All-Instances Oblivious Chase Termination is Undecidable for Single-Head Binary TGDs*

Bartosz Bednarczyk^{1,2} , **Robert Ferens**^{2} , **Piotr Ostropolski-Nalewaja**^{2}
^{1}Computational Logic Group, TU Dresden
^{2}Institute of Computer Science, University of Wrocław

Abstract

The *chase* is a famous algorithmic procedure in database theory with numerous applications in ontology-mediated query answering. We consider static analysis of the chase termination problem, which asks, given a set of TGDs, whether the chase terminates on all input databases. The problem was recently shown to be undecidable by Gogacz et al. for sets of rules containing only ternary predicates. In this work, we show that undecidability occurs already for sets of single-head TGD over binary vocabularies. This question is relevant since many real-world ontologies, e.g., those from the Horn fragment of the popular OWL, are of this shape.

1 Introduction

The *chase* [Maier *et al.*, 1979] is a fundamental family of algorithms used to solve issues involving the *tuple-generating dependencies* [Fagin, 2018] TGDs, e.g., checking query containment under constraints [Beeri and Vardi, 1984; Maier *et al.*, 1979], computing data-exchange solutions [Fagin *et al.*, 2005], querying databases with views [Halevy, 2001] or querying probabilistic databases [Olteanu *et al.*, 2009]. Moreover, the chase is intensively used in ontological modelling and knowledge representation [Baget *et al.*, 2011; Calì *et al.*, 2010; Grau *et al.*, 2013], especially for answering conjunctive queries over ontologies formulated with TGDs: see e.g. [Benedikt *et al.*, 2017; Urbani *et al.*, 2018] for practical implementations and benchmarks. The main idea behind the chase is fairly simple: given a database $\mathbb{D}$ and a set of TGDs $\mathcal{T}$ the chase computes a universal model of $\mathbb{D}$ and $\mathcal{T}$ i.e., a possibly infinite extension of $\mathbb{D}$, which is self-sufficient to determine query entailment. Whilst TGDs have a precise definition i.e., they are first-order sentences of the form $\forall \overline{x}\ \forall \overline{y}\ \alpha(\overline{x}, \overline{y}) \rightarrow \exists \overline{z}\ \beta(\overline{y}, \overline{z})$, where both α and β are positive conjunctions of atoms, the way how the chase constructs a universal model can be formalized in various ways. It results in several versions of the chase, developed during the last decade: the Oblivious Chase [Calì *et al.*, 2013], the Skolem Chase [Marnette, 2009], the Semi-Oblivious Chase [Marnette, 2009], the Standard Chase [Fagin *et al.*, 2005]. In this paper however, we focus on the *Oblivious Chase*, which is an eager (or a naive) version of the chase. Roughly speaking, for each TGD $\forall \overline{x}\ \forall \overline{y}\ \alpha(\overline{x}, \overline{y}) \rightarrow \exists \overline{z}\ \beta(\overline{y}, \overline{z})$ from a set $\mathcal{T}$ the Oblivious $\mathcal{T}$-Chase extends an initial database $\mathbb{D}$ with fresh elements $\overline{z}$ witnessing $\beta(\overline{y}, \overline{z})$ whenever it is possible to find tuples $\overline{x}, \overline{y}$ witnessing $\alpha(\overline{x}, \overline{y})$. The chasing process continues until a fix point is reached. If such fix-point structure is finite, we say that the chase is *terminating*.

A natural question arising from chase termination is *the All-Instances Chase Termination Problem* (AICTP): a static analysis variant of chase termination, where we ask whether for a given set $\mathcal{T}$ of TGDs the $\mathcal{T}$–Chase terminates on all possible databases $\mathbb{D}$ [Grahne and Onet, 2018]. The problem was recently shown to be undecidable in [Gogacz and Marcinkowski, 2014] for sets of ternary rules, no matter what version of the chase we choose. Since the TGDs used in their undecidability proof are far from any well-known and well-behaved rule classes, researchers started looking on AICTP for rule-sets under some shape restrictions [Calautti *et al.*, 2015]. Recently the decidability of AICTP has been shown for sets of linear TGDs [Leclère *et al.*, 2019], sets of single-head guarded TGDs [Gogacz *et al.*, 2020], and for sets of sticky TGDs [Calautti and Pieris, 2019].

1.1 Our Contribution

In this work, we consider the All-Instances Chase Termination Problem for the class of TGDs under two restrictions: first, the allowed arity of relations in TGDs is at most two and second, all rules contain at most one atom in their heads. Such restrictions are very natural, e.g. for ontologies from the Horn fragment of OWL [Hitzler *et al.*, 2009]. More precisely we are going to prove the following theorem:

Theorem 1.1. *All-Instances Oblivious Chase Termination is undecidable for sets of binary single-head TGDs.*

Our result significantly improves the exponential space lower bound provided in [Gogacz and Marcinkowski, 2014]. Moreover, by arguing along the lines of [Gogacz and Marcinkowski, 2014, Sec. 3.5] our proofs can be extended to the undecidability proof of AICTP for binary single-head TGDs for both Standard and Semi-Oblivious Chases.

*B. Bednarczyk was supported by ERC Consolidator Grant 771779 (DeciGUT). P. Ostropolski-Nalewaja was supported by the Polish National Science Centre (NCN) grant 2016/23/B/ST6/01438.

