THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

All-Instances Restricted Chase Termination

Citation for published version:
Gogacz, T, Marcinkowski, J & Pieris, A 2020, All-Instances Restricted Chase Termination. in *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 20).* Association for Computing Machinery (ACM), pp. 245-258, 2020 ACM SIGMOD/PODS International Conference on Management of Data, Portland, United States, 14/06/20. https://doi.org/10.1145/3375395.3387644

Digital Object Identifier (DOI):
10.1145/3375395.3387644

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 20)

OPEN ACCESS

All-Instances Restricted Chase Termination

Tomasz Gogacz
Institute of Informatics
University of Warsaw
t.gogacz@mimuw.edu.pl

Jerzy Marcinkowski
Institute of Computer Science
University of Wroclaw
jma@cs.uni.wroc.pl

Andreas Pieris
School of Informatics
University of Edinburgh
apieris@inf.ed.ac.uk

ABSTRACT

The chase procedure is a fundamental algorithmic tool in database theory with a variety of applications. A key problem concerning the chase procedure is all-instances termination: for a given set of tuple-generating dependencies (TGDs), is it the case that the chase terminates for every input database? In view of the fact that this problem is undecidable, it is natural to ask whether known well-behaved classes of TGDs ensure decidability. We consider here the main paradigms that led to robust TGD-based formalisms, that is, guardedness and stickiness. Although all-instances termination is well-understood for the oblivious version of the chase, the more subtle case of the restricted (a.k.a. the standard) chase is rather unexplored. We show that all-instances restricted chase termination for guarded and sticky single-head TGDs is decidable.

1 INTRODUCTION

The *chase procedure* (or simply chase) is a fundamental algorithmic tool that has been applied to several database problems such as computing data exchange solutions [13], and query answering and containment under constraints [1, 7], to name a few. The chase takes as input a database D and a set $\mathcal{T}$ of constraints – which, for this work, are tuple-generating dependencies (TGDs) of the form $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z}))$ with ϕ and ψ being conjunctions of atoms – and, if it terminates, its result is a finite instance $D_{\mathcal{T}}$ that is a *universal model* of D and $\mathcal{T}$, i.e., a model that can be homomorphically embedded into every other model of D and $\mathcal{T}$. This is the reason for the ubiquity of the chase as discussed in [11]. Indeed, many central database problems, which involve reasoning with TGDs, can be solved by simply exhibiting a universal model. And this is not only in theory. Despite the fact that the instance constructed by the chase can be very large, efficient implementations of the chase procedure have been successfully applied during the last few years in many different contexts [4, 19, 23, 24].

The Chase In a Nutshell. Roughly speaking, the chase adds new tuples to the database D (possibly involving null values that act as

witnesses for the existentially quantified variables), as dictated by the TGDs of $\mathcal{T}$, and it keeps doing this until all the TGDs of $\mathcal{T}$ are satisfied. There are, in principle, two different ways for formalizing this simple idea, which lead to different versions of the chase procedure. The first one, which gives rise to the *oblivious chase*, is as follows: for each pair $\bar{t}, \bar{u}$ of tuples of terms from the instance I constructed so far, trigger a TGD $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z}))$ if $\phi(\bar{t}, \bar{u}) \subseteq I$, and add to I the set of atoms $\psi(\bar{t}, \bar{v})$, where $\bar{v}$ is a tuple of new terms not occurring in I. The second way, which leads to the *restricted* (a.k.a. *standard*) chase, is a refinement of the above with the additional condition that, for a pair $\bar{t}, \bar{u}$ of tuples of terms, a TGD $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \, \psi(\bar{x}, \bar{z}))$ is triggered not only if $\phi(\bar{t}, \bar{u}) \subseteq I$, but also if there is no tuple $\bar{v}$ of terms from I such that $\psi(\bar{t}, \bar{v}) \subseteq I$, i.e., if the TGD is not already satisfied. Thus, the key difference between the oblivious and restricted versions of the chase is that the former triggers a TGD whenever the left-hand side of the implication is satisfied, while the latter triggers a TGD only if it is violated.

It should be clear that the restricted chase, in general, builds much smaller instances than the oblivious one. Actually, it is very easy to devise an example where, according to the restricted chase, none of the TGDs should be triggered, while the oblivious chase builds an infinite instance. Consider, e.g., the database $D = \{R(a, b)\}$ and the TGD $\forall x \forall y (R(x, y) \rightarrow \exists z \, R(x, z))$. The restricted chase will detect that the database already satisfies the TGD, while the oblivious chase will build the infinite instance $\{R(a, b), R(a, v_1), R(a, v_2), \ldots\}$, where $v_1, v_2, \ldots$ are (labeled) nulls. Consequently, the restricted chase has a clear advantage over the oblivious chase when it comes to the size of the result. But, of course, this advantage does not come for free: at each step, the restricted chase has to check that there is no way to satisfy the right-hand side of the TGD at hand, and this is costly. However, as it has been recently observed, the benefit from producing much smaller instances can justify the effort of checking whether a TGD is already satisfied; see, e.g., [4, 19].

1.1 The Challenge of Non-termination

As said above, there are nowadays efficient implementations of the restricted chase that allows us to solve central database problems by adopting a materialization-based approach [4, 19, 23, 24]. But, of course, for this to be feasible in practice we need a guarantee that the restricted chase terminates, which is not always the case. This fact motivated a long line of research on identifying fragments of TGDs that ensure the termination of the restricted chase, for every input database. A prime example is the class of *weakly-acyclic* TGDs [13], which is the standard language for data exchange purposes. A similar formalism, called *constraints with stratified-witness*, has been proposed in [12]. Many other sufficient conditions for the termination of the restricted chase can be found in the literature; see, e.g., [11, 12, 16, 18, 21, 22] – this list is by no means exhaustive, and we refer the reader to [17] for a comprehensive survey.

With so much effort spent on identifying sufficient conditions for the termination of the restricted chase, the question that comes up is whether a sufficient condition that is also *necessary* exists. In other words, given a set $\mathcal{T}$ of TGDs, is it possible to decide whether, for every database D, the restricted chase on D and $\mathcal{T}$ terminates? This has been addressed in [14], where it is shown that the answer is negative, even for the oblivious chase.

The undecidability proof in [14] constructs a sophisticated set of TGDs that goes beyond existing well-behaved classes of TGDs that enjoy certain syntactic properties, which in turn ensure favorable model-theoretic properties. Such well-behaved classes of TGDs have been proposed in the context of ontological reasoning. The two main paradigms that led to robust TGD-based formalisms, without forcing the restricted chase to terminate, are *guardedness* [2, 7, 8] and *stickiness* [9]. A TGD is guarded if the left-hand side of the implication, known as the body of the TGD, has an atom that contains (or "guards") all the universally quantified variables. On the other hand, sticky sets of TGDs are inherently unguarded, and their main goal is to express joins among relations that cannot be expressed via guarded TGDs (details are given in Section 2).

The fact that the set of TGDs given in the undecidability proof of [14] is far from being guarded or sticky brings us to the following question: is the restricted chase termination problem, as described above, decidable for guarded or sticky TGDs? This question is rather well-understood for the oblivious chase. In the case of guarded TGDs, the problem is 2EXPTIME-complete, and becomes PSPACE-complete for linear (one body-atom) TGDs [5]. The sticky case has been recently addressed in [6], where it is shown that the problem is PSPACE-complete. However, despite its clear advantage over the oblivious chase, we know very little about the restricted chase. It has been shown, independently of our work, that the problem is decidable for single-head (one atom in the head) linear TGDs [20]. However, nothing so far was known about guarded or sticky TGDs.

1.2 Research Challenges

We concentrate on guarded and sticky TGDs (in fact, single-head TGDs), and study the restricted chase termination problem. More precisely, we study the following: given a set $\mathcal{T}$ of single-head guarded or sticky TGDs, is it the case that for *every* database D, *every* restricted chase derivation of D w.r.t. $\mathcal{T}$ is finite? It might be the case that some derivations are finite and some others are not, depending on the order that TGDs are triggered, which is not the case for the oblivious chase. The reason for this non-deterministic behavior is the fact that the restricted chase applies a TGD only if it is necessary (recall the restricted vs. oblivious chase discussion above). On the other hand, the oblivious chase applies TGDs whenever the body is satisfied, which ensures a deterministic behavior. Our ultimate goal is to show that the problem in question is decidable. Towards this direction, one has to overcome a couple of non-trivial technical issues, which were not so difficult in the case of the oblivious chase.

Dealing with Fairness. The fairness condition is crucial in the definition of the chase in order to ensure that the result is indeed a model of the input database and set of TGDs. It states that each TGD that is violated at some point of the execution of the chase eventually will be satisfied. One of the main difficulties underlying our problem is to ensure fairness. In other words, focussing on the complement of our problem, it is not enough to simply check whether there exists a database that leads to an infinite derivation w.r.t. the set of TGDs, but we have to ensure that it is also fair.

As shown in [5], for the oblivious chase, the existence of a (possibly unfair) infinite chase derivation implies the existence of a fair one, which in turn implies that we can completely neglect the fairness condition. The question that comes up is whether we can establish the same for the restricted chase, which will crucially simplify our task. Actually, this question has been already posed by Jan Van den Bussche some years ago in a different context [10]. Showing such a result for the restricted chase is significantly more difficult than showing it for the oblivious chase. Note that the recent work [20], which considers the restricted chase, establishes such a result, but only for single-head linear TGDs. Generalizing this to single-head guarded or sticky TGDs, or ideally to arbitrary single-head TGDs, is a non-trivial task. As we shall see, here is the place where we need the TGDs to be single-head.

Existence of a Critical Database. It would be extremely useful to have a special database D^* in place, let us call it *critical*, of a very simple form, that ensures the following: given a set $\mathcal{T}$ of TGDs, if there is a database that leads to an infinite chase derivation w.r.t. $\mathcal{T}$, then already D^* does. With such a critical database in place, one can focus on the complement of our problem, and check whether D^* leads to an infinite chase derivation w.r.t. the given TGDs.

For the oblivious chase such a critical database exists: it simply collects all the atoms of the form $R(c, \ldots, c)$, where R is a relation that occurs in the given set of TGDs [21], and c an arbitrary constant. All the known decidability results about the oblivious chase heavily rely on the critical database D^* [5, 6]. It is an easy exercise, however, to show that D^*, as defined above, does not serve as a critical database in the case of the restricted chase. This brings us to the other technical challenge that we need to overcome, that is, the lack of an obvious database that can serve as a critical database. Let us say that [20], which considers the restricted chase, follows the critical database approach. However, it is easy to see that for single-head linear TGDs (the main concern of [20]) such a critical database is simply a database consisting of a single atom. This is far from being true for single-head guarded or sticky TGDs.

1.3 Summary of Contributions

Our main results (Theorem 5.1 and Theorem 6.1) state that, for a set $\mathcal{T}$ of single-head guarded or sticky TGDs, checking whether, for every database D, every restricted chase derivation of D w.r.t. $\mathcal{T}$ is finite, is decidable in elementary time. To show these results, we had to establish a series of auxiliary results, related to the technical challenges discussed above. Our main contributions follow:

In Section 4, we establish the *Fairness Theorem*, which essentially states that, for single-head (not necessarily guarded or sticky) TGDs, we can neglect the fairness condition. This overcomes the first challenge raised in the previous subsection. Let us stress that this result does *not* hold once we go beyond single-head TGDs, which means that our decision to focus on single-head TGDs is not for simplicity, but it might be crucial for the validity of our main results. This has been also observed, independently of our work, in [20].

In Section 5, we focus on guarded TGDs. We first characterize the existence of an infinite (possibly unfair) restricted chase derivation

of a database D w.r.t. a set $\mathcal{T}$ of single-head (not necessarily guarded) TGDs via the existence of an infinite subset S, called *chaseable*, of the instance $C_{D,\mathcal{T}}$ constructed by applying a variant of the oblivious chase on D using $\mathcal{T}$. Such a chaseable set S enjoys certain properties that allow us to convert it into an infinite restricted chase derivation of D w.r.t.$\mathcal{T}$. We then show that, for a set $\mathcal{T}$ of single-head guarded TGDs, the problem of deciding whether there is a database D such that an infinite chaseable subset of $C_{D,\mathcal{T}}$ exists can be reduced to the satisfiability problem of Monadic Second-Order Logic (MSOL) over infinite trees of bounded degree. The correctness of this reduction relies on another key result of independent interest: if there is a database that leads to a (possibly unfair) infinite chase derivation w.r.t. $\mathcal{T}$, then there is an *acyclic* one with the same property.

Finally, in Section 6, we concentrate on sticky TGDs. Given a set $\mathcal{T}$ of sticky TGDs, we reduce the problem of deciding whether there exists a database D such that an infinite (possibly unfair) restricted chase derivation of D w.r.t. $\mathcal{T}$ exists to the emptiness problem of deterministic Büchi automata. This reduction relies on another key result of independent interest: there exists a database D such that an infinite (possibly unfair) restricted chase derivation of D w.r.t. $\mathcal{T}$ exists iff a so-called *finitary caterpillar* for $\mathcal{T}$ exists. The latter is essentially an infinite "path-like" restricted chase derivation of some database w.r.t. $\mathcal{T}$, and is precisely the existence of such an object that we check via a deterministic Büchi automaton.

2 PRELIMINARIES

We consider the disjoint countably infinite sets **C**, **N**, and **V** of *constants*, *(labeled) nulls*, and *variables* (used in dependencies), respectively. We refer to constants, nulls and variables as *terms*. For an integer $n > 0$, we may 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 $\text{ar}(R)$ for n. A position of **S** is a pair (R, i), where $R/n \in \mathbf{S}$ and $i \in [n]$, that essentially identifies the i-th argument of R. An *atom* over **S** is an expression of the form $R(\bar{t})$, where $R/n \in \mathbf{S}$ and $\bar{t}$ is an n-tuple of terms. A *fact* is an atom whose arguments consist only of constants. We write $R(\bar{t})[i]$ for the term of $R(\bar{t})$ at position (R, i), i.e., the i-th element of $\bar{t}$. For brevity, we may refer to the position (R, i) in $R(\bar{t})$ simply as the i-th position of $R(\bar{t})$ and write $(R(\bar{t}), i)$. Moreover, for a variable x in $\bar{t}$, let $\text{pos}(R(\bar{t}), x) = \{(R, i) : R(\bar{t})[i] = x\}$, i.e., is the set of positions at which x occurs according to $R(\bar{t})$. An *instance* over **S** is a (possibly infinite) set of atoms over **S** that contain constants and nulls, while 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 all terms in I.

Substitutions and Homomorphisms. A *substitution* from a set of terms T to a set of terms T' is a function $h : T \to T'$ defined as follows: $\emptyset$ is a substitution, and if h is a substitution, then $h \cup \{t \mapsto t'\}$, where $t \in T$ and $t' \in T'$, is a substitution. The restriction of h to $S \subseteq T$ is denoted $h_{|S}$. A *homomorphism* from a set of atoms A to a set of atoms B is a substitution h from the terms of A to the terms of B such that (i) $t \in \mathbf{C}$ implies $h(t) = t$, and (ii) $R(t_1, \ldots, t_n) \in A$ implies $h(R(t_1, \ldots, t_n)) = R(h(t_1), \ldots, h(t_n)) \in B$.

Single-Head Tuple-Generating Dependencies. A *single-head tuple-generating dependency* σ is a constant-free first-order sentence

$\forall \bar{x} \forall \bar{y} \, (\phi(\bar{x}, \bar{y}) \to \exists \bar{z} \, R(\bar{x}, \bar{z}))$, where $\bar{x}, \bar{y}, \bar{z}$ are tuples of variables of **V**, $\phi(\bar{x}, \bar{y})$ is a conjunction of atoms, and $R(\bar{x}, \bar{z})$ is a single atom. For brevity, we write σ as $\phi(\bar{x}, \bar{y}) \to \exists \bar{z} \, R(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for joining atoms. We refer to $\phi(\bar{x}, \bar{y})$ and $R(\bar{x}, \bar{z})$ as the *body* and *head* of σ, denoted $\text{body}(\sigma)$ and $\text{head}(\sigma)$, respectively. Henceforth, we simply say tuple-generating dependency (TGD) instead of single-head TGD. The *frontier* of the TGD σ, 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 σ. The schema of a set $\mathcal{T}$ of TGDs, denoted $\text{sch}(\mathcal{T})$, is the set of predicates in $\mathcal{T}$, and we write $\text{ar}(\mathcal{T})$ for the maximum arity over all those predicates. An instance I satisfies a TGD σ as the one above, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\phi(\bar{x}, \bar{y})) \subseteq I$, then there exists $h' \supseteq h_{|\bar{x}}$ such that $h'(R(\bar{x}, \bar{z})) \in I$. Note that, by abuse of notation, we sometimes treat a tuple of variables as a set of variables, and a conjunction of atoms as a set of atoms. The instance I satisfies a set $\mathcal{T}$ of TGDs, written $I \models \mathcal{T}$, if $I \models \sigma$ for each $\sigma \in \mathcal{T}$.

Guardedness. A TGD σ is *guarded* if there exists an atom α in its body that contains all the variables occurring in $\text{body}(\sigma)$ [7]. The atom α is the *guard* of σ. In case there are more than one atoms that can serve as the guard of σ, then we fix the left-most such atom in $\text{body}(\sigma)$ as the guard. We write $\text{guard}(\sigma)$ for the guard of σ. The class of guarded TGDs, denoted $\mathbb{G}$, is defined as the family of all possible finite sets of guarded single-head TGDs.

Stickiness. The goal of stickiness is to capture joins that are not expressible via guarded TGDs [9]. The key property is that variables occurring more than once in the body of a TGD should be inductively propagated (or "stick") to the head-atom as follows
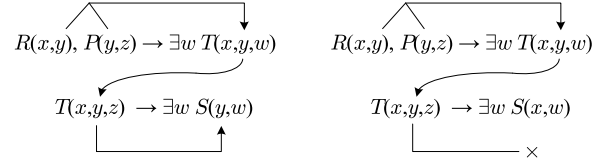
where the first set of TGDs is sticky, while the second is not. The formal definition is based on an inductive procedure that marks the variables that may violate the above property. Roughly, the base step marks a body-variable that does occur in the head. Then, the marking is inductively propagated from head to body as follows
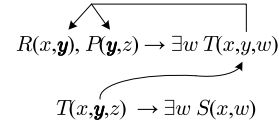
Stickiness requires every marked variable to appear only once in the body of a TGD. The formal definition follows.

Consider a set $\mathcal{T}$ of single-head TGDs; we assume, w.l.o.g., that the TGDs in $\mathcal{T}$ do not share variables. Let $\sigma \in \mathcal{T}$ and x a variable in $\text{body}(\sigma)$. We inductively define when x *is marked in* $\mathcal{T}$:

(1) if x does not occur in $\text{head}(\sigma)$, then x is marked in $\mathcal{T}$, and
(2) assuming that $\text{head}(\sigma) = R(\bar{t})$ and $x \in \bar{t}$, if there is $\sigma' \in \mathcal{T}$ with $R(\bar{t}')$ in its body, and each variable in $R(\bar{t}')$ at a position of $\text{pos}(R(\bar{t}), x)$ is marked in $\mathcal{T}$, then x is marked in $\mathcal{T}$.

The set $\mathcal{T}$ is *sticky* if there is no TGD with two occurrences of a variable that is marked in $\mathcal{T}$. Let $\mathbb{S}$ be the corresponding class.

3 THE CHASE PROCEDURE

The chase procedure accepts as input a database D and a set $\mathcal{T}$ of TGDs, and constructs an instance that contains D and satisfies $\mathcal{T}$. Central notions in this context are the notion of trigger, and the notion of trigger application (see, e.g., [15]).

Definition 3.1. A *trigger* for a set $\mathcal{T}$ of TGDs on an instance I is a pair (σ, h), where $\sigma \in \mathcal{T}$ and h is a homomorphism from $\mathrm{body}(\sigma)$ to I. We call (σ, h) *active* if there is no extension h' of $h_{|\mathrm{fr}(\sigma)}$ such that $h'(\mathrm{head}(\sigma)) \in I$. We denote by $\mathrm{result}(\sigma, h)$ the atom $v(\mathrm{head}(\sigma))$, where v is a mapping from the variables of $\mathrm{head}(\sigma)$ to $\mathbf{N}$ defined as

$$v(x) = \begin{cases} h(x) & \text{if } x \in \mathrm{fr}(\sigma), \\[2ex] c_{\sigma,h}^{x} & \text{otherwise.} \end{cases}$$

An *application* of (σ, h) to I returns the instance

$$J = I \cup \{\mathrm{result}(\sigma, h)\},$$

and such an application is denoted as $I\langle \sigma, h \rangle J$. ∎

In the definition of $\mathrm{result}(\sigma, h)$, each existentially quantified variable x occurring in $\mathrm{head}(\sigma)$ is mapped by v to a "fresh" null value of $\mathbf{N}$ whose name is uniquely determined by the trigger (σ, h) and x itself. Thus, given a trigger (σ, h), we can unambiguously write down the atom $\mathrm{result}(\sigma, h)$. In our analysis, it would be useful to be able to refer to the terms in $\mathrm{result}(\sigma, h)$ that have been propagated (not invented) during the application of (σ, h). Formally, the *frontier* of $\mathrm{result}(\sigma, h)$, denoted $\mathrm{fr}(\mathrm{result}(\sigma, h))$, are the terms of $\mathrm{result}(\sigma, h)$ that occur at the positions of $\bigcup_{x \in \mathrm{fr}(\sigma)} \mathrm{pos}(\mathrm{head}(\sigma), x)$.

3.1 The Real Oblivious Chase

Although this work is about the termination of the restricted chase, we use a variant of the oblivious chase, which we introduce below, as an auxiliary tool. The oblivious chase of a database D w.r.t. a set $\mathcal{T}$ of TGDs is essentially the $\subseteq$-minimal instance $I_{D,\mathcal{T}}$ that contains D and is closed under trigger applications, i.e., for every trigger (σ, h) for $\mathcal{T}$ on $I_{D,\mathcal{T}}$, $\mathrm{result}(\sigma, h) \in I_{D,\mathcal{T}}$. It is well-known that it can be realized by starting from the database D, and applying (active or non-active) triggers, which have not been applied before, for the given set $\mathcal{T}$ of TGDs on the instance constructed so far, and keep doing this until a fixpoint is reached. It is also well-known that $I_{D,\mathcal{T}}$ is unique since it does not depend on the order in which we apply the triggers; for more details see, e.g., [6, 15].

