Loop Restricted Existential Rules and First-order Rewritability for Query Answering*

Vernon Asuncion, Yan Zhang
School of Data, Computer and Mathematical Sciences
Western Sydney University, Australia
Email: {v.asuncion, yan.zhang}@westernsydney.edu.au

Heng Zhang
College of Intelligence and Computing
Tianjin University, China
Email: heng.zhang@tju.edu.cn

Yun Bai
School of Data, Computer and Mathematical Sciences
Western Sydney University, Australia
Email: y.bai@westernsydney.edu.au

Abstract

In ontology-based data access (OBDA), the classical database is enhanced with an ontology in the form of logical assertions generating new intensional knowledge. A powerful form of such logical assertions is the tuple-generating dependencies (TGDs), also called existential rules, where Horn rules are extended by allowing existential quantifiers to appear in the rule heads. In this paper we introduce a new

*An extended abstract of this paper was published in KR-2018 [1].

language called *loop restricted* (LR) TGDs (existential rules), which are TGDs with certain restrictions on the loops embedded in the underlying rule set. We study the complexity of this new language. We show that the conjunctive query answering (CQA) under the LR TGDs is decidable. In particular, we prove that this language satisfies the so-called bounded derivation-depth property (BDDP), which implies that the CQA is first-order rewritable, and its data complexity is in AC^0. We also prove that the combined complexity of the CQA is 2-EXPTIME complete, while the language membership is PSPACE complete. Then we extend the LR TGDs language to the generalised loop restricted (GLR) TGDs language, and prove that this class of TGDs still remains to be first-order rewritable and properly contains most of other first-order rewritable TGDs classes discovered in the literature so far.

Keywords: knowledge representation and reasoning, logic programming, ontological reasoning, query answering, complexity

1 Introduction

In ontology-based data access (OBDA), the classical database is enhanced with an ontology in the form of logical assertions generating new intensional knowledge, e.g., [2, 7, 17, 22, 26]. A powerful form of such logical assertions is the *tuple-generating dependencies (TGDs)*, also called *existential rules*. Generally speaking, TGDs are Horn rules extended by allowing the occurrence of existential quantification in the rule head. With this extension, it is able to reason about the existence of new or missing objects that are not represented in the underlying database [4, 28].

Under the language of TGDs, queries are answered against an ontology represented by a set of TGDs and an input database. In particular, given a database instance D, a finite set Σ of TGDs, and a Boolean query q, we want to decide whether $D \cup \Sigma \models q$. However, this problem is undecidable generally, due to the potential cyclic applications of TGDs in Σ [5, 16].

In recent years, considerable research has been carried out to identify various expressive decidable classes of TGDs. So far several primary such classes have been discovered: *weakly-acyclic* class [18]; *guarded* class [4, 9, 10]; *frontier guarded* class [4]; *sticky sets* class [11]; and *Shy programs* class [24]. By extending and combining these aforementioned classes, more decidable classes can be derived,

Among all these decidable classes, some are of special interests for OBDA, i.e., the classes of first-order rewritable TGDs, where conjunctive query an-

swering can be reduced to the evaluation of a first-order query over the database. As such, traditional database query techniques may be used for developing efficient query answering systems in OBDA, as demonstrated in Description Logics [12, 21]. So far, several useful first-order rewritable classes of TGDs have been discovered: acyclic TGDs, aGRD TGDs, domain-restricted TGDs [4], linear and multi-linear TGDs, sticky and sticky-join TGDs, while multi-linear and sticky-join TGDs generalise linear TGDs and sticky TGDs, respectively [10, 11].

Civili and Rosati [15] further identified another first-order rewritable class called *weakly recursive* TGDs, and showed that by restricting to simple TGDs[1], weakly recursive class contains all other first-order rewritable classes. However, since the weakly recursive class is only defined for simple TGDs, the acyclic, multi-linear and sticky-join classes remain to be incomparable first-order rewritable classes for general TGDs [15]. On the other hand, their extended work in [14] uplifted the notion of weakly recursive TGDs from that fragment of the so-called "simple" TGDs to arbitrary TGDs, resulting in a class called *labeled oblivious acyclic* (LOA), which is a first-order rewritable class strictly containing the classes of aGRD, multi-linear, sticky-join and domain-restricted TGDs.

Nevertheless, there are still real life scenarios that are simple and intuitive but not syntactically recognisable by any of the existing first-order rewritable TGDs classes, as illustrated by the following example.

Example 1. Consider the set of TGDs $\Sigma_{\mathsf{inCharge}}$ comprising of the following two simple rules:

$$\begin{aligned} \mathsf{r}_1 :\ & \mathsf{activeRole}(X, W) \wedge \mathsf{inCharge}(X, Y) \wedge \mathsf{inCharge}(Y, Z) \rightarrow \\ & \quad \mathsf{indInCharge}(X, Z), \\ \mathsf{r}_2 :\ & \mathsf{indInCharge}(X, Z) \rightarrow \exists U\, \mathsf{activeRole}(X, U). \end{aligned}$$

Here, $\mathsf{activeRole}(X, W)$ encodes that "X" has an active role "W", and $\mathsf{inCharge}(X, Y)$ encodes that "X" is in charge of "Y". The rule r_1 encodes that: *if "X" is in charge of "Y" and "Y" is in charge of "Z", then "X" is* indirectly *in charge of "Z"*. On the other hand, the rule r_2 encodes the *integrity constraint* stating that: *"X" indirectly in charge of "Z" implies that X must have some active role*.

Through a careful examination, it is not difficult to see that $\Sigma_{\mathsf{inCharge}}$ is not recognizable under the syntactic conditions of all currently known first-order rewritable TGDs classes.

[1] A TGD is *simple* if there are no constant and repeated variable occurrences in each atom.

On the other hand, by unfolding the derivations of atoms $\mathsf{activeRole}(X, W)$ and $\mathsf{indInCharge}(X, Z)$ from $\Sigma_{\mathsf{inCharge}}$, it turns out that their derivations are always bounded by a fixed length independent from any input database. That is, the underlying $\Sigma_{\mathsf{inCharge}}$ satisfies the so-called bounded derivation-depth property (BDDP, see its formal definition in next section), from which we know that the query answering under $\Sigma_{\mathsf{inCharge}}$ is not only decidable, but also first-order rewritable [8, 10].

Based on these ideas, the main contributions of this paper are summarised as follows:

1. We define notations of *derivation paths* and *derivation trees* for query answering over TGDs (existential rules), and provide a precise characterisation for the traditional TGDs chase procedure through the corresponding derivation tree.

2. Based on the concept of derivation paths, we introduce a new class called *loop restricted* (LR) TGDs, which are TGDs with certain restrictions on the loops embedded in the underlying rule set.