2 Preliminaries

We consider a set of terms $\mathbf{T}$, defined as the union of three countably-infinite mutually-disjoint sets of *constants* $\mathbf{C}$, of *nulls* $\mathbf{N}$, and of *variables* $\mathbf{V}$. A *schema* $\mathbb{S}$ is a finite set of relational symbols. For each relational symbol $R \in \mathbb{S}$ we denote its *arity* with $\mathsf{ar}(R)$. An *atom* over $\mathbb{S}$ is an expression of the form $R(\bar{t})$, where $\bar{t}$ is a $\mathsf{ar}(R)$-tuple of terms. A *fact* is simply an atom $R(\bar{t})$, where $\bar{t}$ consists only of constants. An *instance* $\mathfrak{I}$ over $\mathbb{S}$ is a (possibly infinite) set of atoms, whereas a *database* $\mathbb{D}$ is a finite set of facts. The *active domain* $\mathsf{adom}(\mathfrak{I})$ of an instance $\mathfrak{I}$ is the set of all *domain elements* from relations of $\mathfrak{I}$, i.e., both constants and nulls. Since databases and instances are structures in a sense of mathematical logic, we use satisfaction relation $\models$ to indicate that a structure satisfies a given formula. Moreover, we note here that in Section 3 we work on schemata containing relational symbols of arity at most two only. Thus we often refer to binary relations simply as *edges* and to domain elements of instances as *nodes* or *vertices*.

A *tuple-generating dependency* (TGD) is a sentence of the form $\varphi = \forall \overline{xy}\ \alpha(\overline{x}, \overline{y}) \rightarrow \exists \overline{z}\ \beta(\overline{y}, \overline{z})$, where $\overline{x}, \overline{y}, \overline{z}$ are tuples of variables from $\mathbf{V}$ and both the *body* $\alpha(\overline{x}, \overline{y})$ and the *head* $\beta(\overline{y}, \overline{z})$ of φ (denoted with $\mathsf{body}(\varphi)$ and $\mathsf{head}(\varphi)$ respectively) are positive conjunctions of atoms. For the sake of readability, we often omit quantifiers in TGDs. We say that a TGD φ is: *n-ary* if all relations appearing in φ are of arity at most n, *single-head* if $\mathsf{head}(\varphi)$ contains at most one atom, *existential* if φ contains an existential quantifier, and *datalog* if φ does not contain an existential quantifier.

A *substitution* is a partial function defined over $\mathbf{T}$. 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, which additionally satisfies conditions: (i) $h(t) = t$ for each $t \in \mathbf{C}$, and (ii) $R(h(t_1), \ldots, h(t_n)) \in B$ for all $R(t_1, \ldots, t_n) \in A$.

An instance $\mathfrak{I}$ is *contained* in $\mathfrak{I}'$, (written $\mathfrak{I} \subseteq \mathfrak{I}'$), when there exists a injective homomorphism from $\mathfrak{I}$ to $\mathfrak{I}'$. For two instances $\mathfrak{I}$ and $\mathfrak{I}'$ and schemata $\mathbb{S}' \subseteq \mathbb{S}$ we write $\mathfrak{I} \equiv_{\mathbb{S}'} \mathfrak{I}'$ when $\mathfrak{I}$ and $\mathfrak{I}'$ are isomorphic when restricted to relations from $\mathbb{S}'$.

2.1 The Chase

For a given set $\mathcal{T}$ of TGDs and an input instance $\mathfrak{I}$, the *Oblivious $\mathcal{T}$-Chase* (or simply *$\mathcal{T}$-Chase* from now on) produces the universal model of $\mathcal{T}$ and $\mathfrak{I}$, i.e., a model that can be homomorphically embedded in any other model of $\mathcal{T}$ and $\mathfrak{I}$. The construction is done by exhaustively applying so-called *triggers* on the intermediate instances constructed so far.

A *$\mathcal{T}$-trigger* on an instance $\mathfrak{I}$ is a pair (φ, h) composed of a TGD φ from $\mathcal{T}$ and a homomorphism $h : \mathsf{body}(\varphi) \rightarrow \mathfrak{I}$. An *application* of a trigger (φ, h) to $\mathfrak{I}$ returns a fresh instance[1] $\mathfrak{I}' = \mathfrak{I} \cup \{h'(\mathsf{head}(\varphi))\}$, where the homomorphism $h' \supseteq h$ maps each existentially quantified variable from $\mathsf{head}(\varphi)$ to a fresh null from $\mathbf{N}$. An application of such trigger is denoted with $\mathfrak{I}(\varphi, h)\mathfrak{I}'$.

We say that a (possibly infinite) sequence of instances $\mathfrak{I}_0, \mathfrak{I}_1, \ldots$ (called *intermediate instances*) is a *$\mathcal{T}$-chase sequence* whenever it satisfies:

[1]Note that we identify formulae with structures and vice versa.

- for each index i there exists a $\mathcal{T}$-trigger (φ_i, h_i) such that $\mathfrak{I}_i(\varphi_i, h_i)\mathfrak{I}_{i+1}$ holds, and
- for all indices $i < j$ once the trigger applications $\mathfrak{I}_i(\varphi_i, h_i)\mathfrak{I}_{i+1}$ and $\mathfrak{I}_j(\varphi_j, h_j)\mathfrak{I}_{j+1}$ are given, the inequality $(\varphi_i, h_i) \neq (\varphi_j, h_j)$ holds, and
- for each i if there exists a trigger (φ, h) on instance $\mathfrak{I}_i$ then there exists j such that $(\varphi_j, h_j) = (\varphi, h)$.

Given a $\mathcal{T}$-Chase sequence $\mathfrak{I}_0, \mathfrak{I}_1, \ldots$ we define its *result* as a union $\bigcup_{i=0}^{\infty} \mathfrak{I}_i$. It is well-known [Grahne and Onet, 2018] that the order of application of triggers does not matter in Oblivious Chase: it leads eventually to the same structure (finite or not), up to renaming of the nulls. Henceforth, we denote with $\mathsf{Ch}(\mathcal{T}, \mathbb{D})$ the *result* of the $\mathcal{T}$-Chase procedure on $\mathbb{D}$ (which is equal to the result of any $\mathcal{T}$-Chase sequence that is starting from $\mathbb{D}$). We say that the $\mathcal{T}$-Chase on $\mathbb{D}$ is *terminating* when the resulting instance is finite.