Our intention is to use the (unique) oblivious chase of D w.r.t. $\mathcal{T}$ as a predefined instance in which all the restricted chase derivations live (the formal definition of the restricted chase is given below). Thus, our task will be essentially to search in this instance for an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$. To this end, we need the *parent relation* over the oblivious chase, which essentially gives us the atoms that were involved in the trigger application that produced a certain atom. However, as the following simple example shows, this relation is, in general, not unique:

Example 3.2. Consider the set $\mathcal{T}$ of TGDs consisting of:

$$\sigma_1 : P(x, y) \rightarrow R(x, y) \qquad \sigma_3 : R(x, y) \rightarrow S(x)$$
$$\sigma_2 : P(x, y) \rightarrow S(x) \qquad \sigma_4 : S(x) \rightarrow \exists y\, R(x, y).$$

The oblivious chase of $D = \{P(a, b)\}$ w.r.t. $\mathcal{T}$ is the instance

$$\{P(a, b), R(a, b), S(a), R(a, c)\},$$

where c is a null. However, its atoms could have been produced in different ways: by applying $\sigma_1, \sigma_2, \sigma_4$, or by applying $\sigma_1, \sigma_3, \sigma_4$. In the first case, the parent of $S(a)$ is $P(a, b)$, while in the second case, is the atom $R(a, b)$. Thus, although the oblivious chase is unique, its ambiguous which atom is the parent of $S(a)$. ∎

As the above example illustrates, if we want to know in an unambiguous way who are the parents of a certain atom by simply inspecting the oblivious chase, we need to rely on a more refined structure. This is the purpose of the so-called real oblivious chase.

Definition 3.3. The *real oblivious chase* of a database D w.r.t. a set $\mathcal{T}$ of TGDs is the smallest labeled directed graph $\mathrm{ochase}(D, \mathcal{T}) = \langle V, \prec_p, \lambda, \tau \rangle$, where λ and τ assign atoms over $\mathrm{sch}(\mathcal{T})$ and TGD-mapping pairs (including the empty pair $\perp$) to nodes, such that:

- For each atom $\alpha \in D$, there is a node $v \in V$ with $\lambda(v) = \alpha$, $\tau(v) = \perp$, and, for each $u \prec_p w$, $w \neq v$.
- For each TGD $\sigma \in \mathcal{T}$, with $\mathrm{body}(\sigma) = \{\gamma_1, \ldots, \gamma_m\}$, for each mapping h from the variables in $\mathrm{body}(\sigma)$ to $\mathbf{C} \cup \mathbf{N}$, and for each $(v_1, \ldots v_m) \in V^m$, if $h(\gamma_1) = \lambda(v_1), \ldots, h(\gamma_m) = \lambda(v_m)$, then there exists $v \in V$ such that $v_1 \prec_p v, \ldots, v_m \prec_p v$, $\lambda(v) = \mathrm{result}(\sigma, h)$, and $\tau(v) = (\sigma, h)$.

The elements of $\{\lambda(v) : v \in V\}$ are the atoms of $\mathrm{ochase}(D, \mathcal{T})$, and the relation $\prec_p$ is the *parent relation* of $\mathrm{ochase}(D, \mathcal{T})$. ∎

Here is a simple example that illustrates the real oblivious chase:

Example 3.4. Let D and $\mathcal{T}$ be the database and the set of TGDs from Example 3.2. Then, $\mathrm{ochase}(D, \mathcal{T})$ is the following directed graph (for clarity, the homomorphisms are omitted)
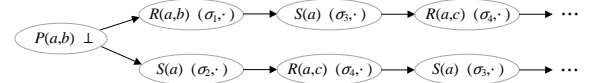
where c is the null determined by the trigger $(\sigma_4, \{x \mapsto a\})$. ∎

Strictly speaking, $\prec_p$ is a relation over the node set of the real oblivious chase of D w.r.t. $\mathcal{T}$. However, for notational convenience, from now on we will usually identify $\mathrm{ochase}(D, \mathcal{T})$ with its atoms, which clearly form a *multiset*, and we will see $\prec_p$ as a relation over this multiset of atoms. Let us also clarify that, although the real oblivious chase may generate several copies of the same atom, it will never produce an atom that is not generated by the oblivious chase, i.e., the oblivious chase coincides with the set consisting of the atoms of the real oblivious chase. The advantage of the real oblivious chase is that it provides a unique multiset instance where all the different restricted chase derivations live, and at the same time we can unambiguously refer to the parents of a certain atom.

REMARK. The name "real oblivious" reflects the fact that an atom is generated and added to the instance under construction even if its already present. On the other hand, the oblivious chase, since it builds a set (not a multiset) of atoms, it implicitly checks, before applying a trigger (σ, h), whether the atom $\mathrm{result}(\sigma, h)$ is already present. This somehow tells us that what we normally call oblivious chase is not completely oblivious, unlike the real oblivious one, which generates an atom no matter if it has been generated before.

Stop Relation. Before we proceed further, let us introduce one more basic relation, in addition to the parent relation, which will be heavily used throughout the paper. This is the "stop" relation $\prec_s$ over $\text{ochase}(D, \mathcal{T})$. Intuitively, $\alpha \prec_s \beta$ means that in the presence of α the atom β is superfluous in the sense that the trigger (σ, h) for $\mathcal{T}$ on an instance that contains α, with $\beta = \text{result}(\sigma, h)$, is not active due to the presence of α. Formally, given two vertices v, u of $\text{ochase}(D, \mathcal{T})$ such that $\tau(u) = (\sigma, h)$, we say that $\lambda(v)$ *stops* $\lambda(u)$, denoted $\lambda(v) \prec_s \lambda(u)$, if there exists a homomorphism h' such that (i) $h'(\lambda(u)) = \lambda(v)$, and (ii) $h'(h(x)) = h(x)$ for every $x \in \text{fr}(\sigma)$. Notice that two copies of the same atom in the real oblivious chase always stop each other. It is also easy to verify that the following holds, which relates the notion of active trigger with $\prec_s$:

Fact 3.5. *Let $I \subseteq \text{ochase}(D, \mathcal{T})$, and (σ, h) a trigger for $\mathcal{T}$ on I. Then, (σ, h) is active iff there is no $\alpha \in I$ such that $\alpha \prec_s \text{result}(\sigma, h)$.*

3.2 The Restricted Chase

We now come to the main object of our study, that is, the restricted (a.k.a. standard) chase. Similarly to the oblivious chase, the main idea of the restricted chase is, starting from a database D, to apply triggers for the given set $\mathcal{T}$ of TGDs on the instance constructed so far, and keep doing this until a fixpoint is reached. However, unlike the oblivious chase, it only applies active triggers. This is formalized as follows. Consider a database D and a set $\mathcal{T}$ of TGDs. We distinguish the two cases where the chase is terminating or not:

- A finite sequence $(I_i)_{0 \leq i \leq n}$ of instances, with $D = I_0$ and $n \geq 0$, is a *restricted chase derivation* of D w.r.t. $\mathcal{T}$ if: for each $0 \leq i < n$, there is an active trigger (σ, h) for $\mathcal{T}$ on I_i with $I_i \langle \sigma, h \rangle I_{i+1}$, and there is no active trigger (σ, h) for $\mathcal{T}$ on I_n.
- An infinite sequence $(I_i)_{i \geq 0}$ of instances, with $D = I_0$, is a *restricted chase derivation* of D w.r.t. $\mathcal{T}$ if, for each $i \geq 0$, there exists an active trigger (σ, h) for $\mathcal{T}$ on I_i such that $I_i \langle \sigma, h \rangle I_{i+1}$. Moreover, $(I_i)_{i \geq 0}$ is called *fair* if, for each $i \geq 0$, and every active trigger (σ, h) for $\mathcal{T}$ on I_i, there exists $j > i$ such that (σ, h) is a non-active trigger for $\mathcal{T}$ on I_j. Notice that in a fair derivation all the active triggers will eventually be deactivated, which is not true for unfair derivations.

A restricted chase derivation is called *valid* if it is finite, or infinite and fair. Infinite but unfair restricted chase derivations are not valid since they do not serve the main purpose of the chase procedure, i.e., build an instance that satisfies the given set of TGDs.

Restricted Chase Termination Problem

It is well-known that even for simple databases and sets of TGDs, we may have infinite chase derivations. The key question is, given a set $\mathcal{T}$ of TGDs, can we check whether, for every database D, every valid chase derivation of D w.r.t. $\mathcal{T}$ is finite? Before formalizing this problem, let us recall a central class of TGDs:

$$\mathbb{CT}_{\forall\forall}^{\text{res}} = \left\{ \mathcal{T} : \begin{array}{l} \text{for } \textit{every } \text{database } D, \\ \textit{every } \text{valid restricted chase derivation} \\ \text{of } D \text{ w.r.t. } \mathcal{T} \text{ is finite.} \end{array} \right\}$$

The superscript res in $\mathbb{CT}_{\forall\forall}^{\text{res}}$ indicates that we concentrate on restricted chase derivations. The main problem tackled in this work is defined as follows, where $\mathbb{C}$ is a class of TGDs:

PROBLEM :	$\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{C})$
INPUT :	A set $\mathcal{T} \in \mathbb{C}$ of TGDs.
QUESTION :	Is it the case that $\mathcal{T} \in \mathbb{CT}_{\forall\forall}^{\text{res}}$?

The above decision problem is, in general, undecidable. In fact, assuming that $\mathbb{TGD}$ is the class of arbitrary (single-head) TGDs:

Theorem 3.6 ([14]). $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{TGD})$ *is undecidable, even if we focus on binary and ternary predicates.*

But what about $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{G})$ and $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{S})$? These are non-trivial problems, and showing that are decidable is our main contribution.

4 THE FAIRNESS THEOREM

As one might expect, to establish the decidability of the problem $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{C})$, for $\mathbb{C} \in \{\mathbb{G}, \mathbb{S}\}$, we focus on its complement and show that, for a set $\mathcal{T} \in \mathbb{C}$ of TGDs, we can decide whether there is a database D such that there exists a fair infinite chase derivation of D w.r.t. $\mathcal{T}$. However, as observed in [5], where the same problem for the simpler case of the oblivious chase is studied, one of the main difficulties is to ensure fairness. For the oblivious chase, the existence of an (unfair) infinite chase derivation of D w.r.t. $\mathcal{T}$ implies the existence of a fair one [5]. Does the same hold for the restricted chase? This is a non-trivial question that is affirmatively answered by the following result dubbed *Fairness Theorem*:

Theorem 4.1 (**Fairness**). *Consider a database D and a set $\mathcal{T}$ of single-head TGDs. If there exists an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$, then there exists a fair one.*

Note that, to our surprise, the above theorem does *not* hold for multi-head TGDs, i.e., TGDs where the head is an arbitrary conjunction of atoms; a counterexample can be found in the appendix.[1] This reveals the subtlety of the restricted chase, and explains that our decision to focus on single-head TGDs is not just for simplicity, but it is crucial for our results. The decidability status of $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{G})$ and $\text{CT}_{\forall\exists}^{\text{res}}(\mathbb{S})$ for multi-head TGDs are challenging open problems.

We now proceed to show the Fairness Theorem. By hypothesis, there exists an infinite restricted chase derivation $(I_i)_{i \geq 0}$ of D w.r.t. $\mathcal{T}$. By exploiting $(I_i)_{i \geq 0}$, we are going to construct an infinite sequence $s_{D, \mathcal{T}} = ((I_i^j)_{i \geq 0})_{j \geq 0}$ of chase derivations of D w.r.t. $\mathcal{T}$ such that $(I_i^i)_{i \geq 0}$ is fair. In other words, $s_{D, \mathcal{T}}$ can be seen as an infinite matrix M, where the j-th row is the chase derivation $(I_i^j)_{i \geq 0}$, while the diagonal is a fair chase derivation of D w.r.t. $\mathcal{T}$.

The Diagonal Property. We start by first exposing a crucial property that $s_{D, \mathcal{T}}$ should enjoy:

Definition 4.2. A sequence $((J_i^j)_{i \geq 0})_{j \geq 0}$ of infinite restricted chase derivations of D w.r.t. $\mathcal{T}$ enjoys the *diagonal property* if, for each $i, j, k \geq 0$, $i \leq j$ and $i \leq k$ implies that $J_i^j = J_i^k$. ∎

In other words, by saying that the sequence $s_{D, \mathcal{T}}$ enjoys the diagonal property, we simply mean that on the i-th column of the matrix M, all instances below the diagonal element I_i^i coincide with I_i^i (hence the name diagonal property). This allows us to show that the diagonal gives rise to an infinite chase derivation of D w.r.t. $\mathcal{T}$:

---

[1] This has been also observed, independently of our work, in the recent paper [20] that concentrates on single-head linear TGDs.

LEMMA 4.3. *Consider a sequence $((J_i^j)_{i \geq 0})_{j \geq 0}$ of infinite restricted chase derivations of D w.r.t. $\mathcal{T}$ that enjoys the diagonal property. Then, $(J_i^i)_{i \geq 0}$ is a restricted chase derivation of D w.r.t. $\mathcal{T}$*

Of course, the diagonal property alone does not guarantee that the chase derivation $(I_i^i)_{i \geq 0}$ is fair. Thus, our main task is to construct $s_{D,\mathcal{T}} = ((I_i^j)_{i \geq 0})_{j \geq 0}$ in such a way that (i) it enjoys the diagonal property, and (ii) $(I_i^i)_{i \geq 0}$ is a fair chase derivation.

The Construction of $s_{D,\mathcal{T}}$. The high-level idea is as follows. The sequence $(I_i^0)_{i \geq 0}$ is defined as $(I_i)_{i \geq 0}$, which exists by hypothesis. Now, our intention is to obtain $(I_i^{n+1})_{i \geq 0}$ from $(I_i^n)_{i \geq 0}$. To this end, we carefully choose a large enough index $\ell > 0$ and (i) we define $(I_i^{n+1})_{0 \leq i \leq \ell}$ as $(I_i^n)_{0 \leq i \leq \ell}$, i.e., by simply copying the first $\ell + 1$ instances of $(I_i^n)_{i \geq 0}$, (ii) we obtain $I_{\ell+1}^{n+1}$ from $I_\ell^{n+1} = I_\ell^n$ by deactivating one of the early active triggers due to which $(I_i^n)_{i \geq 0}$ is not fair, and (iii) we obtain $(I_i^{n+1})_{i \geq \ell+2}$ by mimicking $(I_i^n)_{i \geq \ell+1}$. The formal construction of $s_{D,\mathcal{T}}$ follows.

As said above, $(I_i^0)_{i \geq 0}$ is defined as $(I_i)_{i \geq 0}$. Assume now that $(I_i^n)_{i \geq 0}$ has been defined for some $n \geq 0$. We are going to define $(I_i^{n+1})_{i \geq 0}$. Let $m \geq 0$ be the smallest index such that there exists an active trigger (σ, h) for $\mathcal{T}$ on I_m^n that remains active forever in $(I_i^n)_{i \geq 0}$. (Notice that if such an $m \geq 0$ does not exist, then $(I_i^n)_{i \geq 0}$ is fair and we are done.) Assume that I_{i+1}^n is obtained from I_i^n via the trigger (σ_i, h_i). Let $A = \{i \geq 0 : \text{result}(\sigma, h) \prec_s \text{result}(\sigma_i, h_i)\}$. By exploiting the properties of $\prec_s$, it is not difficult to show that:

LEMMA 4.4. *The set A is finite.*

Let ℓ be an integer greater than all the elements of $\{n, m\} \cup A$, which exists by Lemma 4.4. We define:

$$I_i^{n+1} = \begin{cases} I_i^n & 0 \leq i \leq \ell \\ \\ I_{i-1}^n \cup \{\text{result}(\sigma, h)\} & i > \ell \end{cases}$$

We can show the following; the proof is in the appendix:

LEMMA 4.5. *$(I_i^{n+1})_{i \geq 0}$ is a restricted chase derivation of D w.r.t. $\mathcal{T}$.*

Finalizing the Proof. Lemma 4.5 implies that indeed $s_{D,\mathcal{T}} = ((I_i^j)_{i \geq 0})_{j \geq 0}$ is an infinite sequence of chase derivations of D w.r.t. $\mathcal{T}$. The fact that in the definition of $(I_i^{n+1})_{i \geq 0}$ above we choose the integer ℓ to be greater than n ensures that $s_{D,\mathcal{T}}$ enjoys the diagonal property. Therefore, by Lemma 4.3, we conclude that $(I_i^i)_{i \geq 0}$ is an infinite chase derivation of D w.r.t. $\mathcal{T}$. Moreover, since there are only finitely many active triggers for $\mathcal{T}$ on an instance I_i^j since I_i^j is finite, it follows from the construction of $s_{D,\mathcal{T}}$ that $(I_i^i)_{i \geq 0}$ is fair. Hence, $(I_i^i)_{i \geq 0}$ is a fair infinite chase derivation of D w.r.t. $\mathcal{T}$.

5 CHASE TERMINATION & GUARDEDNESS

We now concentrate on guarded TGDs, and show that:

THEOREM 5.1. *$\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{G})$ is decidable in elementary time.*

By Theorem 4.1, to establish the above result it suffices to show that, for a set $\mathcal{T} \in \mathbb{G}$ of TGDs, we can decide in elementary time whether there is a database D such that there exists an infinite (possibly unfair) restricted chase derivation of D w.r.t. $\mathcal{T}$. To this end, we first characterize the existence of an infinite (possibly unfair) restricted chase derivation of D w.r.t. $\mathcal{T}$ via the existence of

an infinite subset of $\text{ochase}(D, \mathcal{T})$, called chaseable, that enjoys certain properties. We then show that the problem of deciding whether there is a database D such that an infinite chaseable subset of $\text{ochase}(D, \mathcal{T})$ exists can be reduced to the satisfiability problem of Monadic Second-Order Logic (MSOL) over infinite trees of bounded degree, which in turn implies that $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{G})$ is decidable. At first glance, such a reduction looks unfeasible since the above statement talks about arbitrary databases D, and thus $\text{ochase}(D, \mathcal{T})$ can be structurally very complex, i.e., not close to a tree. Nevertheless, we can show that it suffices to concentrate on acyclic databases D, which in turn implies (due to the fact that we consider single-head guarded TGDs) that $\text{ochase}(D, \mathcal{T})$ is acyclic.

5.1 Non-Termination via Chaseable Sets

We proceed to introduce the notion of chaseable set for a database D and a set $\mathcal{T}$ of TGDs. The key idea is to isolate certain properties of an infinite subset of $\text{ochase}(D, \mathcal{T})$ that allow us to convert it into an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$. To this end, we need the "before" relation $\prec_b$ over $\text{ochase}(D, \mathcal{T})$. Intuitively, $\alpha \prec_b \beta$ means that, if the atoms α and β have been generated by some restricted chase derivation δ, then necessarily α has been generated before β; otherwise, δ is not a restricted chase derivation. Given a sequence of instances $I_0, I_1, \ldots$, where each I_i is a subset of $\text{ochase}(D, \mathcal{T})$, there are essentially three reasons why it is not, or it cannot be converted (by merging some of the initial instances) into a restricted chase derivation of D w.r.t. $\mathcal{T}$: there are atoms $\alpha \in I_i \setminus I_{i-1}$ and $\beta \in I_j \setminus I_{j-1}$ such that:

(1) $\alpha \in D$, $\beta \notin D$ and $j < i$, i.e., α is generated after β.
(2) $\alpha \prec_p \beta$ but $j < i$, i.e., the parent of β is generated after β.
(3) $\alpha \prec_s \beta$ but $i < j$, i.e., β is generated after α, while α stops β.

The goal of the relation $\prec_b$ is to ensure that none of the above holds. Having the parent relation $\prec_p$, and the stop relation $\prec_s$ (together with Fact 3.5), it should be clear that the before relation $\prec_b$ is

$$\{\langle \alpha, \beta \rangle : \alpha \in D \text{ and } \beta \in \text{ochase}(D, \mathcal{T}) \setminus D\} \cup \prec_p \cup \prec_s^{-1},$$

where $\prec_s^{-1}$ refers to the inverse relation of $\prec_s$. We write $\prec_b^+$ for the transitive closure of $\prec_b$. The notion of chaseable set follows.

Definition 5.2. Consider a database D, and a set $\mathcal{T}$ of TGDs. A set $A \subseteq \text{ochase}(D, \mathcal{T})$ is called *chaseable* if the following hold:

(1) For each $\alpha \in A$, the set $\{\beta \in A : \beta \prec_b^+ \alpha\}$ is finite.
(2) For each $\alpha \in A$ and $\beta \in \text{ochase}(D, \mathcal{T})$, $\beta \prec_p \alpha$ implies $\beta \in A$.
(3) $\{\langle \alpha, \beta \rangle : \alpha, \beta \in A \text{ and } \alpha \prec_b \beta\}$ is a directed acyclic graph, i.e., there are no cycles in the relation $\prec_b$ over A. ∎

The first condition states that, for each $\alpha \in A$, only finitely many atoms of A should come before α. The second condition says that the parent of an atom $\alpha \in A$ should be in A. Finally, the third condition states that, for every pair of distinct atoms $\alpha, \beta \in A$, either α should come before β, or β should come before α. It is not difficult to show that indeed the existence of an infinite chaseable set characterizes the existence of an infinite restricted chase derivation.

THEOREM 5.3. *Consider a database D and a set $\mathcal{T}$ of TGDs. The following are equivalent:*

(1) There exists an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$.
(2) There exists an infinite set $A \subseteq \text{ochase}(D, \mathcal{T})$ that is chaseable.

Let us clarify that Theorem 5.3 holds for arbitrary, not necessarily guarded TGDs. The importance of guardedness is revealed in the next section, where we show that we can focus on acyclic databases.

5.2 The Treeification Theorem

We first need to recall the standard notion of acyclicity for instances. Intuitively, an instance I is acyclic if its atoms can be rearranged in a tree T in such a way that, for each term $t \in \mathrm{dom}(I)$, the set of atoms that mention t induces a connected subtree of T.

Definition 5.4. A *join tree* of an instance I is a pair (T, λ), where $T = (V, E)$ is a tree, and λ is the labeling function $V \to I$, such that:

(1) For each $\alpha \in I$, there exists $v \in V$ with $\lambda(v) = \alpha$.
(2) For each term $t \in \mathrm{dom}(I)$, the set $\{v \in V : t \text{ occurs in } \lambda(v)\}$ induces a connected subtree of T.

We say that I is *acyclic* if it admits a joint tree. ∎

We then show the following result dubbed *Treeification Theorem*:

Theorem 5.5 (**Treeification**). *Let $\mathcal{T} \in \mathbb{G}$. If there exists a database D such that there is an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$, then there is an acyclic database with the same property.*

This is a rather involved result and its proof can be found in the appendix. In what follows, we give the high-level idea underlying the construction of the desired acyclic database. By hypothesis, there exists an infinite restricted chase derivation $(I_i)_{i \geq 0}$ of some database D w.r.t. $\mathcal{T}$. From $\mathrm{ochase}(D, \mathcal{T}) = (V, \prec_p, \lambda, \tau)$ we can naturally obtain the *guard-parent* (resp., *side-parent*) relation $\prec_{gp}$ (resp., $\prec_{sp}$) over V as the subrelation of $\prec_p$ by keeping only the pairs of nodes (v, u) where v corresponds to the guard atom (resp., to a side atom, i.e., an atom other than the guard) of the TGD in $\tau(u)$. Let $\prec_{gp}^+$ be the transitive closure of $\prec_{gp}$. Observe that, due to guardedness, $\mathrm{ochase}(D, \mathcal{T})$ can be seen as a forest w.r.t. $\prec_{gp}$, where the nodes of V labeled with atoms of D are the roots of the trees, and all the other nodes are the non-root nodes. As with $\prec_p$, for convenience, we will usually see $\prec_{gp}$ and $\prec_{sp}$ as relations over the multiset consisting of the atoms of $\mathrm{ochase}(D, \mathcal{T})$.

