. In KR. 445–454.

[19] Gösta Grahne and Adrian Onet. 2018. Anatomy of the Chase. Fundam. Inform. 157, 3 (2018), 221–270.

[20] Markus Krötzsch, Maximilian Marx, and Sebastian Rudolph. 2019. The Power of the Terminating Chase (Invited Talk). In ICDT. 3:1–3:17.

[21] Michel Leclère, Marie-Laure, Michael Thomazo, and Federico Ulliana. 2019. A Single Approach to Decide Chase Termination on Linear Existential Rules. In ICDT. 18:1–18:19.

[22] Bruno Marnette. 2009. Generalized schema-mappings: from termination to tractability. In PODS. 13–22.

[23] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. Journal on Data Semantics 10 (2008), 133–173.