2.2 Undecidability of AIOCTP

In the *All-Instances Oblivious Chase Termination Problem* (abbreviated as AIOCTP) we ask if, for given set of TGDs $\mathcal{T}$, the $\mathcal{T}$-Chase terminates on all possible databases $\mathbb{D}$. It was recently shown in [Gogacz and Marcinkowski, 2014]:

Theorem 2.1. *The AIOCTP is undecidable.*

Theorem 2.2. *The AIOCTP for the class of binary single-head TGDs is* ExpSpace*-hard.*

Moreover, it was observed by Marnette [Marnette, 2009] that to decide all-instance termination for the oblivious $\mathcal{T}$-Chase, it is sufficient to restrict the problem to the termination on a very specific database called the *critical instance*. For a given schema $\mathbb{S}$, the critical instance $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$ is simply a database containing a single constant s, called the *source*, and all atoms of the form $R(s, s, \ldots, s)$ where $R \in \mathbb{S}$. The main property of the critical instance is presented below:

Lemma 2.3 ([Marnette, 2009]). *For any set of TGDs $\mathcal{T}$ over a schema $\mathbb{S}$, the $\mathcal{T}$-Chase terminates on all databases $\mathbb{D}$ if and only if the $\mathcal{T}$-Chase terminates on the critical instance $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$.*

Now we present the main undecidable problem used in our reduction, namely the *boundedness of Conway functions*. Fix a natural number $n \in \mathbb{N}$. Let π_n be a product of n prime numbers and let $q_0, q_1, \ldots, q_{\pi_n - 1}, r_0, r_1, \ldots, r_{\pi_n - 1}$ be natural numbers such that for every $m \in \mathbb{N}$ congruent to j modulo π_n the fraction $m \cdot \frac{q_j}{r_j}$ is a natural number.

The *Conway function* $g : \mathbb{N} \rightarrow \mathbb{N}$ is defined as:

$$g(m) = m \cdot \frac{q_{m \bmod \pi_n}}{r_{m \bmod \pi_n}}$$

The *Conway set* $\mathcal{G}$ is defined as $\{g^i(2) \colon i \in \mathbb{N}\}$, i.e., the smallest set containing 2 and closed under applications of g. It is known that checking if $\mathcal{G}$ is bounded is undecidable [Gogacz and Marcinkowski, 2014, Sec. 3.2–3.3].

Lemma 2.4. *For a given natural numbers n, π_n and q_i, r_i for $i \in [0, \pi_n)$, the problem of deciding finiteness of $\mathcal{G}$ is undecidable.*

Without losing undecidability status of the above problem, we may assume that the sequence $g(2), g^2(2), g^3(2), \ldots$ is non-decreasing. It implies that for a finite Conway set $\mathcal{G}$ the sequence $g^n(2)$ will eventually be constant.

3 Undecidability Proof

This section is entirely devoted to the proof of the main theorem in our paper, namely:

Theorem 1.1 (restated). *All-Instances Oblivious Chase Termination is undecidable for sets of binary single-head TGDs.*

The proof of Theorem 1.1 is done by a reduction from the boundedness of Conway functions. We recall that n, π_n, g and $\mathcal{G} = \{g^n(2) \colon n \in \mathbb{N}\}$ are fixed and defined as in Section 2.2. Note that by applying Lemma 2.3 and Lemma 2.4, one can conclude that in order to show the main result it is sufficient to find a set $\mathcal{T}_{all}$ of TGDs such that:

Lemma 3.1. *The $\mathcal{T}_{all}$–chase terminates on the critical instance $\mathfrak{I}_{\mathsf{cr}}$ iff the Conway set $\mathcal{G}$ is bounded.*

Constructing such set of TGDs, even when there is no restriction on the arity of relations, is no easy feat. It can already be seen in [Gogacz and Marcinkowski, 2014] that working with the critical instance is a tedious fight. As our reduction shares some of the ideas with the mentioned work, it needs to overcome the same (and even a few more) problems. We briefly discuss our main ideas in the context of the existing works of Gogacz and Marcinkowski.

The TGDs used in the undecidability proof in [Gogacz and Marcinkowski, 2014] build a path starting from a critical instance. For every vertex v on such a path the chase computes a “private” Conway set $\mathcal{G}_v$ of v, where the numbers in $\mathcal{G}_v$ are implicitly represented as the distances between v and the other vertices from the path. When the source “gets” into the Conway set $\mathcal{G}_v$ then v is allowed to give birth to the next node on the path. The machinery required for computing $\mathcal{G}_v$ in their work employed multiplication and hence it lead to seemingly inherent ternarity of their construction: to say that one segment is c times longer than the other one they need to name two such segments in a single relation and thus the presence of three vertices in one atom were required.

In our construction, instead of just building a path, we build a whole binary tree. The beauty of the construction is that it allows us to uniquely identify the L^*R-ancestor of each node in a tree. Moreover, it grants us the ability to specify that a certain condition holds for nodes u, v and “the unique ancestor of u” rather to quantify over triples of elements. Continuing the idea from Gogacz et al., each node v in a tree also computes its own Conway set $\mathcal{G}_v$ and gives birth to two new children if v “reaches” the source via $\mathcal{G}_v$. It means that if the $\mathcal{G}$ is unbounded then the new vertices will be created indefinitely.

3.1 Schema and Treelike Instances

We define a schema $\mathbb{S}$ composed of the following relations:

$Src(x)$ a unary relation used to distinguish the source from other nodes (no TGD will use Src in its head).