3. Under our derivation tree framework, we show that the conjunctive query answering (CQA) under LR TGDs satisfies a property called bounded derivation tree depth property (BDTDP). We further prove that BDTDP implies the well-known bounded derivation-depth property (BDDP). This result implies that conjunctive query answering under LR TGDs is not only decidable but also first-order rewritable.

4. We further extend LR TGDs to *generalised loop restricted* (GLR) TGDs, and prove that the class of GLR TGDs is also first-order rewritable and contains most of other first-order rewritable TGD classes discovered in literature so far.

The rest of this paper is organised as follows. Section 2 presents necessary preliminaries we will need through out this paper. Section 3 defines the concepts of derivation paths and derivation trees, from which a characterisation for the chase procedure is provided. By defining the notion of loop patterns, section 4 then introduces the loop restricted (LR) TGDs, proves that this new class of TGDs satisfies the BDTDP property, and provides the main complexity results. Section 5 further generalises the class of LP TGDs to a class called generalised loop restricted (GLR) TGDs and proves the important properties of this class of TGDs. Section 6, on the other hand, investigates the relationship between the class of GLR TGDs and other existing first-order rewritable classes of TGDs. Finally, section 7 concludes the paper with some remarks.

2 Preliminaries

Databases and queries. We define the following pairwise disjoint (countably infinite) sets of symbols: a set Γ of *constants*, which constitute the domain of databases, a set Γ_N of *labeled nulls* that will be used as "fresh" Skolem terms as placeholders for unknown values, and a set Γ_V of regular *variables*. For convenience, we usually use $a, b, c, \cdots$ to denote constants, $\mathsf{n}, \mathsf{n}', \mathsf{n}'' \cdots$ to denote nulls, and $X, Y, Z, \cdots$ to denote variables[2]. Note that different nulls may also represent the same value. We use $\mathbf{X}$ to denote a sequence of variables $X_1, \cdots, X_n$, where $n \geq 0$. Sometimes, we also represent such $\mathbf{X}$ as a n-ary tuple of variables $(X_1, \cdots, X_n)$. A similar notion also applies to nulls.

A *relational schema* $\mathcal{R}$ is a finite set of *relation symbols* (or *predicates*). A *term* is a constant, null or variable. An *atom* has the form $p(t_1, \cdots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. Note that, here we slightly extend the traditional notion of an atom to allow it containing nulls for the sake of presentation simplicity . We denote by $|p|$ and $\mathsf{dom}(p)$ as p's arity and the set of all its terms respectively. The latter notion is naturally extended to sets of atoms and conjunctions of atoms. A conjunction of atoms is often identified with the set of all its atoms.

A *database* D for a relational schema $\mathcal{R}$ is a finite set of atoms with predicates from $\mathcal{R}$ and constants from Γ. That is, $\mathsf{dom}(D) \subseteq \Gamma$. We also use $\mathsf{pred}(D)$ to denote the set of all predicates occurring in D. An *instance* I for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and terms from $\Gamma \cup \Gamma_N$. Clearly, each database D for $\mathcal{R}$ may be viewed as a special form of instance, and further, it can be extended to an instance I such that $D \subseteq I$ and $\mathsf{pred}(I) = \mathcal{R}$.

A *homomorphism* from a set of atoms $\mathbf{A}$ to a set of atoms $\mathbf{A}'$ is a mapping $h : \Gamma \cup \Gamma_N \cup \Gamma_V \to \Gamma \cup \Gamma_N \cup \Gamma_V$, such that (i) if $t \in \Gamma$, then $h(t) = t$; (ii) if $t \in \Gamma_N$, then $h(t) \in \Gamma \cup \Gamma_N$; and (iii) if $p(t_1, \cdots, t_n) \in \mathbf{A}$, then $p(h(t_1), \cdots, h(t_n)) \in \mathbf{A}'$. Let $\mathbf{T}$ be the set of all terms occurring in $\mathbf{A}$. The *restriction* h' of h to $\mathbf{S} \subseteq \mathbf{T}$, denoted as $h' = h|_{\mathbf{S}}$, is simply the subset of h : $h' = \{t \to h(t) \mid t \in \mathbf{S}\}$. Here we also call h is an *extension* of h' to $\mathbf{T}$.

A *conjunctive query (CQ)* q of arity n over a schema $\mathcal{R}$ has the form $p(\mathbf{X}) \leftarrow \exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y})$, where $\varphi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms with the variables $\mathbf{X}$ and $\mathbf{Y}$ from Γ_V and constants from Γ, but without nulls, and p is an n-ary predicate not occurring in $\mathcal{R}$. We allow $\varphi(\mathbf{X}, \mathbf{Y})$ to contain equalities but no inequalities. When $\varphi(\mathbf{X}, \mathbf{Y})$ is just a single atom, then we say that the CQ q is *atomic*. A *Boolean Conjunctive Query (BCQ)* over $\mathcal{R}$ is a CQ of zero

[2]Possibly these constants, nulls and variable are subscripted with indexes.

arity. In this case, we can simply write a BCQ q as $\exists \mathbf{Y} \varphi(\mathbf{Y})$. For simplicity, we usually write a BCQ as the set of atoms with corresponding constants and variables as the arguments, and omitting the quantifiers.

The CQ answering problem, or called CQA problem, is defined to be the *answer* to a CQ q with n arity over an instance I, denoted as $q(I)$, that is the set of all n-tuples $\mathbf{t} \in \Gamma^n$ for which there exists a homomorphism $h : \mathbf{X} \cup \mathbf{Y} \to \Gamma \cup \Gamma_V$ such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$ and $h(\mathbf{X}) = \mathbf{t}$. The answer to a BCQ is *positive* over I, denoted as $I \models q$, if $\langle\rangle \in q(I)$.

TGDs and conjunctive query answering (CQA). A tuple-generating dependency (TGD) σ, also called *existential rule*, over a schema $\mathcal{R}$ is a first-order formula of the form

$$\sigma : \forall \mathbf{X} \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z} \psi(\mathbf{X}, \mathbf{Z}), \tag{1}$$

where $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} \subset \Gamma \cup \Gamma_V$, φ and ψ are conjunctions of atoms over $\mathcal{R}$. When there is no confusion, we usually omit the universal quantifiers from (1). In this case, we also use $\mathsf{head}(\sigma)$ and $\mathsf{body}(\sigma)$ to denote formulas $\exists \mathbf{Z} \psi(\mathbf{X}, \mathbf{Z})$ and $\varphi(\mathbf{X}, \mathbf{Y})$ respectively.

Let I be an instance over $\mathcal{R}$. We say that σ *is satisfied* in I, denoted as $I \models \sigma$, if whenever there is a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exits an extension h' of $h|_{\mathbf{X}}$ such that $h'(\psi(\mathbf{X}, \mathbf{Z})) \subseteq I$.