Let $\mathcal{I} = \bigcup_{i \geq 0} I_i$. For an atom $\beta \in \mathcal{I}$, we define $\mathcal{I}_\beta$ as the set $\{\alpha \in \mathcal{I} : \beta \prec_{gp}^+ \alpha\}$. Since D is finite, while $\mathcal{I}$ is infinite, we can conclude that there exists an atom $\alpha^\infty \in D$ such that the set $\mathcal{I}_{\alpha^\infty}$ is infinite. At this point, one may think that the desired acyclic database consists of the atom α^∞ together with the atoms of D that can serve as its side atoms, i.e., the database

$$\{\alpha^\infty\} \cup \{R(t_1, \ldots, t_n) \in D : t_1, \ldots, t_n \text{ occur in } \alpha^\infty\}.$$

Unfortunately, as shown below, this is not the case:

Example 5.6. Assume that $\mathcal{T}$ consists of the TGDs

$$\sigma_1 \ : \ S(x, y) \to T(x)$$
$$\sigma_2 \ : \ R(x, y), T(y) \to P(x, y)$$
$$\sigma_3 \ : \ P(x, y) \to \exists z \, P(y, z).$$

It is clear that there exists an infinite restricted chase derivation of $\{R(a, b), S(b, c)\}$ w.r.t. $\mathcal{T}$: first apply σ_1 and obtain $T(b)$, then apply σ_2 and obtain $P(a, b)$, and then apply σ_3 infinitely many times. Observe that the key atom α^∞ is $R(a, b)$. However, there is no infinite restricted chase derivation of $\{R(a, b)\}$ w.r.t $\mathcal{T}$. In fact, there are no active triggers for $\mathcal{T}$ on $\{R(a, b)\}$ ∎

As it can be seen from the above example, the reason why α^∞, together with its potential side atoms from D, do not give rise to an infinite restricted chase derivation is the need of what we call here remote side-parents. In particular, referring to Example 5.6, we have an infinite restricted chase derivation of $\{R(a, b), S(b, c)\}$ w.r.t. $\mathcal{T}$ due to the atom $P(a, b)$, which has as a guard-parent the atom $\alpha^\infty = R(a, b)$, and as a side-parent the atom $T(b)$. However, $T(b)$ is not a database atom, but is obtained due to the database atom $S(b, c)$, which cannot serve as a side atom of $R(a, b)$. So, somehow, the atom $S(b, c)$ is a remote side-parent of $P(a, b)$. This situation can be formalized as follows.

Definition 5.7. Consider two distinct atoms $\alpha, \beta \in D$, and two atoms $\alpha', \beta' \in \mathcal{I}$. The tuple $\langle \alpha, \alpha', \beta, \beta' \rangle$ is a *remote-side-parent situation* if the following hold: $\alpha \prec_{gp}^+ \alpha'$, $\beta \prec_{gp}^+ \beta'$, and $\beta' \prec_{sp} \alpha'$. If this is the case, then we say that α *longs for* β. ∎

It is now not difficult to show that there exists a natural number ℓ_∞ such that, if $\langle \alpha^\infty, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation, then $\beta' \in I_{\ell_\infty}$. In fact, if $\langle \alpha^\infty, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation, then, due to guardedness, all the terms occurring in β' occur also in α^∞ and β. This implies that there are only finitely many pairs of atoms (β, β'), where $\beta \in D$ and $\beta' \in \mathcal{I}$, such that, for some $\alpha' \in \mathcal{I}$ with $\alpha^\infty \prec_{gp}^+ \alpha'$, $\langle \alpha^\infty, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation. The latter implies the existence of ℓ_∞ claimed above, which is crucial in the construction of the desired acyclic database. We can now give the intuition underlying this construction.

Our intention is to explicitly construct from D a join tree (T_{ac}, λ), where $T_{ac} = (V, E)$, and the desired acyclic database D_{ac} will be the set of atoms $\{\lambda(v) : v \in V\}$. Imagine D as a directed multigraph: the atoms of D are the vertices of this graph, while the edge-relation is "longs for". Now, T_{ac} is the set of all directed paths in this directed graph, starting from α^∞, of length at most ℓ_∞. There is a natural tree ordering on such a set of paths, and this is exactly the ordering E of T_{ac}. Every path is labelled with an isomorphic copy of the atom being its end-point, but in a particular way: if x and y are two vertices of T_{ac}, with $(x, y) \in E$, which means that x comes from some $\alpha \in D$ and y comes from some $\beta \in D$ such that α longs for β, then, if α, β share a term, then $\lambda(x)$ and $\lambda(y)$ share the respective terms. Thanks to that, we are able to show that (the offspring of) $\lambda(y)$ can offer to (the offspring of) $\lambda(x)$ the same service in D_{ac} as (the offspring of) β provides to (the offspring of) α in D.

5.3 Deciding $\mathrm{CT}_{\forall\forall}^{\mathrm{res}}(\mathbb{G})$ via MSOL

By Theorems 4.1, 5.3 and 5.5, given a set $\mathcal{T} \in \mathbb{G}$, deciding whether $\mathcal{T} \notin \mathbb{CT}_{\forall\forall}^{\mathrm{res}}$ is equivalent to the problem of checking whether there is an *acyclic* database D such that an infinite chaseable subset of $\mathrm{ochase}(D, \mathcal{T})$ exists. Our goal is to reduce the latter to the satisfiability problem of Monadic Second-Order Logic (MSOL) over infinite trees of bounded degree, which is decidable in k-ExpTime, where k is the number of quantifier alternations.

We need to devise an MSOL sentence $\phi_{\mathcal{T}}$ such that the following statements are equivalent:

(1) There is an acyclic database D such that an infinite chaseable subset of $\mathrm{ochase}(D, \mathcal{T})$ exists.
(2) $\phi_{\mathcal{T}}$ is satisfiable over $\Lambda_{\mathcal{T}}$-labeled infinite trees of bounded degree, where $\Lambda_{\mathcal{T}}$ is a finite alphabet that depends on $\mathcal{T}$.

Abstract Join Trees

Whenever $\mathcal{T}$ consists of single-head guarded TGDs and D is acyclic, then $\mathrm{ochase}(D, \mathcal{T})$ is also acyclic, which means that it has a join tree [3]. Thus, one may think that this join tree is a natural candidate for a tree that our MSOL formula could talk about. But this is not going to work for the simple reason that the codomain of the labeling function λ of such a join tree is infinite. We therefore need to invent something similar to a join tree, i.e., a structure that encodes an instance as a labeled tree, but much more parsimonious with respect to the labeling function. This is precisely the purpose of what we call abstract join trees.

We define the finite alphabet $\Lambda_\mathcal{T}$ as a set of triples

$$\Lambda_\mathcal{T} \;=\; \mathrm{sch}(\mathcal{T}) \;\times\; (\{F\} \cup \mathcal{T}) \;\times\; \mathcal{E}Q_\mathcal{T}$$

that encode atoms. Here is the idea underlying this encoding:

- The first element of each triple is a predicate; it simply tells us the predicate of the atom in question.
- Concerning the second element, F stands for "database fact", and indicates that the encoded atom is an atom from the original database. If an atom does not come from the database, then the second element of the triple tells us which TGD of $\mathcal{T}$ was used to generate it.
- Concerning the third element, we define $\mathcal{E}Q_\mathcal{T}$ as the set of all equivalence relations on $\{f, m\} \times \{1, 2, \ldots \mathrm{ar}(\mathcal{T})\}$, where f and m stand for "father" and "me". The idea is that, for example, the pair $[[m, i], [m, j]]$ says that the encoded atom has the same term at its i-th and j-th position, while the pair $[[m, i], [f, j]]$ says that the term at the i-th position in the atom in question equals to the term at the j-th position of its father (with respect to the tree relation).

In what follows, for brevity, given a node v that is labeled by the triple $\langle x, y, z \rangle$, we write $\mathrm{pr}(v)$ for the predicate x, $\mathrm{org}(v)$ for y, i.e., the origin of the encoded atom, and $\mathrm{eq}(v)$ for the equivalence relation z. Recall also that, for an atom α, we write $\alpha[i]$ for its i-th term. We are now ready to formally define abstract join trees.

Definition 5.8. An *abstract join tree* for a set $\mathcal{T} \in \mathbb{G}$ of TGDs is a (finite or infinite) $\Lambda_\mathcal{T}$-labeled rooted tree $T = \langle V, \prec \rangle$, of degree at most $\max\{\mathrm{ar}(\mathcal{T}), |\mathcal{T}|\}$, that satisfies the following conditions:

(1) The set $\{x \in V : \mathrm{org}(x) = F\}$ is non-empty but finite.
(2) If $x \prec y$ and $\mathrm{org}(y) = F$, then $\mathrm{org}(x) = F$.
(3) If $x \prec y$ and $\mathrm{org}(y) = \sigma$, then $\mathrm{pr}(x)$ is the predicate of $\mathrm{guard}(\sigma)$ and $\mathrm{pr}(y)$ is the predicate of $\mathrm{head}(\sigma)$.
(4) If $x \prec y$, then $[[m, i], [m, j]] \in \mathrm{eq}(x)$ iff $[[f, i], [f, j]] \in \mathrm{eq}(y)$.
(5) If $x \prec y$ and $\mathrm{org}(y) = \sigma$, for some $\sigma \in \mathcal{T}$ with $\alpha = \mathrm{guard}(\sigma)$ and $\beta = \mathrm{head}(\sigma)$, then:
 (a) $\alpha[i] = \beta[j]$ implies $[[f, i], [m, j]] \in \mathrm{eq}(y)$,
 (b) $\alpha[i] = \alpha[j]$ implies $[[f, i], [f, j]] \in \mathrm{eq}(y)$, and
 (c) if $\beta[j]$ is an existentially quantified variable in σ, then $[[m, i], [m, j]] \in \mathrm{eq}(y)$ iff $\beta[j] = \beta[i]$. ∎

We now need to explain how an abstract join tree is transformed into an instance. Consider an abstract join tree $T = \langle V, \prec \rangle$ for a set $\mathcal{T} \in \mathbb{G}$. We define $\mathrm{Eq}_T \subseteq (V \times \{1, \ldots, \mathrm{ar}(\mathcal{T})\}) \times (V \times \{1, \ldots, \mathrm{ar}(\mathcal{T})\})$ as the smallest equivalence relation such that, for every edge $x \prec y$, if $[[m, i], [m, j]] \in \mathrm{eq}(x)$ then $[[x, i], [x, j]] \in \mathrm{Eq}_T$, and if $[[f, i], [m, j]] \in \mathrm{eq}(y)$ then $[[x, i], [y, j]] \in \mathrm{Eq}_T$. The instance $\Delta(T)$

is defined as the set of atoms $\{\delta(x) : x \in V\}$, where (i) for each $x \in V$, the predicate of $\delta(x)$ is $\mathrm{pr}(x)$, and (ii) for each $x, y \in V$, $\delta(x)[i] = \delta(y)[j]$ iff $[[x, i], [y, j]] \in \mathrm{Eq}_T$.

For an abstract join tree T, we write $T_{|F}$ for the restriction of T to its nodes that are labeled with a label of the form $\langle \cdot, F, \cdot \rangle$. Then, it is not hard to see that the following equivalence holds:

LEMMA 5.9. *For a set $\mathcal{T} \in \mathbb{G}$, and an acyclic database D, the following are equivalent:*

(1) *There exists an infinite chaseable subset of $\mathrm{ochase}(D, \mathcal{T})$.*
(2) *There exists an abstract join tree T such that $\Delta(T_{|F})$ and D are isomorphic, and $\Delta(T)$ is an infinite and chaseable subset of $\mathrm{ochase}(\Delta(T_{|F}), \mathcal{T})$.*

Therefore, in order to prove Theorem 5.1, it is now enough to construct, for a given $\mathcal{T}$, an MSOL formula $\phi_\mathcal{T}$ such that, for any abstract join tree T, it holds that: $T \models \phi_\mathcal{T}$ iff $(\star)$ $\Delta(T)$ is an infinite and chaseable subset of $\mathrm{ochase}(\Delta(T_{|F}), \mathcal{T})$.

Chaseable Abstract Join Trees

Our MSOL formula $\phi_\mathcal{T}$ (under construction) is supposed to express some property of $\Delta(T)$, for a given abstract join tree T, namely the property $(\star)$. But, it does not see $\Delta(T)$. It can only talk about T. Moreover, talking about nodes, let say x and y, of T, and relations between these nodes, it must actually mean the atoms $\delta(x)$ and $\delta(y)$, and relations among those atoms. Thus, it will be convenient to have a language to talk about the nodes of T but to mean atoms of $\Delta(T)$. We now define such a language, allowing ourselves to slightly abuse the notation and overload the symbols $\prec_p$, $\prec_s$ and $\prec_b$.

First, we need a way to say that an atom $\alpha \in \Delta(T)$ can act as a side atom for some other atom $\beta \in \Delta(T)$, and also to specify which terms of β occur in α and at which positions. This can be achieved via the notion of sideatom type. A *sideatom type* π (w.r.t $\mathcal{T}$) is a triple $\langle P, m, \xi \rangle$, where $P/n \in \mathrm{sch}(\mathcal{T})$, $m \le \mathrm{ar}(\mathcal{T})$ is a natural number, called the arity of π, and $\xi : [n] \to [m]$. Given two atoms α and β, we say that α is a π-*sideatom* of β, denoted $\alpha \subseteq_\pi \beta$, if the predicate of α is P, the predicate of β has arity m, and $\alpha[i] = \beta(\xi(i))$ for each $i \in [n]$. For example, the atom $\alpha = P(a, b, c)$ is a π-sideatom of $\beta = R(a, d, c, b)$ with $\pi = \langle P, 4, \{1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 3\}\rangle$. In what follows, it would be convenient to represent a guarded body by directly using sideatom types. More precisely, for a guarded TGD σ, where $\mathrm{body}(\sigma) = \gamma, \gamma_1, \ldots, \gamma_m$ with $\gamma = \mathrm{guard}(\sigma)$, its body can be represented in the obvious way as $\gamma, \pi_1, \ldots \pi_m$, where $\pi_1, \ldots, \pi_m$ are sideatom types of arity equal to the arity of the predicate of γ.

Parent Relation. Consider an abstract join tree $T = \langle V, \prec \rangle$ for a set $\mathcal{T}$ of guarded TGDs. The parent relation is defined as follows:

- Given an edge $x \prec y$ in T, with $\mathrm{org}(y) = \sigma$, for some $\sigma \in \mathcal{T}$ such that $\mathrm{body}(\sigma) = \gamma, \pi_1, \ldots, \pi_m$, we say that a node $z \in V$ is a π_i-*side-parent* of y, denoted $z \prec_{sp}^{\pi_i} y$, if $\delta(z) \subseteq_{\pi_i} \delta(x)$.
- Given two nodes $x, y \in V$, x is a *parent* of y, denoted $x \prec_p y$, if $x \prec y$, or $x \prec_{sp}^\pi y$ for some sideatom type π.

Stop Relation. Consider two nodes $x, y \in V$, with $\mathrm{org}(y) = \sigma$. We say that x *stops* y, denoted $x \prec_s y$, if there exists a homomorphism h such that $h(\delta(y)) = \delta(x)$, and, for each term t in $\delta(y)$ that occurs at a position of $\bigcup_{x \in \mathrm{fr}(\sigma)} \mathrm{pos}(\mathrm{head}(\sigma), x)$, $h(t) = t$.

Before Relation. The before relation is defined as expected:

$$\prec_b = \{\langle x, y \rangle : x, y \in V, \mathrm{org}(x) = F \text{ and } \mathrm{org}(y) \neq F\} \cup \prec_p \cup \prec_s^{-1}.$$

We write $\prec_b^+$ for the transitive closure of $\prec_b$.

Having the above relations in place, we can now define the notion of chaseable abstract join tree:

Definition 5.10. Consider an abstract join tree $T = \langle V, \prec \rangle$ for a set $\mathcal{T} \in \mathbb{G}$. We say that T is *chaseable* if the following hold:

(1) For each $x \in V$, the set $\{y \in V : y \prec_b^+ x\}$ is finite.
(2) For each edge $x \prec y$, where $\mathrm{org}(y) = \sigma$ for some $\sigma \in \mathcal{T}$ with $\mathrm{body}(\sigma) = \gamma, \pi_1, \ldots, \pi_m$, there exists $z \in V$ such that $z \prec_{sp}^{\pi_i} y$ for each $i \in [m]$.
(3) $\{\langle x, y \rangle : x, y \in V \text{ and } x \prec_b y\}$ is a directed acyclic graph, i.e., there are no cycles in the relation $\prec_b$ over V. ∎

It follows, by construction, that for a set $\mathcal{T}$ of guarded TGDs, and an abstract join tree T for $\mathcal{T}$, $\Delta(T)$ is an infinite chaseable subset of $\mathrm{ochase}(\Delta(T_{|F}), \mathcal{T})$ iff there exists an infinite chaseable abstract join tree $\hat{T}$ for $\mathcal{T}$ such that $\Delta(T_{|F})$ is isomorphic to $\Delta(\hat{T}_{|F})$. Then:

LEMMA 5.11. *For a set $\mathcal{T} \in \mathbb{G}$, the following are equivalent:*

(1) *There exists an abstract join tree T for $\mathcal{T}$ such that $\Delta(T)$ is an infinite chaseable subset of $\mathrm{ochase}(\Delta(T_{|F}), \mathcal{T})$.*
(2) *There exists an infinite chaseable abstract join tree for $\mathcal{T}$.*

Chaseable Abstract Join Trees are MSOL-definable

The last task is to show the following:

LEMMA 5.12. *Consider a set $\mathcal{T} \in \mathbb{G}$. There is an MSOL sentence $\phi_{\mathcal{T}}$ such that, for a $\Lambda_{\mathcal{T}}$-labeled tree T of degree at most $\max\{\mathrm{ar}(\mathcal{T}), |\mathcal{T}|\}$, it holds that $T \models \phi_{\mathcal{T}}$ iff T is a chaseable abstract join tree for $\mathcal{T}$.*

The sentence $\phi_{\mathcal{T}}$ has to check whether a tree is an abstract join tree, and also whether the three conditions of Definition 5.10 are satisfied. Since, given an abstract join tree $T = \langle V, \prec \rangle$, for each term t in $\Delta(T)$, $\{x \in V : t \text{ occurs in } \delta(x)\}$ induces a connected subtree of T, it should be evident that indeed the above conditions can be checked via an MSOL sentence. More details concerning the MSOL sentence $\phi_{\mathcal{T}}$ can be found in the appendix.

Having Lemmas 5.9, 5.11 and 5.12, we get that $\mathrm{CT}_{\forall\forall}^{\mathrm{res}}(\mathbb{G})$ is decidable in elementary time, and Theorem 5.1 follows.

6 CHASE TERMINATION & STICKINESS

We now concentrate on sticky sets of TGDs, and show that:

THEOREM 6.1. $\mathrm{CT}_{\forall\forall}^{\mathrm{res}}(\mathbb{S})$ *is decidable in elementary time.*

As in the case of guarded TGDs, to establish the above result we are going to show that the complement of $\mathrm{CT}_{\forall\forall}^{\mathrm{res}}(\mathbb{S})$ is decidable in elementary time. In fact, our ultimate goal is to reduce the complement of $\mathrm{CT}_{\forall\forall}^{\mathrm{res}}(\mathbb{S})$ to the emptiness problem of deterministic Büchi automata, which is feasible in linear time in the size of the automaton. To this end, given a set $\mathcal{T} \in \mathbb{S}$, we characterize the existence of a database D such that there is an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$ via the existence of a finitary caterpillar for $\mathcal{T}$. The latter is essentially an infinite "path-like" restricted chase derivation of some database w.r.t. $\mathcal{T}$, and, as we shall see, its existence can be checked via a deterministic Büchi automaton.

6.1 Non-Termination via Caterpillars

To formally introduce the notion of finitary caterpillar, we first need the notion of proto-caterpillar.

Definition 6.2. Consider a set $\mathcal{T}$ of TGDs. A *proto-caterpillar for* $\mathcal{T}$ is a tuple $\diamond = (L^{\diamond}, B^{\diamond}, T^{\diamond}, G^{\diamond})$, where:

- $L^{\diamond}$ is a (possibly infinite) instance over $\mathrm{sch}(\mathcal{T})$, the *legs* of $\diamond$,
- $B^{\diamond} = (\alpha_i^{\diamond})_{i \geq 0}$ is a sequence of atoms over $\mathrm{sch}(\mathcal{T})$ with constants and nulls (no variables), the *body* of $\diamond$,[2]
- $T^{\diamond} = (\sigma_i^{\diamond}, h_i^{\diamond})_{i>0}$ is a sequence of TGD-mapping pairs where $h_i^{\diamond}$ maps the variables in $\mathrm{body}(\sigma_i^{\diamond})$ to $\mathbf{C} \cup \mathbf{N}$, and
- $G^{\diamond} = (\gamma_i^{\diamond})_{i>0}$ is a sequence of atoms with $\gamma_i^{\diamond} \in \mathrm{body}(\sigma_i^{\diamond})$,

such that, for each $i \geq 0$, the following holds:

(1) $(\sigma_{i+1}^{\diamond}, h_{i+1}^{\diamond})$ is a trigger for $\mathcal{T}$ on $L^{\diamond} \cup \{\alpha_i^{\diamond}\}$;
(2) $\alpha_i^{\diamond} = h_{i+1}^{\diamond}(\gamma_{i+1}^{\diamond})$;
(3) $\alpha_{i+1}^{\diamond} = \mathrm{result}(\sigma_{i+1}^{\diamond}, h_{i+1}^{\diamond})$. ∎

It should not be difficult to see that a proto-caterpillar $\diamond$ for $\mathcal{T}$ as above encodes a "path-like" oblivious chase derivation (modulo repetition of triggers) of the (possibly infinite) instance $L^{\diamond} \cup \{\alpha_0^{\diamond}\}$ w.r.t. $\mathcal{T}$. Indeed, each atom $\alpha_i^{\diamond}$, for $i > 0$, of the sequence $B^{\diamond}$ can be derived from $L^{\diamond} \cup \{\alpha_{i-1}^{\diamond}\}$, i.e., the previous atom on the sequence and atoms of $L^{\diamond}$, via the trigger $(\sigma_i^{\diamond}, h_i^{\diamond})$. In other words, the infinite sequence of instances $(I_i)_{i \geq 0}$, where $I_0 = L^{\diamond} \cup \{\alpha_0^{\diamond}\}$ and, for $i > 0$, $I_i = I_{i-1} \cup \{\alpha_i^{\diamond}\}$, is an oblivious chase derivation (modulo repetition of triggers) of I_0 w.r.t. $\mathcal{T}$. But, even if we remove the repeated triggers, there is no guarantee that it is a restricted chase derivation for the following two reasons: an atom from $L^{\diamond}$ may stop an atom $\alpha_i^{\diamond}$ for $i > 0$, or an atom $\alpha_i^{\diamond}$ may stop an atom $\alpha_j^{\diamond}$ for $j > i$. This brings us to the notion of caterpillar, which is essentially a proto-caterpillar with the guarantee that the above two cases are excluded.