$L(x, y)$ “left” successor relation.

$R(x, y)$ “right” successor relation.

Relations L and R are used to construct the set of edges in treelike instances defined later.

$E(x, y)$ a binary relation denoting “left or right” successor.

$Re_i(x, y)$ for any $i \in [0, \pi_n - 1]$ is a binary relation denoting that the distance between x and y modulo π_n is equal to i.

$Sp(x, y)$ a binary relation stating that x is on the *side path* of y. This relation along with the next one is the main building block used in encoding multiplication in treelike structures.

$Lvl(x, y)$ a binary relation connecting relevant vertices on equal “depth”.

$Mult_i(x, y)$ for any $i \in [0, \pi_n - 1]$ is a binary relation that works closely with the Sp relation to encode multiplication.

$G(x, y)$ a binary relation denoting that the distance between x and y is in $\mathcal{G}$.

Families of Treelike Instances

Analyzing the intermediate instances produced by the chase is a tedious task. To make such reasoning significantly easier, one can define a family of instances having some “neat” properties, i.e., satisfying the following: when $\mathcal{G}$ is bounded, rather than showing that the chase terminates, we can observe that all intermediate instances are contained in some finite structure from such family; otherwise, intermediate instances will contain bigger and bigger members of this family. Hence, analysing the chase boils down to analysing such family. In this section, we present three such families. Before we follow, let us stress that all paths in instances from $\mathfrak{T}$ go up toward the source (rather than go down as usual in computer science).

Let us move to the definition of the first family of instances. We encourage the reader to take a look at Figure 1.

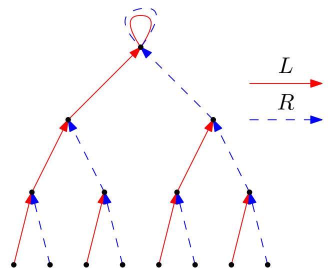

Figure 1: The instance $\mathfrak{T}_3$ restricted to the relations L and R. Note that the root is the source and thus the restriction happens only there.

Definition 3.2 (family $\mathfrak{T}$)**.** *For any $i \in \mathbb{N}$ let $\mathfrak{T}_i$ be a full binary tree of depth i with successor relations L and R such that:*

- *the root is replaced with the source and*
- *if x is a node from $\mathfrak{T}_i$ and x_l (resp. x_r) is its left (resp. right) child, then $\mathfrak{T}_i \models L(x_l, x)$ (resp. $\mathfrak{T}_i \models R(x_r, x)$) holds.*

The family $\mathfrak{T}$ is defined as a set $\{\mathfrak{T}_i \mid i \in \mathbb{N}\}$.

Observe that the instance $\mathfrak{T}_0$ is equivalent to the critical instance $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$. Note that due to the treelike nature of the family $\mathfrak{T}$ we can distinguish *leafs* and the *root* in instances from $\mathfrak{T}$.

Paths. Let $\mathfrak{I}$ be an instance over $\mathbb{S}$ and let x and y be two of its nodes. If x is connected to y by a directed path p composed of relations from some $\mathbb{S}' \subseteq \mathbb{S}$ we say that the path p is an $\mathbb{S}'$*–restricted* path. Also, for a regular language $\mathcal{L}$ over $\mathbb{S}$ a path p is $\mathcal{L}$–restricted if there exists a word from $\mathcal{L}$ that is a naturally corresponding to relations used by p. For a path p we denote its length with $|p|$.

Distances. We discuss the notion of the distance in the instances from $\mathfrak{T}$ family. Henceforth let us fix a natural number i and an instance $\mathfrak{I} \equiv_{\{L,R\}} \mathfrak{T}_i$. Let x, y be nodes from $\mathfrak{I}$. Contrary to what was expected here, we cannot simply define the distance between nodes x and y as the length of the $\{L, R\}$–restricted path between those nodes. When the source is the endpoint of a path it warps the natural notion of distance due to its L and R loops. Hence, we employ a more complicated definition using a set-based approach. With $\mathsf{dist}(x, y)$ we denote the set of all $|p|$ for p being a $\{L, R\}$–restricted path from x to y, i.e. the set of all possible "distances" from x to y. Note that there are only three possible cardinalities of the distance set: $0, 1$ or ∞. If nodes x, y are not connected by a directed $\{L, R\}$–restricted path then $|\mathsf{dist}(x, y)| = 0$, if x and y are connected and y is not the root then $|\mathsf{dist}(x, y)| = 1$ and otherwise $|\mathsf{dist}(x, y)| = \infty$. While this definition is a bit confusing it is quite important one as the construction would not work properly with L and R absent from the source.

Side paths. The notion of a *side path* plays an instrumental role in our construction. Let us fix an instance $\mathfrak{I} \equiv_{\{L,R\}} \mathfrak{T}_i$. For any given node $y \in \mathfrak{I}$ the *side path* of y is the set of nodes x s.t. there is a (L^*R)–restricted path from x to y. Moreover for any x from the side path of y, we say that y is the *side parent* of x. We stress that every node from $\mathfrak{I}$ has a unique side parent.

As a next step, we define the family $\mathfrak{T}^{built}$.