Given a database D, a (finite) set Σ of TGDs and a CQ q of arity n over schema $\mathcal{R}$. The *models* of D with respect to Σ, denoted as $\mathsf{mod}(D, \Sigma)$, is the set of all instances I such that $I \supseteq D$ and $I \models \Sigma$.

Then the CQ answering problem, or called CQA problem, denoted as $\langle \mathcal{R}, D, \Sigma, q \rangle$, is described as follows: the *answer* to q with respect to D and Σ, denoted as $\mathsf{ans}(q, D, \Sigma)$, is the set of all tuples: $\{\mathbf{t} \mid \mathbf{t} \in q(I)$, for each $I \in \mathsf{mod}(D, \Sigma)\}$. When q is a BCQ, the answer to q, denoted as BCQA, is called *positive* if $\langle\rangle \in \mathsf{ans}(q, D, \Sigma)$. It is well known that the CQA problem and the problem of CQ containment under TGDs are LOGSPACE-equivalent, and hence, in the rest of this paper, we will only focus on the BCQA problem, because all complexity results can be carried over to other problems [11].

The chase. Consider an instance I and a TGD σ of the form (1). We say that σ is *applicable* to I if:

1. there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$;
2. there does not exists an extension h' of $h\restriction_{\mathbf{XY}}$ such that:
   (a) $h'(\varphi(\mathbf{X}, \mathbf{Y})) = h(\varphi(\mathbf{X}, \mathbf{Y}))$;

(b) $h'(\psi(\mathbf{X}, \mathbf{Z})) \subseteq I$.

Intuitively, 2(b) above is to restrict the unlimited application of "identical" TGD rules in the sense that in the application of σ to I via a homomorphism h, there already exists a homomorphism h' such that $h \restriction_{\mathbf{XY}} = h' \restriction_{\mathbf{XY}}$, i.e., applying σ to I again will be redundant.

The *result* of applying σ to I is an instance $I' = I \cup h'(\psi(\mathbf{X}, \mathbf{Z}))$, where h' is an extension of $h \restriction_{\mathbf{XY}}$ such that for each $Z \in \mathbf{Z}$, $h'(Z)$ is a "fresh" labeled null of Γ_N not occurring in I. Then the oblivious TGD chase algorithm for a database D and a set Σ of TGDs consist of an exhaustive application of chase steps, which leads to a collection of all instances I' generated from such chase steps, denoted as $\mathsf{chase}(D, \Sigma)$. Note that each instance of $\mathsf{chase}(D, \Sigma)$ is a model of $D \cup \Sigma$.

The above procedure gives rise to the so-called chase sequence. A *chase sequence* ζ:

$$I_0 \xrightarrow{\sigma_0, h_0} I_1, \ldots, I_k \xrightarrow{\sigma_k, h_k} I_{k+1} \tag{2}$$

denotes the sequence of applications of the TGD chase rules such that:

1. $I_0 = D$;
2. for each $i \in \{1, \ldots, k\}$, $I_i \xrightarrow{\sigma_i, h_i} I_{i+1}$ denotes the instance $I_{i+1} = I_i \cup \{h'_i(\mathsf{head}(\sigma_i))\}$ such that, assuming $\sigma_i = \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \psi(\mathbf{X}, \mathbf{Z})$, then h'_i is the extension of the homomorphism h_i such that $h_i(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I_i$.

Then for $k \geq 1$, we denote by $\mathsf{chase}_\zeta^{[k]}(D, \Sigma)$ as the resulting instance I_k that is the result of the chase sequence ζ: $I_0 \xrightarrow{\sigma_0, h_0} I_1, \ldots, I_{k-1} \xrightarrow{\sigma_{k-1}, h_{k-1}} I_k$. Moreover, we say that a chase sequence ζ is *fair* if whenever a TGD rule of the form (1) is applicable to an instance I_i through a homomorphism h, then there exists some $k > i$ and extension h' of $h \restriction_{\mathbf{XY}}$ such that $h'(\psi(\mathbf{X}, \mathbf{Z})) \subseteq I_k$.

Finally, we denote by $\mathsf{chase}_\zeta(D, \Sigma)$ as the instance such that:

1. $\mathsf{chase}_\zeta(D, \Sigma) = \bigcup_{i=1}^{i=\infty} \mathsf{chase}_\zeta^{[i]}(D, \Sigma)$;
2. ζ is a fair chase sequence.

When it is clear from the context, we simply denote by $\mathsf{chase}(D, \Sigma)$ for $\mathsf{chase}_\zeta(D, \Sigma)$ (i.e., omitting the "ζ" subscript) to assume that $\mathsf{chase}(D, \Sigma)$ is obtained via some chase sequence ζ.

Given an atom $p(\mathbf{t})$ such that $\mathbf{t} \in (\Gamma \cup \Gamma_N)^{|\mathbf{t}|}$, we say that $\mathsf{chase}(D, \Sigma)$ *entails* $p(\mathbf{t})$, denoted as $\mathsf{chase}(D, \Sigma) \models p(\mathbf{t})$, iff there exists some atom of the same relational symbol $p(\mathbf{t}') \in \mathsf{chase}(D, \Sigma)$ and a homomorphism $h : \mathbf{t} \longrightarrow \mathbf{t}'$ such that $h(p(\mathbf{t})) = p(\mathbf{t}')$.

The notion *level* in a chase under a chase sequence ζ is defined inductively as follows [11]: (1) for an atom $\alpha \in D$, we set $\text{LEVEL}(\alpha) = 0$; (2) then inductively, for an atom $\alpha \in \mathsf{chase}_\zeta(D, \Sigma)$ obtained via some chase step $I_k \xrightarrow{\sigma, h} I_{k+1}$ of the chase sequence ζ, we set $\text{LEVEL}(\alpha) = \text{MAX}\big(\{\text{LEVEL}(\beta) \mid \beta \in h(body(\sigma))\}\big) + 1$. Then finally, for some given $k \in \mathbb{N}$, we set $\mathsf{chase}^k_\zeta(D, \Sigma) = \big\{\alpha \mid \alpha \in \mathsf{chase}(D, \Sigma) \text{ and } \text{LEVEL}(\alpha) \leq k\big\}$. Intuitively, $\mathsf{chase}^k_\zeta(D, \Sigma)$ (or simply $\mathsf{chase}^k(D, \Sigma)$ when clear from the context) is the instance containing atoms that can be derived in a less than or equal to k chase steps.

More technical discussions about the chase can be found from [6, 11].