Definition 6.3. Consider a set $\mathcal{T}$ of TGDs. A *caterpillar for* $\mathcal{T}$ is a proto-caterpillar $\diamond = (L^{\diamond}, (\alpha_i^{\diamond})_{i \geq 0}, \cdot, \cdot)$ for $\mathcal{T}$ such that:

(1) for each $\beta \in L^{\diamond}$ and $i > 0$, $\beta \not\prec_s \alpha_i^{\diamond}$, and
(2) for each $0 \leq i < j$, $\alpha_i^{\diamond} \not\prec_s \alpha_j^{\diamond}$. ∎

It is an easy task to verify that a caterpillar $\diamond$ for $\mathcal{T}$ as above encodes a "path-like" restricted chase derivation of the (possibly infinite) instance $L^{\diamond} \cup \{\alpha_0^{\diamond}\}$ w.r.t. $\mathcal{T}$. However, it should not be forgotten that we are interested on finite databases. This brings us to the central notion of finitary caterpillar.

Definition 6.4. Consider a set $\mathcal{T}$ of TGDs. A *finitary caterpillar for* $\mathcal{T}$ is a caterpillar $(L^{\diamond}, \cdot, \cdot, \cdot)$ for $\mathcal{T}$ such that $L^{\diamond}$ is finite. ∎

Our goal is to characterize the existence of a database that gives rise to an infinite restricted chase derivation w.r.t. a sticky set $\mathcal{T}$ of TGDs via the existence of a finitary caterpillar for it.

THEOREM 6.5. *Let $\mathcal{T} \in \mathbb{S}$. The following are equivalent:*

(1) *There exists a database D such that there is an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$.*
(2) *There exists a finitary caterpillar for $\mathcal{T}$.*

The fact that $(2) \Rightarrow (1)$ follows by definition, and holds for every set of TGDs, not necessarily sticky. The interesting direction is

---

[2]By abuse of notation, we may sometimes treat $B^{\diamond}$ as the set of atoms $\{\alpha_i^{\diamond}\}_{i \geq 0}$.

(1) $\Rightarrow$ (2), which relies on stickiness. To this end, we are going to introduce refined variants of caterpillars, which will eventually lead to finitary caterpillars. In particular, we are going to introduce the notions of *(uniformly) connected caterpillar*, and *free caterpillar*, and establish the following chain of implications:

there exists a database D such that there is an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$
$\Rightarrow$ there exists a free connected caterpillar for $\mathcal{T}$
$\Rightarrow$ there exists a free uniformly connected caterpillar for $\mathcal{T}$
$\Rightarrow$ there exists a finitary caterpillar for $\mathcal{T}$,

which shows that indeed (1) $\Rightarrow$ (2). In the rest of the section, let $\mathcal{T}$ be a sticky set of single-head TGDs. For brevity, we will usually say (proto-)caterpillar meaning (proto-)caterpillar for $\mathcal{T}$.

Variants of Caterpillars

We first need some auxiliary terminology. Let $\alpha \in \mathrm{ochase}(D, \mathcal{T})$, for some database D, and assume that $\alpha = \mathrm{result}(\sigma, h)$. Let γ be an atom of $\mathrm{body}(\sigma)$, which means that $h(\gamma) \in \mathrm{ochase}(D, \mathcal{T})$ with $h(\gamma) \prec_p \alpha$. We say that the i-th position of $h(\gamma)$ and the j-th position of α are related, denoted as $(h(\gamma), i) \simeq (\alpha, j)$, if $\gamma[i] = \mathrm{head}(\sigma)[j]$. Moreover, the i-th and j-th positions of α are related, written as $(\alpha, i) \simeq (\alpha, j)$, if $\mathrm{head}(\sigma)[i] = \mathrm{head}(\sigma)[j]$. Now, for an instance $\mathcal{I} \subseteq \mathrm{ochase}(D, \mathcal{T})$, let $\Pi(\mathcal{I}) = \{(R(\bar{t}), i) : R(\bar{t}) \in \mathcal{I} \text{ and } 1 \leq i \leq \mathrm{ar}(R)\}$. We denote by $\simeq_{\mathcal{I}}^*$ the smallest equivalence relation that contains $(\Pi(\mathcal{I}))^2 \cap (\simeq)$. Intuitively, $(\alpha, i) \simeq_{\mathcal{I}}^* (\beta, j)$, for some atoms $\alpha, \beta \in \mathrm{ochase}(D, \mathcal{T})$, means that the terms $\alpha[i]$ and $\beta[j]$ are provably equal via a proof that uses only atoms of $\mathcal{I}$. Notice that $(\alpha, i) \simeq_{\mathcal{I}}^* (\beta, j)$ implies $\alpha[i] = \beta[j]$, but the opposite implication is not always true.

We also need the notion of the "birth atom" of a null value. Consider a parent-closed instance $\mathcal{I} \subseteq \mathrm{ochase}(D, \mathcal{T})$, and let $c \in \mathrm{dom}(\mathcal{I})$ be a null. We write $\beta^B(c)$ (which reads "the birth atom of c") for the atom of $\mathcal{I}$ such that: (1) c occurs in $\beta^B(c)$, and (2) for every $\alpha \in \mathcal{I}$ with $\alpha \prec_p \beta^B(c)$, c does not occur in α. It is clear that there is only one birth atom of c. Notice also that for an atom $\beta \in \mathcal{I}$ such that $\beta[j] = c$, it holds that β is the birth atom of c iff for each parent α of β and each position i of α, $(\alpha, i) \not\simeq (\beta, j)$.

We finally need to introduce the notion of immortal position, which relies on the marking procedure used in the definition of stickiness; see Section 2. Let $\alpha \in \mathrm{ochase}(D, \mathcal{T})$, for some database D, with $\alpha = \mathrm{result}(\sigma, h)$. The i-th position of α is *immortal* (w.r.t. $\mathcal{T}$) if the variable at the i-th position of $\mathrm{head}(\sigma)$ is *not* marked in $\mathcal{T}$. The name "immortal" reflects the fact that $\alpha[i]$ will be propagated forever, i.e., for every β such that $\alpha \prec_p \beta$, $\alpha[i] \in \mathrm{fr}(\beta)$.

We are now ready to introduce the first variant of caterpillars.

Definition 6.6. A caterpillar (or proto-caterpillar) $\diamond = (\cdot, B^\diamond, \cdot, \cdot)$, where $B^\diamond = (\alpha_i^\diamond)_{i \geq 0}$, is *connected* if there exist an infinite sequence $(\mathfrak{c}_i)_{i \geq 0}$ of terms, called the *relay terms* of $\diamond$, an infinite sequence $(b_i)_{i > 0}$ of integers with $b_1 < b_2 < b_3 < \cdots$, called the *pass-on points* of $\diamond$, and infinite sequences $(p_i)_{i > 0}$ and $(m_i)_{i \geq 0}$ (p for "parent" and m for "me") of integers from $[\mathrm{ar}(\mathcal{T})]$, such that, for each $k \geq 0$:

(1) $\mathfrak{c}_0$ occurs in $\alpha_0^\diamond$;
(2) $\alpha_{b_k}^\diamond = \beta^B(\mathfrak{c}_k)$;
(3) $\mathfrak{c}_k = \alpha_{b_k}^\diamond[m_k]$ and $\left(\alpha_{b_k}^\diamond, m_k\right) \simeq_{B^\diamond}^* \left(\alpha_{b_{k+1}}^\diamond, p_{k+1}\right)$;
(4) $\alpha_j^\diamond[i] = \mathfrak{c}_k$, for $i > 0$, $j \geq 0$, implies $\left(\alpha_j^\diamond, i\right)$ is not immortal. ∎

The above definition is indeed a bit technical. Intuitively, we can imagine the sequence of terms $\mathfrak{c}_0, \mathfrak{c}_1, \mathfrak{c}_2, \ldots$ as an infinite relay race, where the $\mathfrak{c}_i$'s are mortal runners, and their birth atoms are the batton passing points. In other words, connectedness ensures the continuous propagation of a new null in the underlying "path-like" chase derivation. However, the distance between two consecutive pass-on points can be arbitrarily large, i.e., there is no uniform bound. As we shall see later, having such a uniform bound is crucial for going from connected caterpillars to finitary caterpillars. This brings us to the next refined variant of caterpillars.

Definition 6.7. A caterpillar $\diamond$ is *uniformly connected* if it is connected and, with $(b_i)_{i > 0}$ being its pass-on points, there exists an integer $d \geq 0$ such that, for each $k \geq 0$, $b_{k+1} - b_k < d$. ∎

Let us now introduce the last variant of caterpillars that we need, namely free caterpillars. Recall that $(\alpha, i) \simeq_{L^\diamond \cup B^\diamond}^* (\beta, j)$, for $\alpha, \beta \in \mathrm{ochase}(L^\diamond \cup B^\diamond, \mathcal{T})$, means that $\alpha[i]$ and $\beta[j]$ are provably equal via a proof that uses only atoms of $L^\diamond \cup B^\diamond$. It would be very useful to ensure that also the other direction holds.

Definition 6.8. A (proto-)caterpillar $\diamond = (L^\diamond, B^\diamond, \cdot, \cdot)$ is *free* if, for each $(\alpha, i), (\beta, j) \in \Pi(L^\diamond \cup B^\diamond)$, $\alpha[i] = \beta[j]$ iff $(\alpha, i) \simeq_{L^\diamond \cup B^\diamond}^* (\beta, j)$. ∎

6.2 Implication 1: Extracting a Free Connected Caterpillar

We are now ready to establish the chain of implications discussed above, immediately after Theorem 6.5. We first focus on the first implication that states the following: if there exists a database D such that there is an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$, then there exists a free connected caterpillar for $\mathcal{T}$. Suppose $(I_i)_{i \geq 0}$ is an infinite restricted chase derivation of some database D w.r.t. $\mathcal{T}$, and let $\mathcal{I} = \bigcup_{i \geq 0} I_i$. We are going to extract from $(I_i)_{i \geq 0}$ a free connected caterpillar. The construction proceeds in three steps:

(1) First, we are going to construct a proto-caterpillar ♣.
(2) Then, we will convert ♣ into a connected proto-caterpillar ♠.
(3) Finally, from ♠ we will get a free connected caterpillar ♡.

Step 1: Construct a Proto-Caterpillar

We proceed to extract from $(I_i)_{i \geq 0}$ a proto-caterpillar ♣ that is "almost connected". Given a term (constant or null) c and a null c', both in $\mathrm{dom}(\mathcal{I})$, we say that c is a *parent term* of c' (w.r.t. $\mathcal{I}$), denoted $c <_p^t c'$, if c occurs in $\mathrm{fr}(\beta^B(c'))$. Notice that, for c to be a parent term of c' it is not enough to be in one of the parent atoms of the birth atom of c', but it needs to be propagated, via a frontier variable, during the application of the trigger that generates the birth atom of c'. Now, for each term c occurring in $\mathcal{I}$, we inductively define the *rank* of c (w.r.t. $\mathcal{I}$) as follows:

$$\mathrm{rank}(c) = \begin{cases} 0 & \text{if } c \in \mathrm{dom}(D), \\ 1 + \max\{\mathrm{rank}(c') : c' <_p^t c\} & \text{otherwise.} \end{cases}$$

For a term $c \in \mathrm{dom}(\mathcal{I})$ with $\mathrm{rank}(c) > 0$, select a term $c' \in \mathrm{dom}(\mathcal{I})$ such that $\mathrm{rank}(c') = \mathrm{rank}(c) - 1$ and $c' <_p^t c$. We call c' *the favourite parent* of c, and we write $c' <_{fp}^t c$.[3]

---

[3] We assume that there exists some fixed mechanism that selects c'. For example, c' can be the lexicographically first element of $\{c'' : \mathrm{rank}(c'') = \mathrm{rank}(c) - 1 \text{ and } c'' <_p^t c\}$.

It should be clear that the binary relation $<^t_{fp}$ over $\mathrm{dom}(I)$ forms an infinite forest F consisting of a finite number of trees, where the roots are terms from $\mathrm{dom}(D)$ of rank 0. But what about the out-degree of each node of F? We can show that, for each $i \geq 0$, the set $\{c \in \mathrm{dom}(I) : \mathrm{rank}(c) = i\}$ is finite. This can be shown by induction on $i \geq 0$, while the key fact is that only finitely many triggers can be formed due to which a null with rank $i + 1$ is generated (since, by induction hypothesis, the set of terms with rank at most i is finite). Thus, the nodes of F have finite out-degree. By applying König's Lemma[4] on F, we get that F contains an infinite simple path starting from a root node; let $c_0, c_1, c_2, \ldots$ be such a path.

By construction, for each $i \geq 0$, c_i occurs in the birth atom of c_{i+1}. Moreover, there exists a sequence of atoms $\beta^i_0, \beta^i_1, \beta^i_2, \ldots, \beta^i_{m_i} = \beta^B(c_{i+1})$, where $\beta^i_0 \in D$ if $i = 0$ and $\beta^i_0 = \beta^B(c_i)$ if $i > 0$, such that $\beta^i_k <_p \beta^i_{k+1}$, for each $0 \leq k < m_i$, and there are positions j in $\beta^B(c_i)$ and j' in $\beta^B(c_{i+1})$ such that $(\beta^B(c_i), j) \simeq^*_{P_i} (\beta^B(c_{i+1}), j')$, where P_i is the set of atoms $\{\beta^i_0, \beta^i_1, \ldots \beta^i_{m_i}\}$.

We are now ready to define $\clubsuit$. For brevity, let P be the infinite set of atoms $\bigcup_{i \geq 0} P_i$. Let $\clubsuit = (L^\clubsuit, B^\clubsuit, T^\clubsuit, G^\clubsuit)$, where

- $L^\clubsuit = \{\alpha \in I \setminus P : \text{ there is } \beta \in P \text{ such that } \alpha <_p \beta\}$,
- $B^\clubsuit = (\alpha^\clubsuit_i)_{i \geq 0}$ with $\alpha^\clubsuit_0, \alpha^\clubsuit_1, \alpha^\clubsuit_2, \ldots$ being the enumeration of the atoms of P such that $\alpha^\clubsuit_i <_p \alpha^\clubsuit_{i+1}$, for each $i \geq 0$,
- $T^\clubsuit = (\sigma^\clubsuit_i, h^\clubsuit_i)_{i>0}$ with $(\sigma^\clubsuit_i, h^\clubsuit_i)$, for $i > 0$, being the trigger for $\mathcal{T}$ on $B^\clubsuit \cup \{\alpha^\clubsuit_{i-1}\}$ such that $\alpha^\clubsuit_i = \mathrm{result}(\sigma^\clubsuit_i, h^\clubsuit_i)$, and
- $G^\clubsuit = (\gamma^\clubsuit_i)_{i>0}$ with $\gamma^\clubsuit_i \in \mathrm{body}(\sigma^\clubsuit_i)$ and $\alpha^\clubsuit_{i-1} = h^\clubsuit_i(\gamma^\clubsuit_i)$.

It should be clear, from the above construction, that the sequence of triggers $(\sigma^\clubsuit_i, h^\clubsuit_i)_{i>0}$ exists, and thus, $\clubsuit$ is well-defined. Then:

LEMMA 6.9. $\clubsuit$ is a proto-caterpillar for $\mathcal{T}$.

As said at the beginning of Step 1, the goal was to extract from $(I_i)_{i \geq 0}$ a proto-caterpillar that is "almost connected". It is easy to verify that the proto-caterpillar $\clubsuit$ is "almost connected" in the sense that all the conditions of Definition 6.6 are satisfied, with $c_0, c_1, \ldots$ playing the role of the relay terms, apart from (4). Indeed, there is no guarantee that $c_0, c_1, \ldots$ do not occur at immortal positions. Can we convert $\clubsuit$ into a connected proto-caterpillar that satisfies also condition (4)? This is the goal of the next step.

Step 2: Construct a Connected Proto-Caterpillar

It is clear that if a term c_i, for $i \geq 0$, occurs in an immortal position in some atom $\alpha^\clubsuit_j$, then it occurs in every $\alpha^\clubsuit_k$ for $k > j$. Since $\mathrm{ar}(\mathcal{T})$ is finite, we can have only finitely many integers $i \geq 0$ such that the term c_i occurs at an immortal position. Let i_0 be an integer greater than all such numbers i, which means that $c_{i_0}, c_{i_0+1}, c_{i_0+2}, \ldots$ do not occur at immortal positions. Let n be such that $\alpha^\clubsuit_n = \beta^B(c_{i_0})$, i.e., is the birth atom of c_{i_0}. It should be now clear how $\clubsuit$ can be converted into a connected proto-caterpillar $\spadesuit$. For brevity, let P be the set of atoms $\{\alpha^\clubsuit_{k+n} : k \geq 0\}$. Let $\spadesuit = (L^\spadesuit, B^\spadesuit, T^\spadesuit, G^\spadesuit)$, where

- $L^\spadesuit = \{\alpha \in I \setminus P : \text{ there is } \beta \in P \text{ such that } \alpha <_p \beta\}$,
- $B^\spadesuit = (\alpha^\spadesuit_i)_{i \geq 0}$ with $\alpha^\spadesuit_i = \alpha^\clubsuit_{i+n}$ for each $i \geq 0$,
- $T^\spadesuit = (\sigma^\spadesuit_i, h^\spadesuit_i)_{i>0}$ with $(\sigma^\spadesuit_i, h^\spadesuit_i) = (\sigma^\clubsuit_{i+n}, h^\clubsuit_{i+n})$ for $i > 0$, and
- $G^\spadesuit = (\gamma^\spadesuit_i)_{i>0}$ with $\gamma^\spadesuit_i = \gamma^\clubsuit_{i+n}$ for each $i > 0$.

---

[4] König's Lemma is a well-known result from graph theory: for an infinite directed rooted graph, if every node is reachable from the root, and every node has finite out-degree, then there exists an infinite directed simple path from the root.

Since, by Lemma 6.9, $\clubsuit$ is a proto-caterpillar for $\mathcal{T}$, we can conclude that $\spadesuit$ is also a proto-caterpillar. It also follows by construction that $\spadesuit$ is connected with $c_{i_0}, c_{i_0+1}, c_{i_0+2}, \ldots$ being its relay terms. Then:

LEMMA 6.10. $\spadesuit$ is a connected proto-caterpillar for $\mathcal{T}$.

Observe that there is no guarantee that $\spadesuit$ is a caterpillar since the two conditions in Definition 6.3 might be violated. Moreover, there is no guarantee that $\spadesuit$ is free, or, equivalently, that, for each $(\alpha, i), (\beta, j) \in \Pi(L^\spadesuit \cup B^\spadesuit)$, $\alpha[i] = \beta[j]$ implies $(\alpha, i) \simeq^*_{L^\spadesuit \cup B^\spadesuit} (\beta, j)$; recall that the other direction holds trivially. Can we convert $\spadesuit$ into a free connected caterpillar? This is the goal of the next step.

Step 3: Construct a Free Connected Caterpillar

To achieve our goal, we are going to carefully replace each term occurring in $L^\spadesuit \cup B^\spadesuit$ at a certain position $\pi \in \Pi(L^\spadesuit \cup B^\spadesuit)$ with a new constant that only depends on the equivalence class of π w.r.t. the equivalence relation $\simeq^*_{L^\spadesuit \cup B^\spadesuit}$. As usual, we write $[\pi]_{\simeq^*_{L^\spadesuit \cup B^\spadesuit}}$ for the equivalence class of π w.r.t. $\simeq^*_{L^\spadesuit \cup B^\spadesuit}$. Let $\bar{h}$ be a function that maps each atom $\alpha = R(t_1, \ldots, t_n) \in L^\spadesuit \cup B^\spadesuit$ to the atom

$$R\left(c_{[(\alpha,1)]_{\simeq^*_{L^\spadesuit \cup B^\spadesuit}}}, \ldots, c_{[(\alpha,n)]_{\simeq^*_{L^\spadesuit \cup B^\spadesuit}}}\right),$$

where, for each $1 \leq i \leq n$, $c_{[(\alpha,i)]_{\simeq^*_{L^\spadesuit \cup B^\spadesuit}}}$ is a constant from C.

Having the function $\bar{h}$ in place, it is not difficult to see how $\spadesuit$ can be converted into the desired free connected caterpillar $\heartsuit$. In particular, $\heartsuit = (L^\heartsuit, B^\heartsuit, T^\heartsuit, G^\heartsuit)$, where

- $L^\heartsuit = \{\bar{h}(\alpha) : \alpha \in L^\spadesuit\}$,
- $B^\heartsuit = (\alpha^\heartsuit_i)_{i \geq 0}$ with $\alpha^\heartsuit_i = \bar{h}(\alpha^\spadesuit_i)$ for each $i \geq 0$,
- $T^\heartsuit = (\sigma^\heartsuit_i, h^\heartsuit_i)_{i>0}$ with $\sigma^\heartsuit_i = \sigma^\spadesuit_i$ and $h^\heartsuit_i = \bar{h} \circ h^\spadesuit_i$, for $i > 0$,
- $G^\heartsuit = (\gamma^\heartsuit_i)_{i>0}$ with $\gamma^\heartsuit_i = \gamma^\spadesuit_i$ for each $i > 0$.

Stickiness allows us to show the following, which concludes the proof of the first implication; for the details see the appendix:

LEMMA 6.11. $\heartsuit$ is a free connected caterpillar for $\mathcal{T}$.

6.3 Implication 2: From a Free Connected Caterpillar to a Uniformly Connected One

Let us now concentrate on the second implication. The proof relies on the fact that we can check whether a free connected caterpillar exists via a deterministic Büchi automaton. As usual, for an automaton $\mathcal{A}$, we write $L(\mathcal{A})$ for its language. Then:

LEMMA 6.12. We can construct a deterministic Büchi automaton $\mathcal{A}_{\mathcal{T}}$ where $L(\mathcal{A}_{\mathcal{T}}) \neq \emptyset$ iff there is a free connected caterpillar for $\mathcal{T}$.

Let us stress that the purpose of the automaton $\mathcal{A}_{\mathcal{T}}$ provided by Lemma 6.12 is twofold: it is used here, together with a pumping argument, for establishing the second implication, and it will be also used in Section 6.5 for showing that the problem of deciding whether a finitary caterpillar exists is decidable in elementary time. We proceed to give some details concerning $\mathcal{A}_{\mathcal{T}}$ that allow us to intuitively explain how we get the second implication, while the detailed construction can be found in the appendix. It should not be surprising that this is the place where freeness plays a role.