Definition 3.3 (family $\mathfrak{T}^{built}$). *For any natural number i the instance $\mathfrak{T}_i^{built}$ is defined as the minimal instance satisfying $\mathfrak{T}_i \equiv_{\{L,R\}} \mathfrak{T}_i^{built}$ such that all following conditions hold for any two nodes $x, y \in \mathfrak{T}_i^{built}$:*

- *$\mathfrak{T}_i^{built} \models E(x, y)$ holds if and only if $\mathfrak{T}_i^{built} \models L(x, y)$ holds or $\mathfrak{T}_i^{built} \models R(x, y)$ holds.*
- *$\mathfrak{T}_i^{built} \models Re_j(x, y)$ if and only if there exists a $\{L, R\}$–restricted path of length k connecting x and y such that k is congruent to j (modulo π_n).*
- *$\mathfrak{T}_i^{built} \models Sp(x, y)$ if and only if there is a (L^*R)–restricted path from x to y.*
- *$\mathfrak{T}_i^{built} \models Lvl(x, y)$ if and only if there exists a node $z \in \mathfrak{T}_i^{built}$ and a natural number k such that $\mathfrak{T}_i^{built} \models Sp(y, z) \wedge E^k(x, z) \wedge E^k(y, z)$ holds.*
- *$\mathfrak{T}_i^{built} \models Mult_j(x, y)$ if and only if there exists a node z witnessing $Sp(x, z)$ such that all values $v \in \mathsf{dist}(x, z)$ satisfy $\frac{q_j}{r_j} \cdot v \in \mathsf{dist}(x, y)$.*

The family $\mathfrak{T}^{built}$ is defined as a set $\{\mathfrak{T}_i^{built} \mid i \in \mathbb{N}\}$.

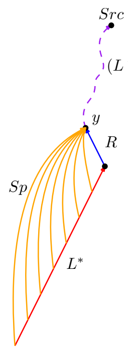

Figure 2: The relation Sp for a single side parent y.

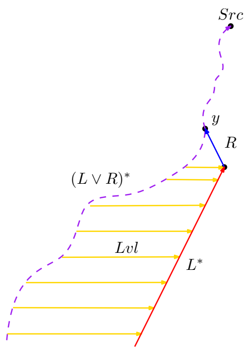

Figure 3: The relation Lvl for a single side path and some $\{L, R\}$–restricted path .

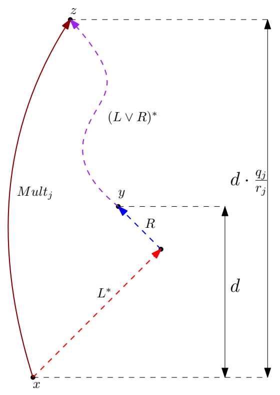

Figure 4: The relation $Mult_j$. Observe that the third vertex y is uniquely defined as a side parent of x.

One may wonder about the purpose of instances from $\mathfrak{T}^{built}$ family. Those instances are simply nodes from the previously defined family $\mathfrak{T}$ equipped with a framework for computing the Conway set $\mathcal{G}$. On the other hand, looking somewhat further ahead, the instance $\mathfrak{T}_i^{built}$ is exactly the result of starting the $\mathcal{T}_{build}$-chase on $\mathfrak{T}_i$, where the set of TGDs $\mathcal{T}_{build}$ is going to be defined soon. We postpone the definition of the TGDs until Section 3.2, as we believe that it is easier to grasp the meaning of relations from $\mathbb{S}$ and hence the proof, by understanding those "declarative" definitions first.

Most of the properties of $\mathfrak{T}^{built}$ family are easy to understand, except the property of relations $Mult_j$. Along those lines lies the essence of our proof as the binary relation Sp neatly stores (with a help from side paths) the ternary relation responsible for multiplication. We strongly encourage the reader to take more time in visualising the family $\mathfrak{T}^{built}$ and taking a look at the Figures 2-4.

Finally, we define the last family of tree-like instances required in our undecidability proof. Similarly to the previous case, one may think that those instances are constructed by the chase on $\mathfrak{T}_i^{built}$. One can also view $\mathfrak{T}_i^{comp}$ as an instance $\mathfrak{T}_i^{built}$ after computation of Conway set $\mathcal{G}$ reaches a fixed point due to the limited depth of the instance.

Definition 3.4 (family $\mathfrak{T}_i^{comp}$). *For any natural number i the instance $\mathfrak{T}_i^{comp}$ is defined as a minimal instance satisfy-*

ing $\mathfrak{T}_i^{comp} \equiv_{\mathbb{S}\setminus\{G\}} \mathfrak{T}_i^{built}$ such that for any two nodes x and y from $\mathfrak{T}_i^{comp}$, the condition $\mathfrak{T}_i^{comp} \models G(x,y)$ holds iff there exists $k \in \mathsf{dist}(x,y)$ which also belongs to the Conway set $\mathcal{G}$. The family $\mathfrak{T}^{comp}$ is defined as the set $\{\mathfrak{T}_i^{comp} \mid i \in \mathbb{N}\}$.

3.2 Tuple Generating Dependencies

Now, once the intended meaning of all relations is known, we define the set of TGDs $\mathcal{T}_{all}$ containing fourteen TGDs. Whilst this number may seem overwhelming, this set can be easily divided into three parts having a coherent and hopefully easy to grasp meaning. We divide the set $\mathcal{T}_{all}$ as follows: TGDs from 1 to 10 are denoted with $\mathcal{T}_{build}$, TGDs from 11 to 12 are denoted with $\mathcal{T}_{comp}$ and the set of the last two TGDs is denoted with $\mathcal{T}_{grow}$. We also note that the TGDs 4, 9, 10 and 12 are actually sets of TGDs (having one TGD per each $i \in [0, \pi_n - 1]$) rather than just single rules. Note that TGDs from $\mathcal{T}_{comp}$ and $\mathcal{T}_{build}$ are datalog, while TGDs from $\mathcal{T}_{grow}$ are existential.