Theorem 1. *[11] Given a BCQ q over $\mathcal{R}$, a database D for $\mathcal{R}$ and a set Σ of TGDs over $\mathcal{R}$, $D \cup \Sigma \models q$ iff $\mathsf{chase}(D, \Sigma) \models q$.*

Definition 1 (BDDP). *A class $\mathcal{C}$ of TGDs satisfies the* bounded derivation-depth property *(BDDP) if for each BCQ q over a schema $\mathcal{R}$, for every input database D for $\mathcal{R}$ and for every set $\Sigma \in \mathcal{C}$ over $\mathcal{R}$, $D \cup \Sigma \models q$ implies that there exists some $k \geq 0$ which only depends on q and Σ such that $\mathsf{chase}^k(D, \Sigma) \models q$.*

It has been shown that the BDDP implies the first-order rewritability [10, 11]. Formally, the BCQA problem is *first-order rewritable* for a class $\mathcal{C}$ of sets of TGDs if for each $\Sigma \in \mathcal{C}$, and each BCQ q, there exists a first-order query q_Σ such that $D \cup \Sigma \models q$ iff $D \models q_\Sigma$, for every input database D. In this case, we also simply say that the class $\mathcal{C}$ of TGDs is *first-order rewritable*.

3 Derivation Paths and Derivation Trees

For a given set Σ of TGDs, by taking different input databases D, the chase procedure $\mathsf{chase}(D, \Sigma)$ generates different results. However, it is not difficult to observe that atoms of $\mathsf{chase}(D, \Sigma)$ are actually generated by following certain derivation pattens embedded in the rules of Σ, which, in some sense, are independent from the input database D. In this section, we will provide a characterization for this derivation property underlying every given Σ.

Firstly, to simplify our investigations, from now on, we will assume that for any given set Σ of TGDs, each TGD σ in Σ is of a specific form: σ has

only one atom in the head where each existentially quantified variable occurs only once. That is, Σ consists of the following rule:

$$\sigma : \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} p(\mathbf{X}, \mathbf{Z}). \tag{3}$$

Theorem 2. *Let q be a BCQ over $\mathcal{R}$ and Σ a set of TGDs over $\mathcal{R}$. Then the following results hold:*

1. *There exists a* LOGSPACE *construction of an atomic BCQ q' and a set of TGDs Σ' of schema $\mathcal{R}' \supseteq \mathcal{R}$, where $|\mathsf{head}(\sigma')| = 1$ for each $\sigma' \in \Sigma'$, such that for all database D, $D \cup \Sigma \models q$ iff $D \cup \Sigma' \models q'$ [11].*

2. *If Σ' satisfies BDDP then Σ also satisfies BDDP.*

Under Theorem 2, it is clear that considering such special TGDs of the form (3) as well as the atomic BCQ $\exists \mathbf{Z} p(\mathbf{Z})$ will be sufficient, in the sense that all results related to these forms of TGDs and atomic BCQ can be carried over to the general case. So in the rest of this paper, we will only focus on these forms of TGDs and atomic BCQ in our study.

3.1 Comparability and derivation paths

Let $\mathtt{t} = (t_1, \cdots, t_n)$ and $\mathtt{t}' = (t'_1, \cdots, t'_n)$ be two tuples of terms. We say that $\mathtt{t}$ and $\mathtt{t}'$ are *type comparable* if $\mathtt{t}$ and $\mathtt{t}'$ satisfy the following conditions: for each i ($1 \leq i \leq n$), (1) constant $c \in \Gamma$, $t_i = c$ iff $t'_i = c$; (2) $t_i \in \Gamma_V$ iff $t'_i \in \Gamma_V$; and (3) $t_i \in \Gamma_N$ iff $t'_i \in \Gamma_N$. Intuitively, two tuples $\mathtt{t}$ and $\mathtt{t}'$ are type comparable if each position between the two tuples agrees on the type of term they contain, i.e., constants are mapped to (the same) constants, variables to variables and labeled nulls to labeled nulls.

Definition 2 (Position comparable tuples). *Let $\mathtt{t} = (t_1, \cdots, t_n)$ and $\mathtt{t}' = (t'_1, \cdots, t'_n)$ be two tuples of terms with length n. We say that $\mathtt{t}$ and $\mathtt{t}'$ are* position comparable *(or simply called* comparable*), denoted as $\mathtt{t} \sim \mathtt{t}'$, if $\mathtt{t}$ and $\mathtt{t}'$ satisfy the following conditions:*

1. *$\mathtt{t}$ and $\mathtt{t}'$ are type comparable;*

2. *for each pair (i, j) ($1 \leq i < j \leq n$), $t_i = t_j$ iff $t'_i = t'_j$;*

3. *for each $t \in \left(\mathtt{t} \cap \mathtt{t}'\right)$, $t_i = t$ iff $t'_i = t$ ($1 \leq i \leq n$);*

4. *for each* i *(*$1 \leq i \leq n$*),* $t_i \in \big(\mathbf{t} \setminus \mathbf{t}'\big) \cap \big(\Gamma_V \cup \Gamma_N\big)$ *iff* $t'_i \in \big(\mathbf{t}' \setminus \mathbf{t}\big) \cap \big(\Gamma_V \cup \Gamma_N\big)$[3].

We also use $\mathbf{t_1} \not\sim \mathbf{t}_2$ *if it is not the case that* $\mathbf{t_1} \sim \mathbf{t}_2$.

Under Definition 2, we have $(X, X', \mathsf{n}) \sim (Z, Y, \mathsf{n}')$, but $(\mathsf{n}, \mathsf{n}, \mathsf{n}', Z) \not\sim (\mathsf{n}, \mathsf{n}', \mathsf{n}', W)$, because in the latter, the null patterns in the first three positions of the two tuples are not "comparable".

Let X be a variable from Γ_V, and t a term from $\Gamma \cup \Gamma_N \cup \Gamma_V$. A *binding* is an expression of the form X/t. In this case, we also say that t is a binding of variable X. A *substitution* $[\mathbf{X}/\mathbf{t}]$ is a finite set of bindings containing at most one binding for each variable from $\mathbf{X}$. For a given tuple of terms $\mathbf{t}$, we can apply a substitution θ to $\mathbf{t}$ and obtain a different tuple of terms, denoted as $\mathbf{t}\theta$. For example, $(X, Y, \mathsf{n}, W)[X/\mathsf{n}', Y/Y, W/Z] = (\mathsf{n}', Y, \mathsf{n}, Z)$. Naturally, for a quantifier-free formula $\varphi(\mathbf{X})$ and a substitution $\theta = [\mathbf{X}/\mathbf{t}]$, applying θ to $\varphi(\mathbf{X})$, i.e., $\varphi(\mathbf{X})\theta$, will result in formula $\varphi(\mathbf{t})$ which is obtained from $\varphi(\mathbf{X})$ by replacing each free variable X by its corresponding binding from $\varphi(\mathbf{X})$.