The automaton $\mathcal{A}_{\mathcal{T}}$ operates on what we call caterpillar words over a finite alphabet $\Lambda_{\mathcal{T}}$ consisting of triples of the form (σ, γ, P), where $\sigma \in \mathcal{T}$, $\gamma \in \mathrm{body}(\sigma)$, and, with R being the predicate of

head(σ), $P \subseteq [\text{ar}(R)]$. Intuitively, a caterpillar word $\mathbf{w} = w_1 w_2, \cdots$, with $w_i = (\sigma_i, \gamma_i, P_i)$, is a candidate symbolic representation of a free connected caterpillar, where w_i marks a pass-on point iff P_i is non-empty. In fact, P_i indicates at which positions of head(σ_i) the new relay term appears. Roughly, $\mathcal{A}_{\mathcal{T}}$ accepts $\mathbf{w}$ iff $\mathbf{w}$ encodes a free connected caterpillar, while it enters an accepting state only when it reads a symbol w_i that marks a pass-on point.

We get the second implication from the following observation, which can be shown by an obvious pumping argument:

OBSERVATION 1. *Let $\mathcal{A}$ be a deterministic Büchi automaton with $n_{\mathcal{A}}$ states. If $L(\mathcal{A}) \neq \emptyset$, then there is $\mathbf{w} \in L(\mathcal{A})$ s.t. among each $n_{\mathcal{A}}$ consecutive states visited by $\mathcal{A}$ on input $\mathbf{w}$, at least one is accepting.*

Suppose now that there exists a free connected caterpillar. By Lemma 6.12, $L(\mathcal{A}_{\mathcal{T}}) \neq \emptyset$. By applying the above observation to the automaton $\mathcal{A}_{\mathcal{T}}$, we get a word $\mathbf{w}$ that encodes a free connected caterpillar $\diamond$ such that the distance between two consecutive pass-on points is bounded by the number of states of $\mathcal{A}_{\mathcal{T}}$. Thus, $\diamond$ is a free uniformly connected caterpillar, as needed.

6.4 Implication 3: From a Free Uniformly Connected Caterpillar to a Finitary One

We now proceed with the last implication. Consider a free uniformly connected caterpillar $\diamond = (L^\diamond, B^\diamond, T^\diamond, G^\diamond)$, where $B^\diamond = (\alpha_i^\diamond)_{i \geq 0}$, $T^\diamond = (\sigma_i^\diamond, h_i^\diamond)_{i > 0}$, and $G^\diamond = (\gamma_i^\diamond)_{i > 0}$. Our intention is to obtain from $\diamond$ a finitary caterpillar by unifying some terms of dom($L^\diamond$) in order to make $L^\diamond$ finite, while the rest remains a valid caterpillar. This can be done via what we call a unifying function.

A *unifying function* for $\diamond$ is a function $h : \mathfrak{B} \to \mathfrak{T}$, where $\mathfrak{B} \subseteq$ dom($L^\diamond$), and $\mathfrak{T}$ a set of new terms not occurring in $L^\diamond \cup B^\diamond$; it is called unifying since it essentially unifies the terms of $\mathfrak{B}$. Let $h(\diamond) = \left(h(L^\diamond), (h(\alpha_i^\diamond))_{i \geq 0}, (\sigma_i, h \circ h_i^\diamond)_{i > 0}, (\gamma_i^\diamond)_{i > 0} \right)$. Then:

LEMMA 6.13. *There exists a unifying function h for $\diamond$ such that $h(\diamond)$ is a finitary caterpillar for $\mathcal{T}$.*

It is not difficult to show that no matter how a unifying function h for $\diamond$ is defined, $h(\diamond)$ is a proto-caterpillar that satisfies condition (1) of Definition 6.3. The non-trivial task is to define h is such a way that $h(L^\diamond)$ is finite, and $h(\diamond)$ satisfies condition (2) of Definition 6.3. The key here is, by exploiting uniformity, which provides a bound on the distance between two consecutive pass-on points, to define a sufficiently large *finite* set of new terms to which infinitely many carefully chosen terms of dom($L^\diamond$) are mapped to; the details can be found in the appendix. This completes the proof of Theorem 6.5.

6.5 Deciding $\text{CT}_{\forall\forall}^{\text{res}}(\mathbb{S})$ via Büchi Automata

By Theorems 4.1 and 6.5, given a set $\mathcal{T} \in \mathbb{S}$, deciding whether $\mathcal{T} \notin \mathbb{CT}_{\forall\forall}^{\text{res}}$ is equivalent to the problem of checking whether there exists a finitary caterpillar for $\mathcal{T}$. By exploiting the Büchi automaton provided by Lemma 6.12, we can easily show that:

LEMMA 6.14. *The problem of deciding whether there exists a finitary caterpillar for $\mathcal{T}$ is decidable in elementary time.*

Since the emptiness problem of deterministic Büchi automata is feasible in linear time in the size of the automaton, and since the automaton provided by Lemma 6.12 can be constructed in elementary

time, checking whether a free connected caterpillar for $\mathcal{T}$ exists is feasible in elementary time. Now, observe that the three implications established above, together with the (2) $\Rightarrow$ (1) direction of Theorem 6.5, imply that there exists a free connected caterpillar iff there exists a finitary caterpillar, and Lemma 6.14 follows.

7 FUTURE WORK

Here are some non-trivial questions that beg for an answer: (1) What about the exact complexity of our problems? (2) What about restricted chase termination for guarded or sticky sets of multi-head TGDs? (3) What about the more liberal version of the problem that asks whether there is a finite restricted chase derivation?

REFERENCES

[1] Alfred. V. Aho, Yehoshua Sagiv, and Jeffrey D. Ullman. 1979. Efficient Optimization of a Class of Relational Expressions. *ACM Trans. Database Syst.* 4, 4 (1979).
[2] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175, 9-10 (2011), 1620–1654.
[3] Pablo Barceló, Georg Gottlob, and Andreas Pieris. 2016. Semantic Acyclicity Under Constraints. In *PODS*. 343–354.
[4] Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. 2017. Benchmarking the Chase. In *PODS*. 37–52.
[5] Marco Calautti, Georg Gottlob, and Andreas Pieris. 2015. Chase Termination for Guarded Existential Rules. In *PODS*. 91–103.
[6] Marco Calautti and Andreas Pieris. 2019. Oblivious Chase Termination: The Sticky Case. In *ICDT*. 17:1–17:18.
[7] 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.
[8] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14 (2012), 57–83.
[9] Andrea Calì, Georg Gottlob, and Andreas Pieris. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193 (2012), 87–128.
[10] Jan Van den Bussche. 2015. Personal Communication.
[11] Alin Deutsch, Alan Nash, and Jeff B. Remmel. 2008. The Chase Revisisted. In *PODS*. 149–158.
[12] Alin Deutsch and Val Tannen. 2003. Reformulation of XML Queries and Constraints. In *ICDT*. 225–241.
[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 Jerzy Marcinkowski. 2014. All-Instances Termination of Chase is Undecidable. In *ICALP*. 293–304.
[15] Gösta Grahne and Adrian Onet. 2018. Anatomy of the Chase. *Fundam. Inform.* 157, 3 (2018), 221–270.
[16] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. 2013. Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies. *J. Artif. Intell. Res.* 47 (2013), 741–808.
[17] Sergio Greco, Cristian Molinaro, and Francesca Spezzano. 2012. *Incomplete Data and Data Dependencies in Relational Databases*. Morgan & Claypool Publishers.
[18] Sergio Greco, Francesca Spezzano, and Irina Trubitsyna. 2011. Stratification Criteria and Rewriting Techniques for Checking Chase Termination. *PVLDB* 4, 11 (2011), 1158–1168.
[19] Markus Krötzsch, Maximilian Marx, and Sebastian Rudolph. 2019. The Power of the Terminating Chase (Invited Talk). In *ICDT*. 3:1–3:17.
[20] Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo, and Federico Ulliana. 2019. A Single Approach to Decide Chase Termination on Linear Existential Rules. In *ICDT*. 18:1–18:19.
[21] Bruno Marnette. 2009. Generalized schema-mappings: from termination to tractability. In *PODS*. 13–22.
[22] Michael Meier, Michael Schmidt, and Georg Lausen. 2009. On Chase Termination Beyond Stratification. *PVLDB* 2, 1 (2009), 970–981.
[23] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. 2015. RDFox: A Highly-Scalable RDF Store. In *ISWC*. 3–20.
[24] Jacopo Urbani, Markus Krötzsch, Ceriel J. H. Jacobs, Irina Dragoste, and David Carral. 2018. Efficient Model Construction for Horn Logic with VLog - System Description. In *IJCAR*. 680–688.

A ADDITIONAL NOTIONS

An *equality type* over a schema S is a pair (R, E), where $R \in S$ and E is a partition of $\{1, \ldots, \text{ar}(R)\}$ (which we see as a set of subsets of $\{1, \ldots, \text{ar}(R)\}$). Let $\text{etypes}(S)$ be the set of all possible equality types over S, which is clearly finite. Given an atom $\alpha = R(t_1, \ldots, t_n)$ over S, its equality type, denoted $\text{et}(\alpha)$, is the equality type $(R, E) \in \text{etypes}(S)$ such that $t_i = t_j$ iff i, j coexist in a set of E. A homomorphism from a set of atoms A to a set of atoms B is an *isomorphism* from A to B if it is 1-1, and its inverse h^{-1} is a homomorphism from B to A.

B PROOFS FROM SECTION 4

B.1 Fairness Theorem and Multi-head TGDs

Here is an example showing that the Fairness Theorem (Theorem 4.1) does not hold for multi-head TGDs:

Example B.1. Consider the set $\mathcal{T}$ of TGDs consisting of

$$R(x, y, y) \;\rightarrow\; \exists z\, R(x, z, y), R(z, y, y) \qquad R(x, y, z) \;\rightarrow\; R(z, z, z).$$

It should be clear that there exists an infinite restricted chase derivation of $\{R(a, b, b)\}$ w.r.t. $\mathcal{T}$; apply only the first TGD. However, every valid restricted chase derivation of $\{R(a, b, b)\}$ w.r.t. $\mathcal{T}$ is finite. ∎

Let us say that the recent paper [20], which studies the restricted chase termination problem for linear TGDs, provides an example that refutes Theorem 4.1 even if we use only binary predicates.

B.2 Proof of Lemma 4.3

Clearly, $J_0^0 = D$ since $(J_i^0)_{i \geq 0}$ is a chase derivation of D w.r.t. $\mathcal{T}$. It remains to show that, for $k \geq 0$, there is an active trigger (σ, h) for $\mathcal{T}$ on J_k^k such that $J_k^k \langle \sigma, h \rangle J_{k+1}^{k+1}$. Note that $J_k^k = J_k^{k+1}$ by the diagonal property. Thus, it suffices to show that there exists an active trigger (σ, h) for $\mathcal{T}$ on J_k^{k+1} such that $J_k^{k+1} \langle \sigma, h \rangle J_{k+1}^{k+1}$. This holds since $(J_i^{k+1})_{i \geq 0}$ is a chase derivation of D w.r.t. $\mathcal{T}$.

B.3 Proof of Lemma 4.4

Let $\mathcal{I}^n = \bigcup_{i \geq 0} I_i^n$. Since $\mathcal{T}$ is finite, it suffices to show that, for each TGD $\hat{\sigma} \in \mathcal{T}$, the set of atoms $B_{\hat{\sigma}}$ that collects all the atoms of $\mathcal{I}^n$ that are stopped by $\text{result}(\sigma, h)$ and are generated by a trigger that involves $\hat{\sigma}$, is finite. Indeed, $B_{\hat{\sigma}}$ is finite implies A is finite, since the cardinality of $B_{\hat{\sigma}}$ coincides with the cardinality of $\{i \geq 0 : \text{result}(\sigma, h) \prec_s \text{result}(\sigma_i, h_i) \text{ and } \sigma_i = \hat{\sigma}\}$.

Since all the atoms of $B_{\hat{\sigma}}$ have been created by the same TGD, and they are all stopped by $\text{result}(\sigma, h)$, we can conclude that they are equal when restricted to their frontier. Towards a contradiction, assume that two atoms $\alpha, \beta \in B_{\hat{\sigma}}$ have the same equality type. This implies that $\alpha \prec_s \beta$ and $\beta \prec_s \alpha$, which contradicts the fact that α and β belong to the result of a restricted chase derivation. Thus, for every two distinct atoms $\alpha, \beta \in B_{\hat{\sigma}}$, α and β have different equality types. Since there are only finitely many equality types over $\text{sch}(\mathcal{T})$, we conclude that $B_{\hat{\sigma}}$ is finite, as needed.

B.4 Proof of Lemma 4.5

By construction, $I_0^{n+1} = I_0^n = D$. It remains to show that, for $i \geq 0$, there is an active trigger (σ_i, h_i) for $\mathcal{T}$ on I_i^{n+1} such that $I_i^{n+1} \langle \sigma_i, h_i \rangle I_{i+1}^{n+1}$. We proceed by considering the following cases:

Case 1. For $0 \leq i \leq \ell$, the claim is trivial since $(I_i^{n+1})_{0 \leq i \leq \ell} = (I_i^n)_{0 \leq i \leq \ell}$, while $(I_i^n)_{i \geq 0}$ is a chase derivation of D w.r.t. $\mathcal{T}$.

Case 2. For $i = \ell + 1$, the claim holds since (σ, h) is an active trigger for $\mathcal{T}$ on $I_\ell^n = I_\ell^{n+1}$ and $I_i^{n+1} = I_{i-1}^{n+1} \cup \{\text{result}(\sigma, h)\}$.

Case 3. Finally, assume that $i \geq \ell + 2$. Recall that I_{i-1}^n is obtained from I_{i-2}^n by applying the active trigger (σ_{i-2}, h_{i-2}). We are going to show that $I_{i-1}^{n+1} \langle \sigma_{i-2}, h_{i-2} \rangle I_i^{n+1}$. Clearly, $I_i^{n+1} = I_{i-1}^{n+1} \cup \{\text{result}(\sigma_{i-2}, h_{i-2})\}$ since $I_{i-1}^n = I_{i-2}^n \cup \{\text{result}(\sigma_{i-2}, h_{i-2})\}$. It is also clear that (σ_{i-2}, h_{i-2}) is a trigger for $\mathcal{T}$ on I_{i-1}^{n+1}. It remains to show that it is also active. Assume that this is not the case. Fact 3.5 implies that there is an atom $\alpha \in I_{i-1}^{n+1}$ such that $\alpha \prec_s \text{result}(\sigma_{i-2}, h_{i-2})$. Recall that $I_{i-1}^{n+1} = I_{i-2}^n \cup \{\text{result}(\sigma, h)\}$. Since (σ_{i-2}, h_{i-2}) is an active trigger for $\mathcal{T}$ on I_{i-2}^n, we conclude that $\alpha \notin I_{i-2}^n$. Moreover, since $i-2$ is greater that all the elements of A, we get that $\text{result}(\sigma, h)$ does not stop $\text{result}(\sigma_{i-2}, h_{i-1})$, which implies that $\alpha \neq \text{result}(\sigma_{i-2}, h_{i-1})$. Hence, $\alpha \notin I_{i-1}^{n+1}$, which is a contradiction.

C PROOFS FROM SECTION 5

C.1 Proof of Theorem 5.3

The fact that $(1) \Rightarrow (2)$ is easy: simply define A as the set $\bigcup_{i \geq 0} I_i$, where $(I_i)_{i \geq 0}$ is the infinite restricted chase derivation of D w.r.t $\mathcal{T}$ that exists by hypothesis. For the other direction, by exploiting A, we are going to inductively construct an infinite restricted chase derivation $(I_i)_{i \geq 0}$ of D w.r.t $\mathcal{T}$. Clearly, I_0 is defined as D. Suppose that we have already constructed $(I_i)_{1 \leq i \leq n-1}$, for some natural number $n > 1$. Due to condition (1) of Definition 5.2, there exists an atom $\alpha \in A \setminus B$, where $B = \bigcup_{0 \leq i < n} I_i$, that is minimal w.r.t. $\prec_b$, i.e., for every $\beta \in A \setminus B$, $\alpha \prec_b \beta$. We define I_n as the instance $B \cup \{\alpha\}$. It remains to show that there exists an active trigger (σ, h) for $\mathcal{T}$ on B such that $\alpha = \text{result}(\sigma, h)$. By condition (2) of Definition 5.2, we get that all the parents of α occur in B, and thus (σ, h) is a trigger for $\mathcal{T}$ on B. To show that (σ, h) is active, by Fact 3.5, we need to show that there is no $\beta \in B$ such that $\beta \prec_s \alpha$. Towards a contradiction, assume that such β exists. This implies that $\alpha \prec_b \beta$ (recall that $\prec_s^{-1} \subseteq \prec_b$). But this implies that $\prec_b$ over B contains a cycle, which is a contradiction due to the third condition of Definition 5.2, and the claim follows.

C.2 Proof of Theorem 5.5

In the rest of the subsection, let $\mathcal{T}$ be a set of single-head guarded TGDs. As it is common when studying guarded TGDs, we need a refined version of the parent relation over the real oblivious chase that distinguishes between guard- and side-parents.

Guard- and Side-Parent Relation. Consider the real oblivious chase $\text{ochase}(D, \mathcal{T}) = \langle V, \prec_p, \lambda, \tau \rangle$ of a database D w.r.t. $\mathcal{T}$. We can naturally define the *guard-parent* relation $\prec_{gp}$ over V as the subrelation of $\prec_p$ by keeping only the pairs of nodes (v, u) where v corresponds to the guard atom of the TGD in $\tau(u)$. Formally, the guard-parent relation $\prec_{gp}$ (over V) is defined as

$$\{\langle v, u \rangle \;:\; v \prec_p u \text{ and, with } \tau(u) = (\sigma, h),\, h(\text{guard}(\sigma)) = \lambda(v)\}.$$

For a node u, we may write $gp(u)$ for its guard-parent, i.e., if $v \prec_{gp} u$, then $gp(u) = v$. We denote by $\prec_{gp}^+$ the transitive closure of $\prec_{gp}$. Observe that, due to guardedness, $\text{ochase}(D, \mathcal{T})$ can be seen as a forest w.r.t. $\prec_{gp}$, with the nodes of V labeled with atoms of D being

the roots of the trees, and all the other nodes are the non-root nodes. It would be conceptually useful to have this forest in mind.

Regarding the side-parents, it is not enough to simply keep the pairs $v \prec_p u$ where v corresponds to a side atom (i.e., an atom different than the guard) of the TGD in $\tau(u)$. In addition, we need to know which terms of the atom $\lambda(gp(u))$ occur in $\lambda(v)$ and at which positions.[5] This can be achieved via the notion of sideatom type.[6] A *sideatom type* π (w.r.t $\mathcal{T}$) is a triple $\langle P, m, \xi \rangle$, where $P/n \in \text{sch}(\mathcal{T})$, $m \leq \text{ar}(\mathcal{T})$ is a natural number, called the arity of π, and $\xi : [n] \rightarrow [m]$. Given two atoms β and γ, we say that β is a π-sideatom of γ, denoted $\beta \subseteq_\pi \gamma$, if the predicate of β is P, the predicate of γ has arity m, and $\beta[i] = \gamma(\xi(i))$ for each $i \in [n]$. For example, the atom $\beta = P(a, b, c)$ is a π-sideatom of $\gamma = R(a, d, c, b)$ with $\pi = \langle P, 4, \{1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 3\} \rangle$. Consider now a node $u \in V$ such that $v \prec_p u, v_1 \prec_p u, \ldots, v_m \prec_p u, \tau(u) = (\sigma, h)$, where $\text{body}(\sigma) = \gamma, \gamma_1, \ldots, \gamma_m$ with $\gamma = \text{guard}(\sigma)$, and $h(\gamma) = \lambda(v), h(\gamma_1) = \lambda(v_1), \ldots, h(\gamma_m) = \lambda(v_m)$. Let $\pi_1, \ldots, \pi_m$ be sideatom types such that, for each $i \in [m]$, $\lambda(v_i) \subseteq_{\pi_i} \lambda(v)$ (or $\lambda(v_i) \subseteq_{\pi_i} \lambda(gp(u))$). Then, for each $i \in [m]$, we say that v_i is a π_i-*side-parent* of u, written $v_i \prec_{sp}^{\pi_i} u$.

As for the relation $\prec_p$, notice that, strictly speaking, $\prec_{gp}$ and $\prec_{sp}^\pi$, for some sideatom type π, are relations over the node set V of $\text{ochase}(D, \mathcal{T})$. However, for convenience, we will usually see these relations as relations over the set multiset consisting of the atoms of $\text{ochase}(D, \mathcal{T})$. Thus, we will directly refer to the guard-parent of an atom α of $\text{ochase}(D, \mathcal{T})$ and write $gp(\alpha)$.

Let us now proceed with the proof of Theorem 5.5. By hypothesis, there exists an infinite restricted chase derivation $(I_i)_{i \geq 0}$ of D w.r.t. $\mathcal{T}$. The proof proceeds in three main steps:

(1) We first construct from D an acyclic database D_{ac}. In fact, we explicitly construct a join tree (T_{ac}, λ), where $T_{ac} = (V, E)$, and the database D_{ac} is defined as $\{\lambda(v) : v \in V\}$.

(2) We then show that there is an auxiliary infinite sequence of instances $(K_i)_{i \geq 0}$, where $K_0 = D_{ac}$, which somehow mimics the infinite restricted chase derivation $(I_i)_{i \geq 0}$ of D w.r.t. $\mathcal{T}$.

(3) Finally, by exploiting the sequence $(K_i)_{i \geq 0}$, we construct an infinite restricted chase derivation $(J_i)_{i \geq 0}$ of D_{ac} w.r.t. $\mathcal{T}$.

We proceed to give more details for each of the above steps. But first we need to fix some notation. Let $\mathcal{I} = \bigcup_{i \geq 0} I_i$. We write $(\sigma_i^{\mathcal{I}}, h_i^{\mathcal{I}})$ for the trigger such that $I_i \langle \sigma_i^{\mathcal{I}}, h_i^{\mathcal{I}} \rangle I_{i+1}$. For brevity, we write $\beta_i^{\mathcal{I}}$ for the atom $\text{result}(\sigma_i^{\mathcal{I}}, h_i^{\mathcal{I}})$, and $\gamma_i^{\mathcal{I}}$ for $\text{guard}(\sigma_i^{\mathcal{I}})$. Given an atom $\beta \in \mathcal{I}$, we define $\mathcal{I}_\beta$ as the set $\{\alpha \in \mathcal{I} : \beta \prec_{gp}^+ \alpha\}$.