1. $L(x,y) \quad\rightarrow\quad E(x,y)$
2. $R(x,y) \quad\rightarrow\quad E(x,y)$

The first two TGDs are responsible for building the relation E over the relations L and R so that E can act as the disjunction $L \vee R$ in the TGDs appearing later. Also, to simplify the forthcoming TGDs, for the relation L (and analogously for R and E) we use $L^i(x_0, x_i)$ as a syntactic sugar for $L(x_0,x_1) \wedge L(x_1,x_2) \wedge \ldots \wedge L(x_{i-1},x_i)$.

3. $E(x,y) \quad\rightarrow\quad Re_1(x,y)$
4. $Re_i(x,y),\ E(y,z) \quad\rightarrow\quad Re_{i+1 \bmod \pi_n}(x,z)$

TGDs 3–4 construct the relation Re_i such that $Re_i(x,y)$ holds for nodes x and y if the distance from x to y is equal to i modulo π_n.

5. $R(x,y) \quad\rightarrow\quad Sp(x,y)$
6. $L(x,y),\ Sp(y,z) \quad\rightarrow\quad Sp(x,z)$
7. $E(y,x),\ R(y',x) \quad\rightarrow\quad Lvl(y,y')$
8. $Lvl(x,y),\ E(x',x),\ L(y',y) \quad\rightarrow\quad Lvl(x',y')$

The sole purpose of TGDs 5–8 is to build relations Sp and Lvl (see also Figures 2 and 3).

9. $R(s,y),\ L^{r_i-1}(x,s),\ E^{q_i-r_i}(y,z) \rightarrow Mult_i(x,z)$
10. $Mult_i(x,z),\ L^{r_i}(x',x),\ E^{q_i-r_i}(z,z') \rightarrow Mult_i(x',z')$

The TGDs above are responsible for constructing the relation $Mult_i$ which serves as a way (along with the Lvl relation) to simulate multiplication in our construction (see Figure 4). The set $\mathcal{T}_{build}$ of TGDs (i.e., TGDs from 1 to 10) serves as a building mechanism. This mechanism is meant to be able to extend a number of relations over some treelike instances (more precisely the instance $\mathfrak{T}_i$ from Definition 3.2).

11. $E(x,y),\ E(y,z) \quad\rightarrow\quad G(x,z)$
12. $Re_i(x,y),\ G(x,y),\ Lvl(x,x'),\ Sp(x',y),\ Mult_i(x',z) \rightarrow G(x,z)$

The computation of a Conway set is done with the TGDs 11 and 12, defined above. The TGD 12 may look complicated, but it is a crucial element of our proof as the body of that rule is able to extract a "hidden" ternary information. Figures 5–8 might shed some light onto these TGDs.

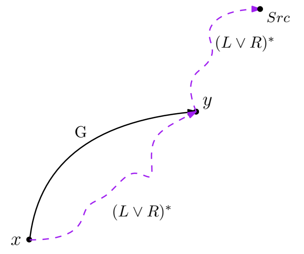

Figure 5: To understand how TGD 12 works, we pick x and y connected with G.

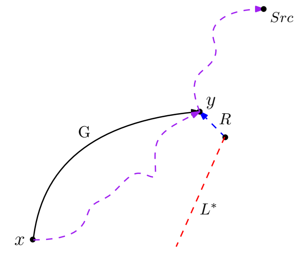

Figure 6: We focus on vertices on the side path of y (located on RL^* path downwards from y).

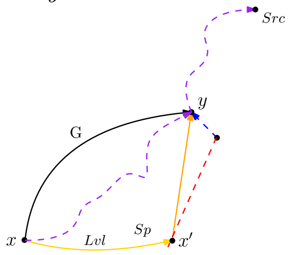

Figure 7: There exists a unique vertex x' satisfying both $Lvl(x,x')$ and $Sp(x',y)$.

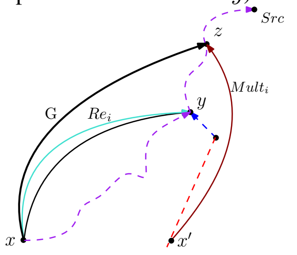

Figure 8: The length $d = g^k(2)$ of the path between x and y is congruent to i modulo π_n and thus Re_i connects x and y. Also from $Mult_i(x',z)$ we know that the distance between x' and z is equal to $d \cdot \frac{q_i}{r_i}$ which in turn is equal to $g^{k+1}(2)$ thus we connect x and z with G.

13. $G(x,h),\ Src(h) \quad\rightarrow\quad \exists z\ L(z,x)$
14. $G(x,h),\ Src(h) \quad\rightarrow\quad \exists z\ R(z,x)$

The last two TGDs (denoted with $\mathcal{T}_{grow}$) are the only existential rules in $\mathcal{T}_{all}$. Those TGDs are used to enlarge instances and to make them look similar to the full binary trees. Notice connection between those two rules and a family of treelike instances $\mathfrak{T}$.

The following observation about TGDs 13–14 will prove useful later:

Observation 3.5. *Every node of $\mathsf{Ch}(\mathcal{T}_{all}, \mathfrak{I}_{\mathsf{cr}})$, except the source, has at most one child connected by L and at most one connected by R.*

3.3 From Unboundedness to Nontermination

In this section we show that if the Conway set $\mathcal{G}$ is infinite then the $\mathcal{T}_{all}$-chase does not terminate on the critical instance $\mathfrak{I}_{\mathsf{cr}}^{\mathbb{S}}$. Let us, for the rest of this section, define $\mathbb{I}$ as the set of all intermediate instances of the $\mathcal{T}_{all}$-chase run on $\mathfrak{I}_{\mathsf{cr}}^{\mathbb{S}}$.