Now we define how a substitution is applied to an existential rule σ. We extend a substitution to existentially quantified variables. We say that substitution $\theta = [\mathbf{X}/\mathbf{t}]$ is *applicable* to σ if the arities of $\mathbf{X}$ in θ match the arities of the tuples of all universally and existentially quantified variables in σ, respectively. We may write a substitution applicable to σ as the form: $\theta = [\mathbf{X}/\mathbf{t}_1, \mathbf{Y}/\mathbf{t}_2, \mathbf{Z}/\mathbf{n}]$. Then by applying θ to rule σ of the form (3), we will obtain a rule of the following form:

$$\sigma\theta : \varphi(\mathbf{t}_1, \mathbf{t}_2) \rightarrow p(\mathbf{t}_1, \mathbf{n}). \tag{4}$$

Note that in (4) existentially quantified variables are always substituted by nulls.The motivation for extending a substitution to existentially quantified variables is quite clear. For a given set Σ of TGDs, we want to represent the underneath derivation of Σ in a generic form so that such derivation may be instantiated by the chase procedure when a specific input database is taken into account. For this purpose, through a substitution, we not only substitute those universally quantified variables in σ, but also intentionally eliminate existentially quantified variables in $\mathsf{head}(\sigma)$ by replacing them with proper nulls. In this way, atom $p(\mathbf{t}_1, \mathbf{n})$ may be used in triggering other rules of Σ through further substitutions.

[3]Note that this condition can be implied from the previous conditions 1, 2 and 3. But for clarity, we still keep this condition in definition.

Definition 3 (Derivation path). *Let Σ be a set of TGDs. A* derivation path P *of Σ is a finite sequence of pairs (α, ρ) of an atom α and a rule ρ:*

$$(\alpha_1, \rho_1), \cdots, (\alpha_n, \rho_n), \tag{5}$$

such that

- *for each $1 \leq i \leq n$, $\alpha_i = \mathsf{head}(\rho_i)$;*
- *for each $1 \leq i \leq n$, $\rho_i = \sigma_i\theta_i$ for some $\sigma_i \in \Sigma$ and substitution θ_i;*
- *for each $1 \leq i < n$, $\alpha_{i+1} \in \mathsf{body}(\rho_i)$;*
- *for each $1 \leq i \leq n$, if a null $\mathsf{n} \in \mathsf{head}(\alpha_i)$ is introduced due to the elimination of existentially quantified variable, then this n must not occur in ρ_j, for all $j \in \{i+1, \ldots, n\}$.*

The following example explains the intuition of derivation paths.

Example 2. Consider a set Σ of TGDs consisting of two rules:

$$\sigma_1 : r(X, Y, Z) \rightarrow s(Y, X),$$
$$\sigma_2 : s(X, Y) \rightarrow \exists Z \exists W r(Y, Z, W).$$

The following are three different derivation paths of Σ:

P_1:
$(s(\mathsf{n}_1, Y_1), \sigma_1[X/Y_1, Y/\mathsf{n}_1, Z/\mathsf{n}_2])$,
$(r(Y_1, \mathsf{n}_1, \mathsf{n}_2), \sigma_2[X/X_1, Y/Y_1, Z/\mathsf{n}_1, W/\mathsf{n}_2])$,
$(s(X_1, Y_1), \sigma_1[X/Y_1, Y/X_1, Z/Z_1])$,

P_2:
$(r(X_2, \mathsf{n}_1, \mathsf{n}_2), \sigma_2[X/\mathsf{n}_3, Y/X_2, Z/\mathsf{n}_1, W/\mathsf{n}_2])$,
$(s(\mathsf{n}_3, X_2), \sigma_1[X/X_2, Y/\mathsf{n}_3, Z/\mathsf{n}_4])$,

P_3:
$(r(X_2, \mathsf{n}_1, \mathsf{n}_2), \sigma_2[X/\mathsf{n}_3, Y/X_2, Z/\mathsf{n}_1, W/\mathsf{n}_2])$,
$(s(\mathsf{n}_3, X_2), \sigma_1[X/X_2, Y/\mathsf{n}_3, Z/\mathsf{n}_4])$,
$(r(X_2, \mathsf{n}_3, \mathsf{n}_4), \sigma_2[X/X_1, Y/X_2, Z/\mathsf{n}_3, W/\mathsf{n}_4])$.

Intuitively, P_1 represents a *pattern* for deriving atom $s(\mathsf{n}_1, Y_1)$ from rules σ_1 and σ_2. For a given database D, atom $s(\mathsf{n}_1, t)$ will be derived following the same pattern with different instantiations of free variables with constants from D. Similarly, P_2 and P_3 represent two different derivation patterns for atom $r(X_2, \mathsf{n}_1, \mathsf{n}_2)$, respectively. $\square$

Definition 4 (Generalising comparability relation). *We generalise the comparability relation $\sim$ defined earlier as follows.*

1. *Let $\sigma \in \Sigma$, and $\theta = [\mathbf{X}/\mathbf{t}, \mathbf{Z}/\mathbf{n}]$ and $\theta' = [\mathbf{X}/\mathbf{t}', \mathbf{Z}/\mathbf{n}']$ be two substitutions applicable to σ. We say that $\sigma\theta$ and $\sigma\theta'$ are* comparable*, denoted as $\sigma\theta \sim \sigma\theta'$, if $\mathbf{tn} \sim \mathbf{t}'\mathbf{n}'$.*

2. *Let P be a derivation path of Σ of the form (5), we use $|P|$ to denote its length. Furthermore, suppose (α_i, ρ_i) and (α_j, ρ_j) are two elements of P, we say that (α_i, ρ_i) and (α_j, ρ_j) are* comparable*, denoted as $(\alpha_i, \rho_i) \sim (\alpha_j, \rho_j)$, if $\rho_i \sim \rho_j$.*

3. *Let $P = ((\alpha_1, \rho_1), (\alpha_2, \rho_2), \cdots)$ and $P' = ((\alpha'_1, \rho'_1), (\alpha'_2, \rho'_2), \cdots)$ be two derivation paths of Σ. P and P' are* comparable*, denoted as $P \sim P'$, if $|P| = |P'|$ and for each i ($1 \leq i \leq |P|$), $(\alpha_i, \rho_i) \sim (\alpha'_i, \rho'_i)$.*

It is easy to observe that $\sim$ defined in Definition 4 is an equivalence relation. In the rest of this paper, whenever we use symbol $\sim$, we always refer to the comparability relation defined in Definition 4.

Although a derivation path may be infinitely long, the following result ensures that for any derivation path, it is sufficient to only consider its finite fragment.

Proposition 1 (Derivation path length bound). *Let Σ be a set of TGDs. There exists a natural number N such that for every derivation path P of the form (5), if $|P| > N$ then there exist i, j ($1 \leq i < j \leq |P|$) such that $(\alpha_i, \rho_i) \sim (\alpha_j, \rho_j)$.*

3.2 Derivation trees

Now we are ready to define a derivation tree which is a key concept for our following investigations.