Step 1: The Acyclic Database D_{ac}

Since D is finite, while $\mathcal{I}$ is infinite, we can conclude that there exists an atom $\alpha^\infty \in D$ such that the set $\mathcal{I}_{\alpha^\infty}$ is infinite. One may think that the acyclic database D_{ac} consists of the atom α^∞ together with the atoms of D that can serve as its side atoms, i.e., the database

$$\{\alpha^\infty\} \cup \{\beta \in D : \beta \subseteq_\pi \alpha^\infty \text{ for some sideatom type } \pi\}.$$

However, as explained in the main body of the paper (see Example 5.6), this is not the case due to what we call remote side-parents:

---

[5]For the discussion in the main body of the paper, the simple side-parent relation $\prec_{sp}$ was enough. However, for the formal proof we need this additional information.
[6]Note that this notion has been also introduced and used in Section 5.3 where we talk about chaseable abstract join trees. We repeat it here for the sake of readability.

Definition C.1. Consider two distinct atoms $\alpha, \beta \in D$, and two atoms $\alpha', \beta' \in \mathcal{I}$. The tuple $\langle \alpha, \alpha', \beta, \beta' \rangle$ is a *remote-side-parent situation* if $\alpha \prec_{gp}^+ \alpha'$, $\beta \prec_{gp}^+ \beta'$, and $\beta' \prec_{sp}^\pi \alpha'$ for some sideatom type π. If this is the case, then we say that α *longs for* β. ∎

The following easy lemma collects a couple of useful facts about the notion of remote-side-parent situation, which would be crucial for the construction of the acyclic database D_{ac}.

LEMMA C.2. (1) *If $\langle \alpha, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation, then $\beta' \subseteq_\pi \alpha$ and $\beta' \subseteq_{\pi'} \beta$ for some types π and π'.*
(2) *There exists a natural number ℓ_∞ such that, if $\langle \alpha^\infty, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation, then $\beta' \in I_{\ell_\infty}$.*

PROOF. It is easy to verify that claim (1) holds due to guardedness. For claim (2) it suffices to observe that the following holds, which is a consequence of (1): for an atom $\alpha \in D$, there are only finitely many pairs of atoms β, β' such that, for some atom α' with $\alpha \prec_{gp}^+ \alpha'$, $\langle \alpha, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation. □

The Construction of D_{ac}. Let us now formally define the acyclic database D_{ac}. We will construct, via simultaneous induction:

(1) A labeled tree (T_{ac}, λ), where $T_{ac} = (V, E)$ and λ is a labeling function from V to $\{R([t_1]_v, \ldots, [t_n]_v) : R/n \in \text{sch}(\mathcal{T}), t_i \in \text{dom}(D) \text{ and } v \in V\}$, i.e., the set of atoms that can be formed using predicates of $\text{sch}(\mathcal{T})$ and constants from the set $\{[t]_v : t \in \text{dom}(D) \text{ and } v \in V\}$.

(2) A mapping h_{ac} from $\{\lambda(v) : v \in V\}$ to D.

(3) A function depth from $\{\lambda(v) : v \in V\}$ to $\mathbb{N}$.

The constants of the form $[t]_v$ used above provide us with a simple mechanism for uniformly renaming a constant $t \in \text{dom}(D)$ into a fresh constant, while this renaming step is performed with respect to a certain node v of T_{ac}. This allows us to break the connection among occurrences of the same constant that are semantically different; this will be made clear in a while. The construction follows:

Base Case. Let $v \in V$ be the root node of T_{ac}. Then, $\lambda(v) = \alpha^\infty$, $h_{ac}(\lambda(v)) = \alpha^\infty$, and $\text{depth}(\lambda(v)) = 0$.

Inductive Step. Assume that $v \in V$ is such that $h_{ac}(\lambda(v)) = \alpha$, for some $\alpha \in D$, with $\text{depth}(\lambda(v)) < \ell_\infty$. Then, for each $\beta \in D$ such that α longs for β, we add a new node u to V, and the edge (v, u) to E, in such a way that:

- the atom $\lambda(u)$ is of the following form:
  - it has the same predicate as the atom β,
  - $\lambda(u)[i] = \lambda(u)[j]$ iff $\beta[i] = \beta[j]$,
  - $\lambda(u)[i] = \lambda(v)[j]$ iff $\beta[i] = \alpha[j]$,
  - if $\lambda(u)[i]$ does not occur in $\lambda(v)$, then $\lambda(u)[i] = [\beta[i]]_u$.
- $h_{ac}(\lambda(u)) = \beta$, and
- $\text{depth}(\lambda(u)) = \text{depth}(\lambda(v)) + 1$.

This completes the construction of (T_{ac}, λ), h_{ac} and depth. Having (T_{ac}, λ) in place, we define D_{ac} as $\{\lambda(v) : v \in V\}$.

Before we proceed any further, it is important to observe that different nodes v, u of T_{ac} (possibly of different depths) may have the same label, i.e. it may happen that $\lambda(v) = \lambda(u)$. In this case, we treat them as two different atoms since, although syntactically the same, they are present in T_{ac} for different reasons. Therefore, strictly speaking, D_{ac} is a *multiset database*, i.e., it can hold many occurrences of the same atom, which are treated as different atoms.

Let us clarify that the notion of acyclicity given in Definition 5.4 can be directly applied to multiset instances, i.e., a multiset instance is acyclic iff it admits a join tree.

Lemma C.3.   (1) D_{ac} is an acyclic multiset database.
    (2) The mapping h_{ac} is a homomorphism from D_{ac} to D.
    (3) For each two vertices $u, v \in T_{ac}$, the mapping h_{ac} is an isomorphism from $\{\lambda(u), \lambda(v)\}$ to $\{h_{ac}(\lambda(u)), h_{ac}(\lambda(v))\}$

Proof. For (1) it suffices to show that T_{ac} is finite, and that it enjoys the connectedness condition (condition (2) of Definition 5.4). By Lemma C.2(1), for $\alpha \in D$, there are only finitely many pairs of atoms β, β' such that $\langle \alpha, \alpha', \beta, \beta' \rangle$ is a remote-side-parent situation for some α'. Thus, by construction, the branching degree of T_{ac} is finite. Since the depth of T_{ac} is bounded by ℓ_∞, T_{ac} is finite. The fact that T_{ac} enjoys the connectedness condition follows by construction; here, the renaming of the constants t of dom(D) to $[t]_v$ is crucial. Claims (2) and (3) also follow by construction. $\square$

It remains to show that there exists an infinite restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$. Indeed, by showing the latter statement for the multiset database D_{ac}, we can conclude that there exists an infinite restricted chase derivation of the acyclic database obtained from D_{ac} by keeping only one occurrence of each atom w.r.t. $\mathcal{T}$.

Step 2: An Auxiliary Infinite Sequence of Instances

As we already explained, D_{ac} consists of several (slightly modified) copies of atoms of D. It is like seeing the atoms of D through several distorting mirrors, where the mirror images are atoms of D_{ac}. Imagine now that we watch the restricted chase derivation $(I_i)_{i\geq0}$ through those mirrors. Although during a restricted chase step only one atom, let us say α, is generated, in the mirrors we see the generation of several atoms, which are the distorted images of α. In order to formalize this phenomenon, we define a variant of the restricted chase, called *weakly restricted chase*.

Weakly Restricted Chase. Our intention is to define a variant of chase that allows us to apply several active triggers at the same time, and operates on multiset instances, i.e., multisets of atoms. The reason why we need to consider multisets is because two different mirror images may be syntactically the same.

Definition C.4. Consider a multiset instance K, and let S be a set of active triggers for $\mathcal{T}$ on K. An application of S to K, called *weakly restricted chase step*, returns the multiset instance

$$K' \;=\; K \;\cup\; \{\text{result}(\sigma, h) : (\sigma, h) \in S\},$$

and is denoted as $K\langle S\rangle K'$. A sequence of multiset instances $(K_i)_{i\geq0}$, where K_0 is the database D', is a *weakly restricted chase derivation* of D' w.r.t. $\mathcal{T}$ if, for each $i \geq 0$, there exists a set S of active triggers for $\mathcal{T}$ on K_i such that $K_i\langle S\rangle K_{i+1}$. $\blacksquare$

The auxiliary infinite sequence of instances that we are looking for, which will eventually lead to an infinite restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$, is an infinite weakly restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$. This is essentially (modulo some condition, called the *depth condition*, given below) the infinite restricted chase derivation $(I_i)_{i\geq0}$ seen through the mirrors discussed above.

The Auxiliary Sequence $(K_i)_{i\geq0}$. We now inductively construct a sequence of multiset instances $(K_i)_{i\geq0}$, together with a mapping $\bar{h}$ from $\bigcup_{i\geq0} K_i$ to $\mathcal{I}$:

Base Case. Let $K_0 = D_{ac}$, and for each $\alpha \in K_0$, $\bar{h}(\alpha) = h_{ac}(\alpha)$. Recall that h_{ac} is the mapping from D_{ac} to D provided by Lemma C.3.

Inductive Step. Suppose now that K_i and $\bar{h} : K_i \to \mathcal{I}$ have been already defined, for $i > 0$. Let S_i be the set of all active triggers for $\mathcal{T}$ on K_i of the form $(\sigma_i^{\mathcal{I}}, h)$, i.e., they use the same TGD $\sigma_i^{\mathcal{I}}$ that has been used in $(I_i)_{i\geq0}$ to generate the atom $\beta_i^{\mathcal{I}}$, such that:

    (1) $\bar{h}(h(\gamma_i^{\mathcal{I}})) = gp(\beta_i^{\mathcal{I}})$, which simply states that the atom of K_i that is now about to become a guard-parent must be a mirror image of the guard-parent of the atom $\beta_i^{\mathcal{I}}$ in $\mathcal{I}$.

    (2) **(Depth Condition)** $\alpha <_{gp}^+ h(\gamma_i^{\mathcal{I}})$, for some atom $\alpha \in D_{ac}$ such that $\alpha = \alpha^\infty$, or depth(α) $< \ell_\infty - i$.

We define K_{i+1} as the multiset instance

$$K_i \;\cup\; \{\text{result}(\sigma_i^{\mathcal{I}}, h) : (\sigma_i^{\mathcal{I}}, h) \in S_i\},$$

i.e., $K_i\langle S_i\rangle K_{i+1}$. Furthermore, for each $\alpha \in K_{i+1} \setminus K_i$, let $\bar{h}(\alpha) = \beta_i^{\mathcal{I}}$. This completes the definition of $(K_i)_{i\geq0}$.

The Structure of the Auxiliary Sequence. By construction, $(K_i)_{i\geq0}$ is a weakly restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$. What is not immediately clear is that $(K_i)_{i\geq0}$ is infinite. Our goal, in the rest of this subsection, is to understand how $\mathcal{K} = \bigcup_{i\geq0} K_i$ relates to $\mathcal{I}$. This analysis will give us useful information about the structure of $\mathcal{K}$, which will be crucial later, and also it will allow us to conclude that $\mathcal{K}$ is infinite.

For an atom $\beta \in \mathcal{K}$, we define $\mathcal{K}_\beta$ as the set $\{\alpha \in \mathcal{K} : \beta <_{gp}^+ \alpha\}$. The main technical lemma that we need to show follows:

Lemma C.5. *For each $i \geq 0$, the following statements hold:*

    (1) *For each $\alpha \in D_{ac}$ such that $\alpha \neq \alpha^\infty$, $\bar{h}$ is an isomorphism from $\mathcal{K}_\alpha \cap K_i$ to $\mathcal{I}_{\bar{h}(\alpha)} \cap I_k$, where $k = \min\{i, \ell_\infty - \text{depth}(\alpha)\}$.*
    (2) *The mapping $\bar{h}$ is an isomorphism from $\mathcal{K}_{\alpha^\infty} \cap K_i$ to $\mathcal{I}_{\alpha^\infty} \cap I_i$.*

Proof. For $\alpha \in D_{ac}$, we define $T_i^K(\alpha) = \mathcal{K}_\alpha \cap K_i$ and $T_i^I(\alpha) = \mathcal{I}_{\bar{h}(\alpha)} \cap I_k$, where $k = \min\{i, \ell_\infty - \text{depth}(\alpha)\}$ if depth(α) > 0, and $k = i$ otherwise. Let also α_i^I be an atom of D such that $\alpha_i^I <_{gp}^+ \beta_i^I$.

The lemma says that, for $i \geq 0$ and $\alpha \in D_{ac}$, $\bar{h}$ is an isomorphism from $T_i^K(\alpha)$ to $T_i^I(\alpha)$. We proceed by induction on i. The lemma holds for $i = 0$, since $I_0 = D$ and $K_0 = D_{ac}$. Assume now that it is also true for some $i \geq 0$.

First observe that, if depth(α) > 0 and depth(α) $\geq \ell_\infty - i$, for $\alpha \in D_{ac}$, then $T_i^K(\alpha) = T_{i+1}^K(\alpha)$ (this follows from the Depth Condition), and $T_i^I(\alpha) = T_{i+1}^I(\alpha)$. Thus, for such α, the claim directly follows from the hypothesis.

It of course follows from the hypothesis (the part where it tells us something about $\bar{h}$) that, if $\beta <_{gp}^+ \beta'$, for some $\beta, \beta' \in K_i$, then also $\bar{h}(\beta) <_{gp}^+ \bar{h}(\beta')$. In other words, if $\beta \in T_i^K(\alpha)$, then $\bar{h}(\beta) \in T_i^I(\bar{h}(\alpha))$. This means that, if $\bar{h}(\alpha) \neq \alpha_i^I$, then $T_i^K(\alpha) = T_{i+1}^K(\alpha)$. In this case, we also have that $T_i^I(\alpha) = T_{i+1}^I(\alpha)$. Hence, again, for such α our claim directly follows from the hypothesis.

Let us now concentrate on the only interesting case, where α is not too deep and is a mirror image of α_i^I (and, therefore, $\mathcal{K}_\alpha$

is a mirror image of $\mathcal{I}_{\alpha_i^I}$), which, formally speaking, means that $\bar{h}(\alpha) = \alpha_i^I$ and depth$(\alpha) = 0$ or depth$(\alpha) < \ell_\infty - i$.

Clearly, for such α, there exists exactly one atom which is in $T_{i+1}^I(\alpha)$ but not in $T_i^I(\alpha)$. This atom is β_i^I. It is also easy to see that $gp(\beta_i^I)$ (or, β_{gp}, for short) is somewhere in $T_i^I(\alpha)$, and, by hypothesis, there is an atom κ_{gp} somewhere in $T_i^K(\alpha)$ such that $\bar{h}(\kappa_{gp}) = \beta_{gp}$.

Now, suppose there is an active trigger (σ_i^I, h) on K_i such that $h(\gamma_i) = \kappa_{gp}$. Then the atom, call it κ_{new}, will appear in K_{i+1} as the result of this trigger, with $\bar{h}(\kappa_{new}) = \beta_i^I$, and it is an easy exercise to verify that the new function $\bar{h}$ will be indeed an isomorphism between $T_{i+1}^K(\alpha)$ and $T_{i+1}^I(\alpha)$.

Thus, the only thing that remains to be shown is that such an active trigger indeed exists. For that we need to show:

(A) All the sideatoms of κ_{gp} required by σ_i^I occur in K_i.

(B) The trigger (σ_i^I, h) for $\mathcal{T}$ on K_i is active.

For (A), suppose that π is a sideatom type of γ_i^I required by σ_i^I. We know that there is $\beta^\pi \in I_i$ such that $\beta^\pi \prec_{sp}^\pi \beta_{gp}$. If $\beta^\pi \in \mathcal{I}_{\alpha_i^I} = \mathcal{I}_{\bar{h}(\alpha)}$, then, by induction hypothesis, there is $\kappa^\pi \in \mathcal{K}_\alpha \cap K_i$ such that $\kappa^\pi \prec_{sp}^\pi \kappa$. But what if β^π is a remote side-parent?

Here is where the essence of the construction of D_{ac}, and of the Depth Condition, reveals itself. If $\beta^\pi \notin \mathcal{I}_{\alpha_i^I}$, then there exists $\alpha' \in D$ such that $\langle \alpha_i^I, \beta_i^I, \alpha', \beta^\pi \rangle$ is a remote-side-parent situation, and $\beta^\pi = \beta_j^I$ for some $j < i$. Thus, in (T_{ac}, λ) there is an edge (v, u) such that $\lambda(v) = \alpha$ and $\lambda(u) = \kappa'$, for some κ' with $\bar{h}(\kappa') = \alpha'$, as postulated in the Inductive Step of the construction of D_{ac}.

We know that either depth$(\alpha) \leq \ell_\infty - i$ or depth$(\alpha) > \ell_\infty - i$ but $\alpha = \alpha^\infty$. In both cases, by induction hypothesis, $\bar{h}$ is an isomorphism from $T_i^K(\kappa')$ to $T_i^I(\kappa')$. Let now κ^π be an element of $T_i^K(\kappa')$ such that $\bar{h}(\kappa^\pi) = \beta^\pi$. It follows from the fact that $\bar{h}$ is an isomorphism from $T_i^K(\alpha)$ and $T_i^I(\alpha)$, and from Lemma C.3(3), that $\bar{h}$ is also an isomorphism from $T_i^K(\alpha) \cup T_i^K(\kappa')$ to $T_i^I(\alpha) \cup T_i^I(\kappa')$ (notice that guardedness is crucial here). Hence, $\kappa^\pi \prec_{sp}^\pi \kappa_{gp}$.

For (B), assume that the trigger (σ_i^I, h) for $\mathcal{T}$ on K_i is not active. Thus, there is $\alpha_{bad} \in K_i$ such that $\alpha_{bad} \prec_s$ result(σ_i^I, h). We can then conclude that $\bar{h}(\alpha_{bad}) \prec_s \bar{h}(\text{result}(\sigma_i^I, h))$. This follows from the fact that, by claim (2) of Lemma C.3, $\bar{h}$ is a homomorphism from $\mathcal{K}$ to $\mathcal{I}$, and thus, if terms are equal in α_{bad} and result(σ_i^I, h), then they are not less equal in $\bar{h}(\alpha_{bad})$ and $\bar{h}(\text{result}(\sigma_i^I, h))$. But then (σ_i^I, h) for $\mathcal{T}$ on I_i is not active due to the atom $\bar{h}(\alpha_{bad}) \in I_i$, which is a contradiction. This concludes the proof of Lemma C.5. $\qquad \square$

Let us try to intuitively explain the above complicated lemma. Clearly, both $\mathcal{I}$ and $\mathcal{K}$ are forests (with $\prec_{gp}$ being the tree relation). The roots of the trees in $\mathcal{I}$ are atoms of D, while the roots of the trees in $\mathcal{K}$ are atoms of D_{ac}. Each atom in D_{ac} has its original atom in D, and $\bar{h}$ tells us which is this atom. Now, the second claim of the lemma (which looks simpler) states the following: at every stage of the construction of $\mathcal{K}$, the tree that has been constructed up to this point over the root $\alpha^\infty \in D_{ac}$, it is isomorphic to the tree that has been built over α^∞ up to the same point of the construction of $\mathcal{I}$. This is actually expected since the construction of $\mathcal{K}$ is exactly the construction of $\mathcal{I}$, but seen in a room full of distorting mirrors, and imagining that α^∞ is *the only* element of D_{ac} that is not a mirror

image, but the real atom. Regarding the first claim, as long as i is small enough, the situation is similar to the one in (2). The tree constructed in $\mathcal{K}$, until stage i, over the root $\alpha \in D_{ac}$ is isomorphic to the tree constructed in $\mathcal{I}$ until the same point in time over the root $\bar{h}(\alpha) \in D$. For some time we can see a faithful image of the construction, despite the fact that many mirror reflections are needed. But, when i is too large (compared to depth(α)) we can no longer see anything new. Notice that, in particular, if depth$(\alpha) = \ell_\infty$, the lemma states that no tree at all will be built over the root α.

Let us now state a useful corollary, which directly follows from Lemma C.5; for the proof of claim (2) recall that $\mathcal{I}_{\alpha^\infty}$ is infinite, while the proof of claim (5) uses claims (3) and (4).

CoROLLARY C.6. (1) $\bar{h}$ is an isomorphism from $\mathcal{K}_{\alpha^\infty}$ to $\mathcal{I}_{\alpha^\infty}$.
(2) The weakly restricted chase derivation $(K_i)_{i \geq 0}$ is infinite.
(3) For each atom $\alpha \in D_{ac}$ such that $\alpha \neq \alpha^\infty$, the mapping $\bar{h}$ is an isomorphism from $\mathcal{K}_\alpha$ to $\mathcal{I}_{\bar{h}(\alpha)} \cap I_{\ell_\infty - \text{depth}(\alpha)}$.
(4) For $\alpha, \alpha' \in D_{ac}$ with $\bar{h}(\alpha) = \bar{h}(\alpha')$ and depth$(\alpha) \leq$ depth(α'), there is a 1-1 homomorphism g from $\mathcal{K}_{\alpha'}$ to $\mathcal{K}_\alpha$ and $g(\alpha) = \alpha'$.
(5) Let $\beta, \beta' \in D_{ac}$ such that depth$(\beta) \leq$ depth(β'). For each $\alpha \in \mathcal{K}_\beta$ and $\alpha' \in \mathcal{K}_{\beta'}$ such that $\bar{h}(\alpha) = \bar{h}(\alpha')$, there exists a 1-1 homomorphism g from $\mathcal{K}_{\alpha'}$ to $\mathcal{K}_\alpha$ and $g(\alpha) = \alpha'$.

Step 3: An Infinite Restricted Chase Derivation

In this last step of the proof of the Treeification Theorem, our task is to extract from the infinite weakly restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$ constructed above, an infinite restricted chase derivation $(J_i)_{i \geq 0}$ of D_{ac} w.r.t. $\mathcal{T}$. For this, we first need to a fix a notation allowing us to directly address the atoms of $\mathcal{K}$.

Let $\mathcal{N}$ be the set of pairs of natural numbers defined as

$$\{(i, j) : i \geq 0 \text{ and } 0 \leq j < |K_{i+1} \setminus K_i|\}.$$

By $\leq$ and $<$ we denote the lexicographic ordering on $\mathcal{N}$. Note that $\langle \mathcal{N}, < \rangle$ and $\langle \mathbb{N}, < \rangle$ are isomorphic. Now, let $(\kappa_w)_{w \in \mathcal{N}}$ be an enumeration of all the atoms of $\mathcal{K}$ such that:

- $\kappa_{[i,j]} \in K_{i+1} \setminus K_i$, and
- if $\kappa_{[i,j]} \in \mathcal{K}_\alpha$ and $\kappa_{[i,j']} \in \mathcal{K}_\beta$, for some $j \leq j'$ and $\alpha, \beta \in D_{ac}$, then depth$(\alpha) \leq$ depth(β).

We now present a simple (not necessarily terminating) procedure, dubbed Extract$(\mathcal{K}, \mathcal{T})$, that extracts from $\mathcal{K}$ an infinite restricted chase derivation $(J_i)_{i \geq 0}$ of D_{ac} w.r.t. $\mathcal{T}$. This algorithm is depicted in the box above. It is clear that each time the while-loop is entered it holds that $J_m = Born$. It also follows by construction that:

LEMMA C.7. The sequence of instances $(J_i)_{i \geq 0}$ produced by Extract$(\mathcal{K}, \mathcal{T})$ is a restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$.