Lemma 3.6. *If $\mathcal{G}$ is not finite then the $\mathcal{T}_{all}$-chase does not terminate on the critical instance $\mathfrak{I}_{\mathsf{cr}}^{\mathbb{S}}$.*

Note that a single step of the chase can increase the structure only by a single atom. Hence it is sufficient to show that the sequence of intermediate instances from the $\mathcal{T}_{all}$-chase run on $\mathfrak{I}_{\mathsf{cr}}^{\mathbb{S}}$ (= $\mathfrak{T}_0$) contains arbitrarily large instances:

Lemma 3.7. *If $\mathcal{G}$ is infinite then for every natural number i there exists an intermediate instance $\mathfrak{I} \in \mathbb{I}$ such that $\mathfrak{T}_i \subseteq \mathfrak{I}$.*

To show Lemma 3.7 we employ an inductive reasoning in which the following lemma plays the central role:

Lemma 3.8. *If $\mathcal{G}$ is infinite then for every natural number i:*

- *if there exists an instance $\mathfrak{I} \in \mathbb{I}$ containing $\mathfrak{T}_i$ then there exists an instance $\mathfrak{I}' \in \mathbb{I}$ containing $\mathfrak{T}_i^{built}$*
- *if there exists an instance $\mathfrak{I} \in \mathbb{I}$ containing $\mathfrak{T}_i^{built}$ then there exists an instance $\mathfrak{I}' \in \mathbb{I}$ containing $\mathfrak{T}_i^{comp}$*
- *if there exists an instance $\mathfrak{I} \in \mathbb{I}$ containing $\mathfrak{T}_i^{comp}$ then there exists an instance $\mathfrak{I}' \in \mathbb{I}$ containing $\mathfrak{T}_{i+1}$*

Proof. Therein we present only the second claimed property. We believe that it is both the most essential and the most difficult bit of our construction. Let us define $\mathfrak{I}_{build}$ as the restriction of the instance $\mathfrak{I}$ to the image of the injective homomorphism witnessing $\mathfrak{T}_i^{built} \subseteq \mathfrak{I}$. Observe that from the definition of the family $\mathfrak{T}_i^{comp}$, it is sufficient to show that for every natural number k and any vertices $x, y \in \mathfrak{I}_{build}$ the following implication holds: if $g^k(2) \in \mathsf{dist}(x, y)$ then $\mathsf{Ch}(\mathcal{T}_{all}, \mathfrak{T}_0) \models G(x, y)$. We prove it by induction over k. In the proof the only TGDs in use are TGDs 11-12 hence it might be helpful to take a look at Figures 5-8. The base of the induction is trivially satisfied as TGD 11 ensures that any two vertices connected by a path of length 2 in $\mathfrak{I}_{build}$ are connected by the G edge in $\mathsf{Ch}(\mathcal{T}_{all}, \mathfrak{T}_0)$.

Let us pick a pair of nodes x and z from $\mathfrak{I}_{build}$ that are connected by a $\{L, R\}$–restricted path of length $g^{k+1}(2)$. We will show that there exists a trigger in the instance $\mathfrak{I}_{build}$ for TGD 12 such that its head is mapped to vertices x and z. Let y be a node on $\{L, R\}$–restricted path from x to z such that $g^k(2) \in \mathsf{dist}(x, y)$. Thus from the induction hypothesis we infer that $\mathfrak{I}_{build} \models G(x, y)$. If y is the source then $y = z$ and from the fact that $\mathfrak{I}_{build} \models G(x, y)$ we conclude that $\mathfrak{I}_{build} \models G(x, z)$. Let us consider the case when u is not the source. Let $x' \in \mathfrak{I}_{build}$ be the node such that $\mathfrak{I}_{build} \models Sp(x', y)$ holds and x' satisfies $g^k(2) \in \mathsf{dist}(x', y)$. Notice that by the definition of $\mathfrak{T}_i^{built}$ we know that Lvl connects x and x' in $\mathfrak{I}_{build}$. Observe that $Re_j(x, y)$ holds in $\mathfrak{I}_{build}$ for $j \equiv g^k(2) \mod \pi_n$. Finally, we need to show that $Mult_j(x', z)$ holds in $\mathfrak{I}_{build}$. By employing the definition of $\mathfrak{T}_i^{built}$, it is sufficient to show that every $v \in \mathsf{dist}(x', y)$ satisfies $\frac{q_j}{r_j} \cdot v \in \mathsf{dist}(x', z)$. We know that y is not the source so set $\mathsf{dist}(x', y)$ contains only $g^k(2)$. See that the following holds:

$$\frac{q_j}{r_j} \cdot g^k(2) = \frac{q_{g^k(2) \bmod \pi_n}}{r_{g^k(2) \bmod \pi_n}} \cdot g^k(2) = g(g^k(2)) = g^{k+1}(2)$$

We know $g^{k+1}(2) \in \mathsf{dist}(x, z) = \mathsf{dist}(x', z)$, which implies the satisfaction of $\mathfrak{I}_{build} \models Mult_i(x', z)$. Hence we conclude that there is the required trigger for the TGD 12 in $\mathfrak{I}_{build}$, which when applied, connects x and z by the relation G. From the definition of the chase sequence we know that each trigger is eventually applied, which guarantees the existence of the demanded instance. □

3.4 From Boundedness to Termination

It remains to prove that if the Conway set $\mathcal{G}$ is bounded then the $\mathcal{T}_{all}$-chase terminates on the critical instance $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$. Here we really reap the benefits of our declarative definitions from Section 3.1 and avoid the tedious work with the ever-expanding chase. The following observations come from the definition of the family $\mathfrak{T}^{comp}$.