For the following, given a tree $\mathcal{T} = (N, E, \lambda)$ and node $v \in N$, we denote by $\mathsf{subtree}_{\mathcal{T}}(v)$ as the subtree of $\mathcal{T}$ rooted at v and such that if there is another subtree $\mathcal{T}'$ of $\mathcal{T}$ that is also rooted at v, then $\mathcal{T}' \subseteq \mathsf{subtree}_{\mathcal{T}}(v)$, i.e., $\mathsf{subtree}_{\mathcal{T}}(v)$ is the *maximal* subtree of $\mathcal{T}$ that is rooted on v.

Definition 5 (Derivation tree). *Given a set Σ of TGDs. A* derivation tree *of Σ, denoted as $T(\Sigma)$, is a finite tree (N, E, λ), with nodes N, edges E and labelling function λ, such that:*

1. *The nodes of* $T(\Sigma)$ *have* labels *of the form* (α, ρ)*, where* $\rho = \sigma\theta$ *for some* $\sigma \in \Sigma$ *and* θ *a substitution, and* $\mathsf{head}(\rho) = \alpha$*;*

2. *For any node* v *labeled by* (α, ρ) *of* $T(\Sigma)$*, let* $\alpha_1, \cdots, \alpha_n$ *be atoms in* $\mathsf{body}(\rho)$*, then* (α, ρ) *has* n *children* $v_1, \ldots, v_n$ *labeled with* (α_1, ρ_1)*,* $\cdots$*,* (α_n, ρ_n)*, respectively, such that for each* $i \in \{1, \ldots, n\}$*,* $\rho_i = \sigma_i\theta_i$ *for some* $\sigma_i \in \Sigma$ *and substitution* θ_i*, and* $\mathsf{head}(\rho_i) = \alpha_i$*;*

3. *For any node* v *labeled with* (α, ρ) *in* $T(\Sigma)$*, all "fresh" nulls* n_i *occurring in* α*, that are introduced through the substitutions in* ρ*, we have that:*

   (a) n_i *must not occur in any labels of a descendant node of* v*;*

   (b) *if* v' *is another node labelled with* (α', ρ') *in* $T(\Sigma)$ *such that* n_i *is also a "fresh" null in* α' *(i.e.,* n_i *is introduced by both* (α, ρ) *and* (α', ρ')*), then* $\mathit{subtree}_{\mathcal{T}}(v') = \mathit{subtree}_{\mathcal{T}}(v)$*;*

4. *If node* v *labeled with* (α, ρ) *is a leaf of* $T(\Sigma)$*, then there does not exist any null* n *appearing in* $\mathsf{body}(\rho)$*.*

A path P *in* $T(\Sigma)$*, denoted as* $P \in T(\Sigma)$*, is a derivation path in* $T(\Sigma)$ *starting from the root and ending at a leaf. We define* $\mathsf{depth}(T(\Sigma)) = \mathsf{max}(\{|P| \mid P \in T(\Sigma)\}$ *to be the* depth *of* $T(\Sigma)$*. By* $\mathsf{root}(T(\Sigma))$*,* $\mathsf{leafNodes}(T(\Sigma))$ *and* $\mathsf{nodes}(T(\Sigma))$*, we denote the root node, the set of leaf nodes and the set of all nodes of* $T(\Sigma)$*, respectively. Also, given a node* v *of* $T(\Sigma)$*, we denote by* $\mathsf{childNodes}(v, T(\Sigma))$ *(or just* $\mathsf{childNodes}(v)$ *when clear from the context) as the set of children nodes of* v *under the tree* $T(\Sigma)$*. Lastly, we use* $\mathcal{T}(\Sigma)$ *to denote the set of all derivation trees of* Σ*.*

According to Definition 5, each path of a derivation tree is a derivation path. Also, since Σ can have an infinite number of possible derivation paths due to possibly arbitrary number of repetitions of path fragments within a path (i.e., a "loop"), $\mathcal{T}(\Sigma)$ may contain an infinite number of derivation trees.

By providing a specific database D, a derivation tree $T(\Sigma)$ is then instantiated, as the following defines.

Definition 6 (Derivation tree instantiation). *Let* Σ *be a set of TGDs,* D *a database over schema* $\mathcal{R}$*, and* $T(\Sigma) = (N, E, \lambda)$ *a derivation tree of* Σ*. Then we obtain a tree* $T' = (N', E', \lambda')$ *from* $T(\Sigma)$*, where* $N \subseteq N'$*, as follows:*

1. *For each leaf node* v *in* $T(\Sigma)$*, where* $\lambda(v) = (\alpha, \rho)$*, do:*

(a) *Set* $\lambda'(v) = (\alpha', \rho')$ *in the tree* T'*, where* $\alpha' = \mathsf{head}(\rho')$*,* $\rho' = \rho\theta$ *for some substitution* θ*, and* $\mathsf{body}(\rho') \subseteq D$*;*

(b) *For each atom* $\beta \in \mathsf{body}(\rho') \subseteq D$ *with* ρ' *as mentioned above, add a node* v' *in* N' *and set* $\lambda'(v') = (\beta, \beta)$ *and a corresponding edge* $\langle v, v' \rangle$ *in* E' *so that* v' *is a leaf node (i.e., we make* v *be a non-leaf node in* T'*);*

2. *For a non-leaf node* v *in* $T(\Sigma)$*, such that* $\lambda(v) = (\alpha, \rho)$*, and all labels of its children have been replaced as in 1 above (i.e., through "*λ'*"), set* $\lambda'(v) = (\alpha', \rho')$*, where* $\rho' = \rho\theta'$ *for some substitution* θ' *such that for each atom* $p(\mathbf{t}) \in \mathsf{body}(\rho')$*, either* $p(\mathbf{t}) \in D$ *or there exists a child node* v' *of* v *such that* $\lambda'(v') = (\alpha^*, \rho^*)$*, where* $p(\mathbf{t}) = \alpha^*$*;*

3. *Continue 2, until no node can be further relablled.*

T' *is called an* instantiation *of* $T(\Sigma)$ *on* D*, denoted as* $T(D, \Sigma)$*, if it does not contain any variables occurring in* $T(\Sigma)$*. Similarly to the case of derivation tree, we use* $\mathsf{depth}(T(D, \Sigma))$ *and* $\mathsf{root}(T(D, \Sigma))$ *to denote the* depth *and* root node *of* $T(D, \Sigma)$*, respectively. Finally, by* $\mathcal{T}(D, \Sigma)$*, we denote the set of all instantiations on* D *for all derivation trees in* $\mathcal{T}(\Sigma)$*.*

By examining Definition 6, we can see that the intuition of derivation tree instantiation $T(D, \Sigma)$ is simply an extension of the derivation tree $T(\Sigma)$ by replacing all variables with some constants from D in a bottom-up manner. After completing such process, the tree $T(D, \Sigma)$ only contains constants from D and nulls.