The crucial question is whether this sequence is infinite. A positive answer to this question will conclude the proof of the treeification theorem. The rest of the section is devoted to showing that:

LEMMA C.8. The sequence of instances $(J_i)_{i \geq 0}$ produced by Extract$(\mathcal{K}, \mathcal{T})$ is infinite.

We first show the following *loop invariant lemma*, that intuitively states the following: at each point of the execution of our iterative procedure, if an atom is not stopped, then there is a whole tuple of candidates that can act as its side-parents that are also not stopped.

```
Pending := 𝒦 \ K_0;
Born := K_0;
Stopped := ∅;
m := 0;
J_0 := K_0;
while Pending ≠ ∅ do
    let κ be the ≤-smallest element of Pending;
    Pending := Pending \ {κ};
    if there is an active trigger (σ, h) for 𝒯 on J_m such that
    κ = result(σ, h) then
        Born := Born ∪ {κ};
        J_{m+1} := J_m ∪ {κ};
        m := m + 1;
    else
        Stopped := Stopped ∪ {κ};
        foreach β ∈ Pending such that κ <_{gp}^+ β do
            Pending := Pending \ {β};
            Stopped := Stopped ∪ {β};

return (J_i)_{i≥0}.
```

LEMMA C.9 (**LOOP INVARIANT**). *Consider two atoms* $\alpha, \beta \in \mathcal{K}$ *such that* $\beta \prec_{sp}^\pi \alpha$, *for some sideatom type* π. *If* $\alpha \in Born \cup Pending$, *then there exists* $\beta' \in Born \cup Pending$ *such that* $\beta' \prec_{sp}^\pi \alpha$.

PROOF. We proceed by induction on the number of iterations of Extract($\mathcal{K}, \mathcal{T}$). Clearly, the loop invariant holds at the beginning of the execution. Suppose now that it holds at some point of the execution when we enter the while-loop. Let $\kappa_{[i,j]}$ be the current atom, i.e., the $\leq$-smallest atom of the set *Pending*.

It is easy to see that the guard-parent of $\kappa_{[i,j]}$ necessarily belongs to *Born*: it cannot be in *Pending* because then $\kappa_{[i,j]}$ would not be minimal in *Pending*, and it cannot be in *Stopped* because in such a case $\kappa_{[i,j]}$ would be in *Stopped* too. It is clear that, if there exists an active trigger for $\mathcal{T}$ on J_m that produces $\kappa_{[i,j]}$, then the set *Born* ∪ *Pending* remains unchanged since the algorithm will simply remove $\kappa_{[i,j]}$ from *Pending* and add it to *Born*. Thus, in this case the loop invariant holds.

Assume now that there is no such an active trigger. There are only two cases in which this can happen:

Case 1. Some of the side-parents $\kappa_{[i,j]}$ needs are not in *Born*. Assume that π' is a sideatom type required by the TGD $\sigma_i^{\mathcal{I}}$, due to which $\kappa_{[i,j]}$ has been generated in $\mathcal{K}$, and let $\kappa_{[i',j']}$ be any candidate from $\mathcal{K}$ such that $\kappa_{[i',j']} \prec_{sp}^{\pi'} \kappa_{[i,j]}$. It should be clear that $i' < i$. By construction, $\kappa_{[i',j']} \in Born \cup Stopped$ (since $\kappa_{[i,j]}$ is minimal in *Pending*). By induction hypothesis, there is, among these candidates, at least one that belongs to *Born* ∪ *Pending*. Hence, there must be at least one of those candidates in *Born*. This implies that this first case does not apply.

Case 2. There is an atom $\kappa_{[i',j']} \in J_m$ such that $\kappa_{[i',j']} \prec_s \kappa_{[i,j]}$. By an argument similar to that for statement (B) in the proof of Lemma C.5 above, we can show that in such a case it would be $\bar{h}(\kappa_{[i',j']}) \prec_s \bar{h}(\kappa_{[i,j]})$. Since $(I_i)_{i\geq 0}$ is a restricted chase derivation, we get that $i' \geq i$. But since $\kappa_{[i',j']} \in J_m$, we get that $i' = i$, and thus, $j' < j$. This means that $\bar{h}(\kappa_{[i,j]}) = \bar{h}(\kappa_{[i',j']})$, and $\kappa_{[i,j]} \in \mathcal{K}_\beta$ and

$\kappa_{[i',j']} \in \mathcal{K}_{\beta'}$ for some $\beta, \beta' \in D_{ac}$ such that depth(β') $\leq$ depth(β). By Corollary C.6, $\mathcal{K}_{\kappa_{[i,j]}}$ is isomorphic to a subset of $\mathcal{K}_{\kappa_{[i',j']}}$ via an isomorphism, let us say g, such that $g(\kappa_{[i,j]}) = \kappa_{[i',j']}$. Assume now that there are $\kappa, \kappa_\pi \in Born \cup Pending$ such that $\kappa_\pi \prec_{sp}^\pi \kappa$. We need to show that, after $\kappa_{[i,j]}$ gets stopped, together with all its $\prec_{gp}^+$-descendants, the loop invariant will still hold. Observe that if $\kappa_\pi \notin \mathcal{K}_{\kappa_{[i,j]}}$, then it is not affected by the removal of atoms of $\mathcal{K}_{\kappa_{[i,j]}}$. Moreover, if $\kappa \in \mathcal{K}_{\kappa_{[i,j]}}$, then it gets stopped, and there is nothing to show about its side-parents. Thus, the only case that we need to worry about is when $\kappa \notin \mathcal{K}_{\kappa_{[i,j]}}$ and $\kappa_\pi \in \mathcal{K}_{\kappa_{[i,j]}}$. In this case all the terms in κ_π occur in fr($\kappa_{[i,j]}$). Furthermore, since $\kappa_{[i',j']} \prec_s \kappa_{[i,j]}$, fr($\kappa_{[i,j]}$) = fr($\kappa_{[i',j']}$). Hence, the atom $g(\kappa_\pi)$, which is in $\mathcal{K}_{\kappa_{[i,j]}}$, and thus in *Born* ∪ *Pending* after the current iteration, is such that $g(\kappa_\pi) \prec_{sp}^\pi \kappa$, and the claim follows. □

In order to understand the meaning of this lemma, recall that $\mathcal{K}$, since it was produced by a weakly restricted chase, is a multiset, and there can be many atoms $\beta \in \mathcal{K}$ such that $\beta \prec_{sp}^\pi \alpha$. This is a phenomenon that never happens in a normal restricted chase. By exploiting the loop invariant lemma, we can show that none of the atoms of $\mathcal{K}_{\alpha^\infty}$ is stopped during our iterative procedure.

LEMMA C.10. *For each* $\alpha \in \mathcal{K}$, *if* $\alpha \in \mathcal{K}_{\alpha^\infty}$, *then* α *occurs in an instance of the sequence* $(J_i)_{i\geq 0}$ *produced by* Extract($\mathcal{K}, \mathcal{T}$).

PROOF. We need to show that $\alpha \notin Stopped$ for any $\alpha \in \mathcal{K}_{\alpha^\infty}$. Assume that there exists an atom of $\mathcal{K}_{\alpha^\infty}$ that belongs to *Stopped*; let $\hat{\alpha}$ be the $<$-smallest such atom. The loop invariant lemma (Lemma C.9) ensures that $\hat{\alpha}$ belongs to *Stopped* not because some of its side-parents are missing, but for a different reason; in fact, for one of the following two reasons:

(1) There exists an atom $\beta \in Stopped$ such that $\beta \prec_{gp}^+ \hat{\alpha}$. Clearly, $\beta \in \mathcal{K}_{\alpha^\infty}$ and also $\beta < \hat{\alpha}$. But this contradicts the fact that $\hat{\alpha}$ is the $<$-smallest atom of $\mathcal{K}_{\alpha^\infty}$ that has been stopped. Thus, this reason does not apply.

(2) There is $\beta \in Born$, with $\beta < \hat{\alpha}$ and $\beta \prec_s \hat{\alpha}$. Let $\beta = \kappa_{[i',j']}$ and $\hat{\alpha} = \kappa_{[i,0]}$; it follows from the definition of $(\kappa_w)_{w\in\mathcal{N}}$ that $\kappa_{[i,j]} \in \mathcal{K}_{\alpha^\infty}$ implies $j = 0$. Clearly, $[i',j'] < [i,0]$. However, since $\kappa_{[i',j']} \prec_s \kappa_{[i,0]}$, we get that $i' \geq i$. Thus, $j < 0$, which is not possible. Hence, also this reason does not apply.

Since none of the above cases apply, the claim follows. □

Having Lemma C.10, it is clear that Lemma C.8 follows. Indeed, since $\mathcal{K}_{\alpha^\infty}$ is infinite (Corollary C.6), and since each restricted chase step generates just one atom, we immediately get that the sequence $(J_i)_{i\geq 0}$ of instances produced by Extract($\mathcal{K}, \mathcal{T}$) is infinite. Therefore, $(J_i)_{i\geq 0}$ is an *infinite* restricted chase derivation of D_{ac} w.r.t. $\mathcal{T}$. This completes the proof of the Treeification Theorem.

C.3 Proof of Lemma 5.12

Let us assume, for the moment, that we have available the following auxiliary MSOL formulas (more details are given below); as usual, we use lower-case letters $x, y, \ldots$ for first-order variables,

and upper-case letters $A, B, \ldots$ for second-order variables:

$$\phi_{fin}(A) \quad\equiv\quad A \text{ is finite}$$
$$\phi_\pi(x, y) \quad\equiv\quad x \prec_{sp}^\pi y, \text{ for the sideatom type } \pi$$
$$\phi_b(x, y) \quad\equiv\quad x \prec_b^+ y.$$

By exploiting the above formulas, we can easily define $\phi_{\mathcal{T}}$ as the conjunction of the following four sentences:

(1) ϕ_{jt} checks whether T is an abstract join tree. It is easy to verify that all the conditions in the definition of abstract join trees (see Definition 5.8) are first-order expressible, apart from the first one, which states that the set $\{x \in V : \text{org}(x) = F\}$ is finite. For this check we exploit the MSOL formula ϕ_{fin}.

(2) ϕ_1 checks for the first condition of Definition 5.10 as follows:

$$\forall x \forall A (\forall y (\phi_b(y, x) \leftrightarrow y \in A) \rightarrow \phi_{fin}(A))$$

(3) ϕ_2 checks for the second condition; in what follows, we assume that σ has body $\alpha, \pi_1, \ldots, \pi_k$:

$$\forall x \forall y \left(x \prec y \wedge \text{org}(y) = \sigma \rightarrow \bigwedge_{i \in \{1, \ldots, k\}} \exists z \, \phi_{\pi_i}(z, y) \right)$$

Notice that $\text{org}(y) = \sigma$ is an abbreviation of a big disjunction that checks, via monadic predicates M_τ, where $\tau \in \Lambda_{\mathcal{T}}$, whether the label of y is of the form $\langle \cdot, \sigma, \cdot \rangle$.

(4) ϕ_3 checks for the third condition as follows:

$$\forall x \, \neg \phi_b(x, x)$$

We proceed to give more details about the auxiliary formulas used in $\phi_{\mathcal{T}}$. The formal definitions are omitted since they are long and tedious, but we give enough evidence that the formulas are indeed expressible in MSOL. Note that the following discussion heavily relies on the obvious fact below, which we will silently use:

FACT C.11. Let $T = \langle V, \prec \rangle$ be an abstract join tree. For each term t in $\Delta(T)$, $\{x \in V : t \text{ occurs in } \delta(x)\}$ induces a connected subtree of T.

Formula $\phi_{fin}(A)$. This formula comes from the general MSOL toolbox. It states that every infinite directed path B in T, starting from the root node of T, has an infinite directed sub-path, starting from some non-root element of B, which is disjoint with A.

Formula $\phi_=^{i,j}(x, y)$, for each $i, j \in \{1, \ldots, \text{ar}(\mathcal{T})\}$. Notice that these formulas have not been explicitly used above. However, they are needed for defining ϕ_π and ϕ_b. The formula $\phi_=^{i,j}(x, y)$ says that the term in $\delta(x)$ at position i is equal to the term in $\delta(y)$ at position j. This can be expressed in MSOL as follows: there is a set $A \subseteq V$ such that (i) A is a path with x and y being its ends, i.e., A is finite, x, y have exactly one neighbor in A, and any other node in A has exactly two neighbors, and (ii) A is a disjoint union of $A_1, \ldots, A_{\text{ar}(\mathcal{T})}$ such that $x \in A_i$, $y \in A_j$, and, for all pairs $z, w \in A$ such that $z \prec w$, $z \in A_k$, $w \in A_\ell$ it holds that $[[f, k], [m, \ell]] \in \text{eq}(w)$.

Formula $\phi_\pi(x, y)$. The formula says that $\delta(x) \subseteq_\pi \delta(y)$. It should be clear that it can be easily expressed by exploiting the formulas $\phi_=^{i,j}$ given above for checking whether terms in atoms are equal.

Formula $\phi_b(x, y)$. We first devise a formula $\psi_b(x, y)$, which states that $x \prec_b y$. Such a formula can be defined by using ϕ_π above, and also the formula $\phi_s(x, y) \equiv x \prec_s y$, which can be in turn defined by exploiting the formulas $\phi_=^{i,j}(x, y)$, for $i, j \in \{1, \ldots, \text{ar}(\mathcal{T})\}$.

Having ψ_b we can then devise a formula $\phi_{cl}(A)$, which states that A is $\prec_b$-downward closed, i.e., for each $x, y \in V$, with $x \prec_b y$ and $y \in A$ there is also $x \in A$.

Finally, $\phi_b(x, y)$ simply says that, for every $\prec_b$-downward closed set A it holds that $y \in A$ implies $x \in A$.

D PROOFS FROM SECTION 6

D.1 Proof of Lemma 6.11

Let us first establish an auxiliary claim, which essentially states that every free connected proto-caterpillar trivially satisfies condition (2) of Definition 6.3:

LEMMA D.1. Consider a free connected proto-caterpillar $\diamond = (L^\diamond, (\alpha_i^\diamond)_{i \geq 0}, (\sigma_i^\diamond, h_i^\diamond)_{i > 0}, (\gamma_i^\diamond)_{i > 0})$. For $\beta \in L^\diamond$ and $i > 0$, $\beta \nprec_s \alpha_i^\diamond$.

PROOF. Towards a contradiction, assume that there exists $\beta \in L^\diamond$ and $i > 0$ such that $\beta \prec_s \alpha_i^\diamond$. This implies that there exists a relay term $\mathfrak{c}$ of $\heartsuit$ occurring in $\text{fr}(\alpha_i^\diamond)$ that occurs also in β. Let $\alpha_j^\diamond$, for $j < i$, be the birth atom of $\mathfrak{c}$. In the special case where $\mathfrak{c}$ is the first relay term of $\diamond$, then $j = 0$. Assuming that $\mathfrak{c}'$ is the next relay term of $\diamond$ after $\mathfrak{c}$, let $k > i$ be such that $\alpha_k^\diamond$ is the birth atom of $\mathfrak{c}'$. By connectedness, $\mathfrak{c}$ occurs in $\text{fr}(\alpha_\ell^\diamond)$ for every $j < \ell \leq k$. Moreover, we know that there exists $j \leq \ell \leq k$ such that $\beta \prec_p \alpha_\ell^\diamond$. Since $\beta \neq \alpha_{\ell-1}^\diamond$ and $\diamond$ is free, we can conclude that the TGD $\sigma_\ell^\diamond$ (recall that the trigger $(\sigma_\ell^\diamond, h_\ell^\diamond)$ generates $\alpha_\ell^\diamond$), apart from the atom $\gamma_\ell^\diamond$, which is mapped by $h_\ell^\diamond$ to $\alpha_{\ell-1}^\diamond$, has another atom in its body that is mapped by $h_\ell^\diamond$ to β, while it shares a variable x with $\gamma_\ell^\diamond$ and $h_\ell^\diamond(x) = \mathfrak{c}$. Since $\mathcal{T}$ is a sticky set of TGDs, we conclude that $\mathfrak{c}$ occurs at an immortal position, which contradicts the fact that $\diamond$ is connected. □

We are now ready to give the proof of Lemma 6.11.

PROOF OF LEMMA 6.11. The fact that $\heartsuit$ is a connected proto-caterpillar follows from the fact that $\spadesuit$ is a connected proto-caterpillar (Lemma 6.10). In particular, by applying $\bar{h}$ on the atoms occurring in $\spadesuit$, there is no way to violate the conditions (1) - (3) of Definition 6.2, or the connectedness condition as defined in Definition 6.6. Moreover, it follows by construction that, for each $(\alpha, i), (\beta, j) \in \Pi(L^\heartsuit \cup B^\heartsuit)$, $\alpha[i] = \beta[j]$ implies $(\alpha, i) \simeq_{L^\heartsuit \cup B^\heartsuit}^* (\beta, j)$, and thus $\heartsuit$ is free. It remains to show that $\heartsuit$ enjoys the two conditions of Definition 6.3, which we recall here:

(1) for each $\beta \in L^\heartsuit$ and $i > 0$, $\beta \nprec_s \alpha_i^\heartsuit$, and
(2) for each $0 \leq i < j$, $\alpha_i^\heartsuit \nprec_s \alpha_j^\heartsuit$.

Since $\heartsuit$ is a free conected proto-caterpillar, (1) immediately follows from Lemma D.1.

For (2), towards a contradiction, assume that $\alpha_i^\heartsuit \prec_s \alpha_j^\heartsuit$ for some $0 \leq i < j$. Since we know that, if a term in $\alpha_i^\heartsuit$ is equal to a term in $\alpha_j^\heartsuit$, then the terms at the same positions in $\alpha_i^\spadesuit$ and $\alpha_j^\spadesuit$ are also equal, we get that $\alpha_i^\spadesuit \prec_s \alpha_j^\spadesuit$. This implies that $\alpha_{i+n}^\clubsuit \prec_s \alpha_{j+n}^\clubsuit$; recall, from the construction of $\spadesuit$, that n is such that $\alpha_n^\clubsuit$ is the birth atom of the relay term $\mathfrak{c}_{i_0}$. Therefore, there are atoms $\beta, \beta' \in \mathcal{I}$ such that $\beta \prec_p^+ \beta'$ and $\beta \prec_s \beta'$. But this contradicts the fact that $\beta \nprec_s \beta'$ since $(I_i)_{i \geq 0}$ is a restricted chase derivation. □

D.2 Proof of Lemma 6.12

The high-level idea of the construction is as follows. We first show that for an equality type $e = (R, E)$ from $\text{etypes}(\text{sch}(\mathcal{T}))$, and a set of positions $\Pi \subseteq \{1, \ldots, \text{ar}(R)\}$, we can build a deterministic Büchi automaton $\mathcal{A}_{e,\Pi}$ such that $L(\mathcal{A}_{e,\Pi}) \neq \emptyset$ iff there exists a free connected caterpillar such that its body starts with an atom of equality type e, and its first relay term occurs at positions Π of this atom. This means that a word $\mathbf{w}$ (which we will call caterpillar word and its over a finite alphabet $\Lambda_{\mathcal{T}}$) accepted by $\mathcal{A}_{e,\Pi}$ is actually a symbolic representation of a free connected caterpillar as the one above. Observe now that there are finitely many pairs (e, Π), where $e = (R, E)$ is an equality type of $\text{etypes}(\text{sch}(\mathcal{T}))$, and $\Pi \subseteq \{1, \ldots, \text{ar}(R)\}$; let $\text{etp}_{\mathcal{T}}$ be the set of all such pairs. Since Büchi automata are closed under union, i.e., given two Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, we can construct a Büchi automaton, denoted $\mathcal{A}_1 \cup \mathcal{A}_2$, that recognizes the language $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, the desired automaton is defined as the deterministic Büchi automaton

$$\mathcal{A}_{\mathcal{T}} = \bigcup_{(e,\Pi) \,\in\, \text{etp}_{\mathcal{T}}} \mathcal{A}_{e,\Pi}.$$

Thus, our main task in the remainder of the section is, for a pair $(e, \Pi) \in \text{etp}_{\mathcal{T}}$, to construct the Büchi automaton $\mathcal{A}_{e,\Pi}$.

Caterpillar Words and Automata

It is easy to see that a free proto-caterpillar is fully described (up to isomorphism, of course) by the equality type of the first atom α_0 of its body, and an infinite sequence of TGD-atom pairs $(\sigma_i, \gamma_i)_{i>0}$, which tells us which TGD of $\mathcal{T}$ should be used to produce the next atom of the proto-caterpillar's body, and which atom of the body of this TGD must match with the previous atom of the proto-caterpillar's body. The remaining atoms of the body of the TGD tell us which are the leg atoms of the proto-caterpillar.[7] Of course, not each such sequence translates to a free proto-caterpillar (as it may happen that some γ_{i+1} does not match with the i-the atom according to α_0 and the sequence $(\sigma_1, \gamma_1), (\sigma_2, \gamma_2), \ldots, (\sigma_i, \gamma_i))$, but if it does then the free proto-caterpillar is unique. However, there is no guarantee that this unique free proto-caterpillar is a connected caterpillar. In order to fully describe a free connected caterpillar we also need somehow to mark the pass-on points. This brings us to the notion of the caterpillar word (for $\mathcal{T}$).

We first define the finite alphabet $\Lambda_{\mathcal{T}}$, which consists of triples of the form (σ, γ, P), where $\sigma \in \mathcal{T}$, $\gamma \in \text{body}(\sigma)$, and, assuming that R is the predicate of $\text{head}(\sigma)$, $P \subseteq \{1, \ldots, \text{ar}(R)\}$ is such that $P \neq \emptyset$ implies there exists $i \in \{1, \ldots, \text{ar}(R)\}$ with $\text{head}(\sigma)[i] \notin \text{fr}(\sigma)$ and $P = \{j : \text{head}(\sigma)[i] = \text{head}(\sigma)[j]\}$. Then:

Definition D.2. A *caterpillar word* (for $\mathcal{T}$) is an infinite word $\mathbf{w} = w_1 w_2 \cdots$ such that, for each $i \geq 1$, $w_i \in \Lambda_{\mathcal{T}}$. ∎

Intuitively, a caterpillar word $\mathbf{w} = w_1 w_2, \cdots$, with $w_i = (\sigma_i, \gamma_i, P_i)$, is a candidate symbolic representation of a free connected caterpillar, where w_i marks a pass-on point iff P_i is non-empty. In fact, P_i indicates at which positions of $\text{head}(\sigma_i)$ the new relay term appears. Now, given a pair $(e_0, \Pi_0) \in \text{etp}_{\mathcal{T}}$, we say that $\mathbf{w}$ *encodes a free connected caterpillar starting at* (e_0, Π_0)[8] if the

---

[7] Notice that here we silently assume, w.l.o.g., that the proto-caterpillar is minimal in the sense that all the leg atoms participate in the generation of a body atom.

[8] We keep this definition semi-formal as the formal one is very tedious and it does not add any technical value to the proof.