Observation 3.9. *For every i the instance $\mathsf{Ch}(\mathcal{T}_{build} \cup \mathcal{T}_{comp}, \mathfrak{T}_i^{comp})$ is isomorphic to $\mathfrak{T}_i^{comp}$.*

Observation 3.10. *If $\mathcal{G}$ is bounded then in $\mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ no leaf is connected by a G-edge with the source.*

We employ these two observations to prove the following:

Lemma 3.11. *If $\mathcal{G}$ is finite then the $\mathcal{T}_{all}$–chase terminates on the critical instance.*

Proof. We will show that every intermediate instance of $\mathcal{T}_{all}$-chase on the critical instance $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$ is contained within a finite instance $\mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ (existing due to fact that $\mathcal{G}$ is bounded) and thus the chase terminates. First observe that $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$ is trivially contained in $\mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$. Let us take a trigger (φ, h) and any pair of intermediate instances $\mathfrak{I}$ and $\mathfrak{I}'$ from $\mathcal{T}_{all}$-chase run on $\mathfrak{I}^{\mathbb{S}}_{\mathsf{cr}}$ such that $\mathfrak{I}(\varphi, h)\mathfrak{I}'$ holds. Suppose that $\mathfrak{I} \subseteq \mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ holds and we will prove that $\mathfrak{I}' \subseteq \mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ holds. Let λ be an injective homomorphism witnessing the fact that $\mathfrak{I} \subseteq \mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ holds. If a TGD φ belongs to $\mathcal{T}_{build} \cup \mathcal{T}_{comp}$ then trivially from Observation 3.9 we infer $\mathfrak{I}' \subseteq \mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$. Otherwise φ is contained in $\mathcal{T}_{grow}$. W.l.o.g suppose that φ is the TGD 13. When the TGD φ was fired, it created the "left" child of some node. From Observation 3.5 we can immediately conclude that φ was applied to a node $v \in \mathfrak{I}$ not having a left child, such that $\mathfrak{I} \models G(v, s)$ holds (where s is the source). By Observation 3.10 we infer that v corresponds through λ to a non-leaf node in $\mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ but then it is easy to extend λ to also witness $\mathfrak{I}' \subseteq \mathfrak{T}^{comp}_{\max(\mathcal{G})+1}$ as follows: let u be a freshly created node and $\lambda(u)$ be equal to the left child of $\lambda(v)$. □

The above lemma, together with Lemma 3.6 from the previous section, allows us to conclude the correctness of Lemma 3.1 and thus the correctness of the main theorem.

4 Conclusions

In this paper, we have shown that the All-Instance Oblivious Chase Termination Problem for the class of single-head binary TGDs is undecidable. By arguing along the lines of (Subsection 3.5 in [Gogacz and Marcinkowski, 2014]) we may also conclude undecidability of All-Instances Termination for the Standard chase and for the Semi-Oblivious chase. It is fair to say that the presented proof is quite technical. We encoded the boundedness problem of Conway functions in the growing structure of the chase executed on the critical instance. The main obstacle was to encode the multiplication function, which is inherently ternary and was used e.g. in [Gogacz and Marcinkowski, 2014]. To avoid ternarity we force the structure generated by the chase to be a binary tree and employ the idea of the side paths to uniquely identify certain elements.

A natural direction for further investigation could be as follows: is there any non-trivial and well-known class of TGDs of arbitrary arity, which also causes undecidability of All-Instances chase Termination? Natural candidates are classes of weakly sticky-join [Calì *et al.*, 2010], glut-frontier-guarded [Krötzsch and Rudolph, 2011] or model faithful-acyclic [Grau *et al.*, 2013] TGDs, since they are the most expressive among other TGD classes.

References

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 2011.

[Beeri and Vardi, 1984] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 1984.

[Benedikt *et al.*, 2017] Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. Benchmarking the chase. In *PODS*, 2017.

[Calautti and Pieris, 2019] Marco Calautti and Andreas Pieris. Oblivious chase termination: The sticky case. In *ICDT 2019*, 2019.

[Calautti *et al.*, 2015] Marco Calautti, Georg Gottlob, and Andreas Pieris. Chase termination for guarded existential rules. In *PODS*, 2015.

[Calì *et al.*, 2010] Andrea Calì, Georg Gottlob, and Andreas Pieris. Query answering under non-guarded rules in datalog+/-. In *RR*, 2010.

[Calì *et al.*, 2013] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 2013.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 2005.

[Fagin, 2018] Ronald Fagin. Tuple-generating dependencies. In *Encyclopedia of Database Systems, Second Edition*. Springer, 2018.

[Gogacz and Marcinkowski, 2014] Tomasz Gogacz and Jerzy Marcinkowski. All-instances termination of chase is undecidable. In *ICALP*, 2014.

[Gogacz *et al.*, 2020] Tomasz Gogacz, Jerzy Marcinkowski, and Andreas Pieris. All-instances restricted chase termination: The guarded case. To appear in *PODS*, 2020.

[Grahne and Onet, 2018] Gösta Grahne and Adrian Onet. Anatomy of the chase. *Fundam. Inform.*, 2018.

[Grau *et al.*, 2013] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.*, 2013.

[Halevy, 2001] Alon Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 2001.

[Hitzler *et al.*, 2009] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer. Technical report, World Wide Web Consortium, 2009.

[Krötzsch and Rudolph, 2011] Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI*, 2011.

[Leclère *et al.*, 2019] Michel Leclère, Marie-Laure Mugnier, Michaël Thomazo, and Federico Ulliana. A single approach to decide chase termination on linear existential rules. In *ICDT*, 2019.

[Maier *et al.*, 1979] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 1979.

[Marnette, 2009] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, 2009.

[Olteanu *et al.*, 2009] Dan Olteanu, Jiewen Huang, and Christoph Koch. SPROUT: lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, 2009.

[Urbani *et al.*, 2018] Jacopo Urbani, Markus Krötzsch, Ceriel J. H. Jacobs, Irina Dragoste, and David Carral. Efficient model construction for horn logic with vlog - system description. In *IJCAR*, 2018.