For convenience from now on and when it is clear from the context, we will mostly refer to a node by its actual label, e.g., for a node $v \in N$ where $\lambda(v) = (\alpha, \rho)$, it is simply referred as (α, ρ).

We say that an atom $p(\mathbf{t})$ *is supported* by $T(D, \Sigma)$, denoted as $T(D, \Sigma) \models p(\mathbf{t})$, if $\lambda(\mathsf{root}(T(D, \Sigma))) = (\alpha, \rho)$ where $\alpha = p(\mathbf{s})$, and there is a homomorphism h such that $h(p(\mathbf{t})) = p(\mathbf{s})$. The following result reveals an important relationship between the chase and derivation trees.

Theorem 3. *Let* Σ *be a set of TGDs,* D *a database over schema* $\mathcal{R}$*, and* q *a BCQ query* $\exists \mathbf{Z} p(\mathbf{Z})$*. Then* $\mathsf{chase}(D, \Sigma) \models q$ *iff there exist an instantiation* $T(D, \Sigma)$ *for some derivation tree* $T(\Sigma)$ *and a substitution* θ*, such that* $T(D, \Sigma) \models p(\mathbf{t})$*, where* $\mathbf{t}$ *is a tuple of terms from* Γ *of the same length as* $\mathbf{Z}$*, and* $\mathbf{t}\theta = \mathbf{Z}$*.*

4 Loop Restricted (LR) TGDs

Theorem 3 shows that derivation trees provide a precise characterisation for the chase procedure. Therefore, the query answering against a set of TGDs together with an input database can be achieved by computing and checking the corresponding instantiation of the underlying derivation tree. However, since the derivation tree for a given set of TGDs may be of an arbitrary depth, this process is generally undecidable.

In this section, we will define a new class of TGDs, named *loop restricted* (LR) TGDs, such that the depth of all derivation trees for this type of TGDs is always bounded in some sense. From this result, we will further prove that LR TGDs satisfy the *bounded derivation-depth property* (BDDP) [11].

Definition 7 (Loop pattern). *Let $P = ((\alpha_1, \rho_1), \cdots, (\alpha_n, \rho_n))$ be a derivation path as defined in Definition 3. Then P is a* loop pattern *if $(\alpha_1, \rho_1) \sim (\alpha_n, \rho_n)$ and $(\alpha_i, \rho_i) \not\sim (\alpha_j, \rho_j)$ for any other i, j ($1 \leq i, j \leq n$ and $i \neq j$).*

Let L be a loop pattern as defined in Definition 7. For each pair (α_i, ρ_i) in L where $1 \leq i < n$, an atom $\beta \in \mathsf{body}(\rho_i)$ is called *recursive atom* if $\beta = \alpha_{i+1}$ for $(\alpha_{i+1}, \rho_{i+1})$.

Example 3. Example 2 continued. It is easy to see that derivation paths P_3 is a loop pattern, while P_1 and P_2 are not. Furthermore, P_3 is the only loop pattern of the given Σ, considering that for all other loop patterns P, it will be $P \sim P_3$[4]. $\square$

Proposition 2. *Given a finite set Σ of TGDs, Σ only has a finite number of loop patterns under the equivalence relation $\sim$.*

4.1 Restricted loop patterns

Example 4. Consider Example 1 in Introduction again, where $\Sigma_{\mathsf{inCharge}}$ is represented as the following set of TGDs Σ by renaming predicates:

$$\sigma_1 : \quad \mathsf{r}(X, W) \wedge \mathsf{t}(X, Y) \wedge \mathsf{t}(Y, Z) \rightarrow \mathsf{s}(X, Z),$$
$$\sigma_2 : \quad \mathsf{s}(X, Z) \rightarrow \exists U \mathsf{r}(X, U).$$

With a careful verification, we can see that Σ does not belong to any of currently known first-order rewritable TGDs classes.

[4] See Definition 4 for derivation path (loop pattern) comparability relation.

Now we consider the derivation of atom $\mathbf{r}(X_1, \mathsf{n}_1)$ from Σ. The following is a derivation tree for $\mathsf{r}(X_1, \mathsf{n}_1)$, which involves recursive calls to σ_1 and σ_2:

$$
\begin{aligned}
&T: \\
&w_0 = (\alpha_0, \rho_0) \\
&\quad = \big(\mathsf{r}(X_1, \mathsf{n}_1), [\mathsf{s}(X_1, Z_1) \rightarrow \mathsf{r}(X_1, \mathsf{n}_1)]\big), \\
&w_1 = (\alpha_1, \rho_1) \\
&\quad = \big(\mathsf{s}(X_1, Z_1), [\mathsf{r}(X_1, \mathsf{n}_2) \wedge \mathsf{t}(X_1, Y_1) \wedge \mathsf{t}(Y_1, Z_1) \rightarrow \mathsf{s}(X_1, Z_1)]\big), \\
&w_2 = (\alpha_2, \rho_2) \\
&\quad = \big(\mathsf{r}(X_1, \mathsf{n}_2), [\mathsf{s}(X_1, Z_2) \rightarrow \mathsf{r}(X_1, \mathsf{n}_2)]\big).
\end{aligned}
$$

Intuitively, T simply contains one derivation path $L = w_0 w_1 w_2$ which is also a loop pattern. If we consider all other derivation trees for atom $\mathsf{r}(X_1, \mathsf{n}_1)$, it is not difficult to observe that all these trees *are subsumed* by T, in the sense that derivations illustrated in T cover those illustrated in all other trees. Informally, we say that a derivation tree T subsumes another derivation tree T', if T and T' share the same tree root, and each derivation path of T is contained by some derivation path presented in T'. We need to mentioned that in order to simplify the presentation, we intently delay our formal definition of tree subsumption to the proof appendix (see Definition 11).

Σ presents an interesting case of satisfying the so-called *bounded derivation tree depth property* (BDTDP) (the definition will be given later). By examining the loop pattern, we find that it can be split in such a way where all variables in the recursive atoms are bounded by the variables occurring in the heads of all corresponding rules. This will make the derived atom in each derivation step from the corresponding derivation tree not rely on any new variables in recursive atoms.

In more details, consider loop pattern L, for each pair (α_i, ρ_i) $(i = 0, 1, 2)$, we can split the set $\mathsf{body}(\rho_i)$ of atoms in the body of ρ_i^1 into two disjoint parts $\mathsf{body_h}(\rho_i)$ and $\mathsf{body_b}(\rho_i)$, such that the common variables in $\{\alpha_i\} \cup \mathsf{body_h}(\rho_i)$ and $\mathsf{body_b}(\rho_i)$ are exactly the common variable occurring in all α_i, which is X_1, whilst the underlying recursive atoms in the loop pattern only occur in $\mathsf{body_b}(\rho_i)$, i.e., $\alpha_{i+1} \in \mathsf{body_b}(\rho_i)$ for $i = 0, 1$. As will be showed next, it turns out that a set Σ of TGDs having this feature always ensures BDTDP. $\square$

Now we are ready to formally define the notion of loop restricted patterns. Let A be a set of atoms, we use $\mathsf{var}(A)$ to denote the set of all variables occurring in A.