---

sequence of TGD-atom pairs $(\sigma_i, \gamma_i)_{i>0}$ translates to a free connected caterpillar $\diamond = (L^{\diamond}, (\alpha_i^{\diamond})_{i \geq 0}, (\sigma_i^{\diamond}, h_i^{\diamond})_{i>0}, (\gamma_i^{\diamond})_{i>0})$, where (i) $\text{et}(\alpha_0^{\diamond}) = e_0$, and the first relay term of $\diamond$ occurs in $\alpha_0^{\diamond}$ at positions Π_0, (ii) $\sigma_i = \sigma_i^{\diamond}$ and $\gamma_i = \gamma_i^{\diamond}$, for each $i > 0$, and (iii) assuming that $b_1 < b_2 < \cdots$ are the pass-on points of $\diamond$, $P_i \neq \emptyset$ iff $i \in \{b_1, b_2, \ldots\}$, and the k-th relay term of $\diamond$ occurs in $\alpha_{b_k}^{\diamond}$ at positions P_k.

Recall that our goal is to construct a deterministic Büchi automaton $\mathcal{A}_{e_0,\Pi_0}$, with $\Lambda_{\mathcal{T}}$ being its alphabet, such that $L(\mathcal{A}_{e_0,\Pi_0})$ is exactly the set of caterpillar words that encode a free connected caterpillar starting at (e_0, Π_0). The automaton $\mathcal{A}_{e_0,\Pi_0}$ is defined as the (almost) cartesian product of three automata:

- $\mathcal{A}_{pc}$ that checks whether a caterpillar word $\mathbf{w}$ encodes a free proto-caterpillar such that e_0 is the equality type of the first atom of its body. Note that the set Π_0 does not play any role here. In fact, this automaton will only read the first two elements of each letter of $\mathbf{w}$.
- $\mathcal{A}_{qc}$ that checks whether a caterpillar word $\mathbf{w}$ that encodes a free proto-caterpillar $\diamond$ is such that $\diamond$ is a *quasi-caterpillar*, i.e., it satisfies condition (2) of Definition 6.3, that is, assuming that $(\alpha_i^{\diamond})_{i \geq 0}$ is the body of $\diamond$, $\alpha_i^{\diamond} \not\prec_s \alpha_j^{\diamond}$ for each $0 \leq i < j$. This automaton is quite involved, and as $\mathcal{A}_{pc}$ above, it will read only the first two elements of each letter of $\mathbf{w}$.
- $\mathcal{A}_{cc}$ that checks whether a caterpillar word $\mathbf{w}$ that encodes a free quasi-caterpillar $\diamond$ is such that $\diamond$ is connected. Since, by Lemma D.1, a free connected quasi-caterpillar is a caterpillar, $\mathcal{A}_{cc}$ essentially checks whether $\diamond$ is a free connected caterpillar, i.e., condition (1) of Definition 6.3 comes for free.

By "almost cartesian product" we mean that the states of $\mathcal{A}_{e_0,\Pi_0}$ will be triples, consisting of the states of $\mathcal{A}_{pc}$, $\mathcal{A}_{qc}$, and $\mathcal{A}_{cc}$. The transition functions of $\mathcal{A}_{pc}$ and $\mathcal{A}_{cc}$ will only depend on the current symbol of the word $\mathbf{w}$ and of the current state of the respective automaton. However, the transition function of $\mathcal{A}_{qc}$ will also use the current state of $\mathcal{A}_{pc}$ as part of its argument. Regarding the acceptance, each of the three automata will have a designated *reject* state. If any of them is ever encountered, then we assume that $\mathcal{A}_{e_0,\Pi_0}$ immediately rejects the input word $\mathbf{w}$. Apart from the *reject* state, $\mathcal{A}_{cc}$ will have an accepting state. The automaton $\mathcal{A}_{e_0,\Pi_0}$, which, as we said, is a Büchi automaton, will accept if $\mathcal{A}_{cc}$ will encounter this accepting state infinitely many times while reading the word $\mathbf{w}$. We can now describe the three automata in question.

The Automaton $\mathcal{A}_{pc}$

Since we are building a finite automaton, and there are infinitely many atoms in a caterpillar's body, there is no hope the automaton, after reading the symbol w_i of the input caterpillar word $\mathbf{w} = w_1 w_2 \cdots$, could "know" the i-th atom of the body of the proto-caterpillar encoded by $\mathbf{w}$. But it can know its equality type. We define the function $\delta_{\text{et}} : \text{etypes}(\text{sch}(\mathcal{T})) \times \Lambda_{\mathcal{T}} \to \text{etypes}(\text{sch}(\mathcal{T})) \cup \{reject\}$ as follows: for each $e \in \text{etypes}(\text{sch}(\mathcal{T}))$ and $(\sigma, \gamma, \cdot) \in \Lambda_{\mathcal{T}}$,

- if there is a homomorphism h that maps γ to $R(\star_1, \ldots, \star_n)$, with $\star_i = \star_j$ iff i, j coexist in a set of E, then $\delta_{\text{et}}(e, (\sigma, \gamma, \cdot)) = \text{et}(\bar{h}(\text{head}(\sigma)))$, where $\bar{h}$ is an extension of h that maps each existentially quantified variable in σ to a distinct symbol;
- otherwise, $\delta_{\text{et}}(e, (\sigma, \gamma, \cdot)) = reject$.

Let $\mathcal{A}_{pc}$ be a Büchi automaton with

- $\text{etypes}(\text{sch}(\mathcal{T})) \cup \{reject\}$ its set of states,

- $\Lambda_{\mathcal{T}}$ its alphabet,
- δ_{et} its transition function, and
- e_0 its initial state.

Clearly, for a caterpillar word $\mathbf{w}$, $\mathcal{A}_{pc}$ does not reject $\mathbf{w}$ iff $\mathbf{w}$ encodes a free proto-caterpillar $\diamond = (\cdot, (\alpha_i^\diamond)_{i \geq 0}, \cdot, \cdot)$ such that $\mathrm{et}(\alpha_0^\diamond) = e_0$; and if it does, then $\delta_{\mathrm{et}}(\mathrm{et}(\alpha_{i-1}^\diamond), w_i) = \mathrm{et}(\alpha_i^\diamond)$, for each $i > 0$.

The Automaton $\mathcal{A}_{qc}$

From now on we assume that an input caterpillar word $\mathbf{w}$ encodes a free proto-caterpillar $\diamond = (\cdot, (\alpha_i^\diamond)_{i \geq 0}, \cdot, \cdot)$ such that the first atom of its body has equality type e_0 (in case it does not encode such a free proto-caterpillar, $\mathcal{A}_{pc}$ will take care of it), and proceed towards checking whether $\diamond$ is a quasi-caterpillar, i.e., whether it satisfies condition (2) of Definition 6.3, that is, $\alpha_i^\diamond \nprec_s \alpha_j^\diamond$ for each $0 \leq i < j$.

Given a finite set $\mathfrak{T}$ of terms, a $\mathfrak{T}$-*equality type* over a schema $\mathbf{S}$ is essentially an equality type (R, E) over $\mathbf{S}$ where, in addition, some of the sets of E are labeled with distinct terms of $\mathfrak{T}$ indicating that a term $t \in \mathfrak{T}$ should occur at certain positions. Formally, a $\mathfrak{T}$-*equality type* over $\mathbf{S}$ is a triple (R, E, λ), where $(R, E) \in \mathrm{etypes}(\mathbf{S})$, and λ is a partial injective function from E to $\mathfrak{T}$. It is clear that there are only finitely many $\mathfrak{T}$-equality types over $\mathbf{S}$. The $\mathfrak{T}$-equality type of an atom α, denoted $\mathrm{et}_{\mathfrak{T}}(\alpha)$, as well as the canonical atom of a $\mathfrak{T}$-equality type e, denoted $\mathrm{can}(e)$, are defined in the expected way.

Now, for brevity, let $\mathfrak{T}_j$ be the set of terms occurring in $\alpha_j^\diamond$. We can easily show the following useful lemma:

LEMMA D.3. *Suppose $i < j < k$ for some $i, j, k \geq 0$. It holds that $\alpha_i^\diamond \prec_s \alpha_k^\diamond$ iff $\mathrm{can}(\mathrm{et}_{\mathfrak{T}_j}(\alpha_i^\diamond)) \prec_s \alpha_k^\diamond$.*

PROOF. First, for each set of terms $\mathfrak{T}$ that contains all the terms occurring both in $\alpha_i^\diamond$ and $\alpha_k^\diamond$, it is easy to show that $\alpha_i^\diamond \prec_s \alpha_k^\diamond$ iff $\mathrm{can}(\mathrm{et}_{\mathfrak{T}}(\alpha_i^\diamond)) \prec_s \alpha_k^\diamond$. Now, since $\diamond$ is free, if some term occurs in $\alpha_i^\diamond$ and $\alpha_k^\diamond$, then it must also occur in $\alpha_j^\diamond$. This implies that $\mathfrak{T}_j$ contains all the terms occurring both in $\alpha_i^\diamond$ and $\alpha_k^\diamond$, and the claim follows. $\square$

For each $j \geq 0$, let $\Theta_j = \{\mathrm{et}_{\mathfrak{T}}(\alpha_i^\diamond) : 0 \leq i \leq j\}$. Of course, Θ_j is a finite set, for each $j \geq 0$. Moreover, if we just know $\mathrm{et}(\alpha_j^\diamond)$, then the number of possible candidates for Θ_j is finite, and uniformly bounded, so Θ_j can be seen as a finite piece of information, or as (part of) a state of a finite automaton. It is possible to construct Θ_{j+1} only knowing Θ_j, $\mathrm{et}(\alpha_j^\diamond)$ and the $(j+1)$-th symbol w_{j+1} of $\mathbf{w}$. Furthermore, knowing Θ_j and $\mathrm{et}(\alpha_j^\diamond)$, we can check whether there is $0 \leq i < j$ such that $\alpha_i^\diamond \prec_s \alpha_j^\diamond$; the latter is a consequence of Lemma D.3. Thus, we can define a function δ_Θ such that:

$$\delta_\Theta \left((\Theta_j, \mathrm{et}(\alpha_j^\diamond)), (\sigma_{j+1}, \gamma_{j+1}, \cdot) \right)$$

$$= \begin{cases} reject & \text{if } \alpha_i^\diamond \prec_s \alpha_j^\diamond \text{ for some } i < j, \\ (\Theta_{j+1}, \mathrm{et}(\alpha_{j+1}^\diamond)) & \text{otherwise.} \end{cases}$$

Let $\mathcal{A}_{qc}$ be a Büchi automaton with

- its set of states consisting of pairs of the form (Θ, e) as above, and the *reject* state,
- $\Lambda_{\mathcal{T}}$ its alphabet,
- δ_Θ its transition function, and
- $(\emptyset, e_0)$ its initial state.

By construction, for a caterpillar word $\mathbf{w}$, $\mathcal{A}_{qc}$ does not reject $\mathbf{w}$ iff $\mathbf{w}$ encodes a free proto-caterpillar $\diamond$ that satisfies condition (2) of Definition 6.3, i.e., $\diamond$ is a quasi-caterpillar.

The Automaton $\mathcal{A}_{cc}$

First, we need to define a function δ_{pos} that will let the automaton remember some terms. Given $\Pi \subseteq \{1, \ldots, \mathrm{ar}(\mathcal{T})\}$ and $w = (\sigma, \gamma, P) \in \Lambda_{\mathcal{T}}$, let $\delta_{pos}(\Pi, w)$ be the set of integers

$$\{i \in \{1, \ldots, \mathrm{ar}(\mathcal{T})\} : \text{ there is } j \in \Pi \text{ such that } \gamma[j] = \mathrm{head}(\sigma)[i]\}.$$

The purpose of δ_{pos} will be made clear in a while.

The states of $\mathcal{A}_{cc}$ will be tuples (Π_1, Π_2, q), where $\Pi_1, \Pi_2 \subseteq \{1, \ldots \mathrm{ar}(\mathcal{T})\}$ and $q \in \{\top, \bot\}$. Roughly, Π_1 will remember the positions where the current relay term appears (we need this information since we must make sure that the current relay term survives until the next pass-on point), and Π_2 will remember the positions where all the relay terms, current and older ones, live at the given moment (we need this to make sure that they will never appear at an immortal position).

We proceed to define the function δ_{cc} as follows: given a state-symbol pair (s, w) with $s = (\Pi_1, \Pi_2, q)$ and $w = (\sigma, \gamma, P)$:

- if $\delta_{pos}(\Pi_1, w) = \emptyset$, or there exists $i \in \delta_{pos}(\Pi_2, w)$ such that $\mathrm{head}(\sigma)[i]$ is not marked in $\mathcal{T}$, then $\delta_{cc}(s, w) = reject$;
- otherwise, $\delta_{cc}(s, w) = (\delta_{pos}(\Pi_1, w), \delta_{pos}(\Pi_2, w), \bot)$ if $P = \emptyset$, and $\delta_{cc}(s, w) = (P, \delta_{pos}(\Pi_1, w) \cup \delta_{pos}(\Pi_2, w), \top)$ if $P \neq \emptyset$.

Intuitively, this means that if we are not at a pass-on point ($P = \emptyset$), then keep track of the positions occupied by the current and the old relay terms. On the other hand, if we are at a pass-on point ($P \neq \emptyset$), then forget the positions occupied by the old relay terms and remember the positions at which the new one occurs. But do not forget them completely; simply add them to the set of positions where all the relay terms appear.

Let $\mathcal{A}_{cc}$ be a Büchi automaton with

- its set of states consisting of triples as described above, and the *reject* state,
- $\Lambda_{\mathcal{T}}$ its alphabet,
- δ_{cc} its transition function,
- $(\Pi_0, \emptyset, e_0)$ its initial state, and
- all the states of the form $(\cdot, \cdot, \top)$ being accepting.

D.3 Proof of Lemma 6.13

Let $\diamond = (L^\diamond, B^\diamond, T^\diamond, G^\diamond)$ be a free uniformly connected caterpillar, where $B^\diamond = (\alpha_i^\diamond)_{i \geq 0}$, $T^\diamond = (\sigma_i^\diamond, h_i^\diamond)_{i > 0}$, and $G^\diamond = (\gamma_i^\diamond)_{i > 0}$. Let $\mathfrak{c}_0, \mathfrak{c}_1, \ldots$ be the relay terms of $\diamond$. Moreover, assuming that $(b_i)_{i > 0}$ are the pass-on points of $\diamond$, let $d \geq 0$ be such that $b_{k+1} - b_k < d$, for each $k \geq 0$. Our goal is to define a unifying function h for $\diamond$ such that $h(\diamond)$ is a finitary caterpillar, which means that $h(L^\diamond)$ is finite.

We first observe that no matter how a unifying function h for $\diamond$ is defined, $h(\diamond)$ is a proto-caterpillar that satisfies condition (1) of Definition 6.3. This is what the next lemma tells us:

LEMMA D.4. *Consider a unifying function h for $\diamond$. Then:*

(1) $h(\diamond)$ is a proto-caterpillar, and
(2) for each $\beta \in h(L^\diamond)$ and $i > 0$, $\beta \nprec_s h(\alpha_i^\diamond)$.

PROOF. It is clear that there is no way to violate the conditions given in Definition 6.2 by unifying terms in the legs of a proto-caterpillar. Since, by hypothesis, $\diamond$ is a proto-caterpillar, we get that $h(\diamond)$ is a proto-caterpillar, and (1) follows.

Concerning (2), the claim follows by the fact that none of the relay terms $c_0, c_1, \ldots$ of $\diamond$ occurs in $L^\diamond$. This implies that none of the terms $c_0, c_1, \ldots$ occurs in $h(L^\diamond)$, while, for each $i \geq 0$, $\mathrm{fr}(h(\alpha_i^\diamond))$ contains a term from $c_0, c_1, \ldots$. Therefore, none of the atoms of $h(L^\diamond)$ can stop an atom of $(h(\alpha_i^\diamond))_{i>0}$, and the claim follows. $\qquad\square$

Having the above lemma in place, it is clear that to establish Lemma 6.13 it remains to construct a unifying function h for $\diamond$ such that $h(L^\diamond)$ is finite, and $h(\diamond)$ satisfies condition (2) of Definition 6.3. The rest of the section is devoted to constructing such a function.

We first define the domain of the desired function as the set of terms $\mathfrak{B} \subseteq \mathrm{dom}(L^\diamond)$ that occur at a position (α, i), for some atom $\alpha \in L^\diamond$, that is not related to any immortal position (β, j), where $\beta \in B^\diamond$. Notice that none of the relay terms of $\diamond$ occur in $\mathfrak{B}$. It would be useful to be able to refer to the terms of $\mathfrak{B}$ that participate in the generation of the atoms between the first body atom and the first pass-on point, as well as the atoms between two consecutive pass-on points. Let $B_0^\diamond = \{\alpha_j^\diamond : 0 < j \leq b_1\} \subseteq B^\diamond$, that is, the set of atoms between $\alpha_0^\diamond$ and the birth atom of c_1. Moreover, for each $i > 0$, let $B_i^\diamond = \{\alpha_j^\diamond : b_i < j \leq b_{i+1}\} \subseteq B^\diamond$, that is, the set of atoms between the birth atom of c_i and the birth atom of c_{i+1}. We also define, for each $i \geq 0$, $L_i^\diamond$ as the set of atoms

$$\big\{\alpha \in L^\diamond : \text{ there exists } j > 0 \text{ such that}$$
$$\mathrm{result}(\sigma_j^\diamond, h_j^\diamond) \in B_i^\diamond, \text{ and } \alpha \in h_j^\diamond(\mathrm{body}(\sigma_j^\diamond))\big\},$$

which are essentially the atoms that are needed to generate $B_i^\diamond$. Then, we let $\mathfrak{B}_i = \mathfrak{B} \cap \mathrm{dom}(L_i^\diamond)$.

For the codomain we need a sufficiently large *finite* set of new terms. Let $m_0 \geq 0$ be greater than the maximal number of variables in a TGD of $\mathcal{T}$ and $m = (d+1) \cdot m_0$; recall that d is the uniform distance between two consecutive pass-on points of $\diamond$. We define $\mathfrak{T}$, which will be the codomain of the desired function, as a set of terms such that $|\mathfrak{T}| = 2m$ and $\mathfrak{T} \cap \mathrm{dom}(L^\diamond \cup B^\diamond) = \emptyset$, i.e., $\mathfrak{T}$ collects m new terms that do not occur in $L^\diamond \cup B^\diamond$. We can then show the following key technical lemma:

LEMMA D.5. *There exists a unifying function $\hbar : \mathfrak{B} \to \mathfrak{T}$ such that, for each $i \geq 0$, the unifying function $\hbar_{|\mathfrak{B}_i}$ is 1-1.*

PROOF. We first observe that:

(*) for each $i \geq 0$, $|\mathfrak{B}_i| \leq m$ – this is a consequence of the definition of m;
(**) for each $i > 0$, $\mathfrak{B}_i \cap \bigcup_{j<i} \mathfrak{B}_j \subseteq \mathfrak{B}_{i-1}$ – since all the terms that occur both in $\bigcup_{j<i} \mathfrak{B}_j$ and in $\mathfrak{B}_i$ must also occur in $\alpha_{b_i}^\diamond$.

We are going to build an ascending sequence $(\hbar_i)_{i\geq0}$ of functions, where $\hbar_i : \bigcup_{j\leq i} \mathfrak{B}_j \to \mathfrak{T}$, such that, for each $i \geq 0$, the function $\hbar_{i|\mathfrak{B}_i}$ is 1-1. Then, $\hbar$ will be defined as $\bigcup_{i\geq0} \hbar_i$.

Let $\hbar_0$ be an 1-1 function of the form $\mathfrak{B}_0 \to \mathfrak{T}$. Notice that such a function exists since, by definition, $\mathfrak{T}$ is sufficiently large. Suppose now that $\hbar_{i-1}$, as specified above, has been defined. In order to define $\hbar_i$ we need to extend $\hbar_{i-1}$ to the terms in $\mathfrak{B}_i \setminus \bigcup_{j<i} \mathfrak{B}_j$ in such a way that the newly defined function is 1-1 on $\mathfrak{B}_i$. From

(**) we know that $\mathfrak{B}_i \setminus \bigcup_{j<i} \mathfrak{B}_j = \mathfrak{B}_i \setminus \mathfrak{B}_{i-1}$, and, by assumption, $\hbar_{i-1|\mathfrak{B}_{i-1}}$ is 1-1. This means that on the subset of $\mathfrak{B}_i$ where $\hbar_i$ is already defined (since $\hbar_{i-1}$ is defined) it is 1-1. Now, to be able to extend it to an 1-1 function on the entire set $\mathfrak{B}_i$ we need to have enough terms in the codomain, which is guaranteed by (*). $\quad\square$

Let $\hbar$ be the unifying function for $\diamond$ provided by Lemma D.5. We proceed to show that:

LEMMA D.6. (1) *The instance $\hbar(L^\diamond)$ is finite.*
(2) *For each $0 \leq i < j$, $\hbar(\alpha_i^\diamond) \not\prec_s \hbar(\alpha_j^\diamond)$.*

PROOF. For (1), since $\mathfrak{B}$ collects all the terms of $\mathrm{dom}(L^\diamond)$ that occur at a position (α, i), for some $\alpha \in L^\diamond$, that is not related to any immortal position (β, j), where $\beta \in B^\diamond$, we can conclude that $\mathrm{dom}(L^\diamond) \setminus \mathfrak{B}$ is finite. Therefore, $\mathrm{dom}(\hbar(L^\diamond))$ is finite, which in turn implies that $\hbar(L^\diamond)$ is finite, as needed.

For (2), we proceed by considering the following two cases:

- $i \leq b_k < j$ for some $k > 0$. In this case, there exists $\ell \geq k$ such that the relay term c_ℓ occurs in $\mathrm{fr}(\alpha_j^\diamond)$ but not in $\mathrm{fr}(\alpha_i^\diamond)$. Since none of the relay terms of $\diamond$ occurs in the domain or the codomain of $\hbar$, we conclude that c_ℓ occurs in $\mathrm{fr}(\hbar(\alpha_j^\diamond))$ but not in $\mathrm{fr}(\hbar(\alpha_i^\diamond))$, which implies that $\hbar(\alpha_i^\diamond) \not\prec_s \hbar(\alpha_j^\diamond)$.
- $b_k \leq i < j \leq b_{k+1}$ for some $k \geq 0$ (with $b_0 = 0$.) Since, by hypothesis, $\diamond$ is a caterpillar, we get that $\alpha_i^\diamond \not\prec_s \alpha_j^\diamond$. The fact that $\hbar$ is an 1-1 function over $\mathfrak{B}_k$ allows us to conclude that $\{\alpha_i^\diamond, \alpha_j^\diamond\}$ is isomorphic to $\{\hbar(\alpha_i^\diamond), \hbar(\alpha_j^\diamond)\}$. Therefore, $\hbar(\alpha_i^\diamond) \not\prec_s \hbar(\alpha_j^\diamond)$, and the claim follows.

This completes the proof of the lemma. $\quad\square$

By Lemma D.4 and D.6, we immediately get Lemma 6.13.