Definition 8 (Loop restricted (LR) patterns). *Let Σ be a set of TGDs. Σ is* loop restricted *(LR), if for each loop pattern $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ of Σ, L satisfies the following conditions: for each pair (α_i, ρ_i) in L $(1 \leq i < n)$, the set of atoms $\mathsf{body}(\rho)$ of the body of ρ can be separated into two disjoint parts $\mathsf{body}(\rho_i) = \mathsf{body_h}(\rho_i) \cup \mathsf{body_b}(\rho_i)$, such that*

- $\mathsf{body_h}(\rho_i) \cap \mathsf{body_b}(\rho_i) = \emptyset$,
- $\alpha_{i+1} \in \mathsf{body_b}(\rho_i)$,
- $\mathsf{var}(\{\alpha_i\} \cup \mathsf{body_h}(\rho_i)) \cap \mathsf{var}(\mathsf{body_b}(\rho_i)) = \bigcap_{j=1}^{n} \mathsf{var}(\alpha_j)$.

Example 5. Example 4 continued. It is easy to see that loop pattern L in Example 4 satisfies the conditions of Definition 8. Furthermore, if we consider the derivation of atom $\mathsf{s}(X, Z)$ from Σ, the underlying loop patterns deduced from its derivations also satisfy the conditions of Definition 8. So Σ is loop restricted. $\square$

4.2 Main results

Now we study the main properties of the new class LR of TGDs. We first define a property called bounded derivation tree depth property (BDTDP).

Definition 9 (BDTDP). *A class $\mathcal{C}$ of TGDs satisfies the* bounded derivation tree depth property *(BDTDP) if for each $\Sigma \in \mathcal{C}$, there exists some $k \geq 0$ such that for every BCQ query $\exists \mathbf{Z} p(\mathbf{Z})$ and every database D, $D \cup \Sigma \models \exists \mathbf{Z} p(\mathbf{Z})$ iff $T(D, \Sigma) \models p(\mathbf{n})$ for some instantiated derivation tree $T(D, \Sigma)$ and atom $p(\mathbf{n})$, where $\mathsf{depth}(T(D, \Sigma)) \leq k$ and $h(\mathbf{Z}) = \mathbf{n}$ for some homomorphism h.*

Basically, Definition 9 states that for any $\Sigma \in \mathcal{C}$, BDTDP ensures that all of its BCQ query answering problem can be always decided within a fixed number k of derivation steps with respect to the corresponding instantiated derivation trees. Note that this k is independent from the input database D and the specific BCQ query q. Also note that BDTDP is different from the previous BDDP, i.e., Definition 1, which is defined based on the chase procedure.

Theorem 4. *The class of LR TGDs satisfies BDTDP.*

The following theorem reveals an important connection between BDTDP and BDDP.

Theorem 5. *If a class $\mathcal{C}$ of TGDs satisfies BDTDP then $\mathcal{C}$ also satisfies BDDP.*

According to Theorem 5 from [11], it is clear that the class of LR TGDs is first-order rewritable.

Theorem 6. *For the class of LR TGDs, the BCQA's data complexity is in* AC^0*, and the combined complexity is* 2-EXPTIME *complete.*

Theorem 7. *Deciding whether a set of TGDs is loop restricted is* PSPACE *complete.*

5 Generalisation of Loop Restricted Patterns

As described in previous section, the notion of loop patterns provides a useful means of defining the class LR of TGDs that is first-order rewritable. Now we show that loop patterns can be employed as a unified notion to significantly extend LR TGDs to a more general class of TGDs.

Definition 10 (Generalised loop restricted (GLR) patterns). *Let* Σ *be a set of TGDs.* Σ *is* generalised loop restricted *(GLR), if each loop pattern* $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ *of* Σ *falls into one of the following five types:*

Type I *For each pair* (α_i, ρ_i) *in* L *(*$1 \leq i < n$*),* $\mathsf{body}(\rho_i)$ *can be separated into two disjoint parts* $\mathsf{body}(\rho_i) = \mathsf{body_h}(\rho_i) \cup \mathsf{body_b}(\rho_i)$ *such that the following three conditions holds:*

1. $\mathsf{body_h}(\rho_i) \cap \mathsf{body_b}(\rho_i) = \emptyset$*,*
2. $\alpha_{i+1} \in \mathsf{body_b}(\rho_i)$*,*
3. $\mathsf{var}\big(\{\alpha_i\} \cup \mathsf{body_h}(\rho_i)\big) \cap \mathsf{var}\big(\mathsf{body_b}(\rho_i)\big) = \bigcap_{j=1}^{n} \mathsf{var}(\alpha_j)$*;*

Type II *There exists a pair* (α_i, ρ_i) *in* L *(*$1 < i \leq n$*) such that* $\mathsf{body}(\rho_i)$ *can be separated into two disjoint parts* $\mathsf{body}(\rho_i) = \mathsf{body_h}(\rho_i) \cup \mathsf{body_b}(\rho_i)$*, where the following three conditions hold:*

1. $\mathsf{body_h}(\rho_i) \cap \mathsf{body_b}(\rho_i) = \emptyset$*,*
2. $\alpha_{i+1} \in \mathsf{body_b}(\rho_i)$*,*
3. $\mathsf{var}\big(\{\alpha_i\} \cup \mathsf{body_h}(\rho_i)\big) \cap \mathsf{var}\big(\mathsf{body_b}(\rho_i)\big) = \emptyset$*;*

Type III *For each pair* (α_i, ρ_i) *in* L *(*$1 \leq i < n$*) and each* $\beta \in \mathsf{body}(\rho_i)$*,* $\mathsf{var}(\rho_i) \subseteq \mathsf{var}(\beta)$*;*

Type IV *For each pair* (α_i, ρ_i) *in* L *(*$1 \leq i < n$*) and each* $\beta \in \mathsf{body}(\rho_i) \setminus \{\alpha_{i+1}\}$, $\big(\mathsf{var}(\alpha_{i+1}) \cap \mathsf{var}(\beta)\big) \neq \emptyset$ *implies* $\big(\mathsf{var}(\alpha_{i+1}) \cap \mathsf{var}(\beta)\big) \subseteq \bigcap_{j=1}^{i} \mathsf{var}(\alpha_j)$*;*

Type V *There exists a pair* (α_i, ρ_i) *in* L *(*$1 \leq i < n$*), such that* $\mathsf{body}(\rho_i)$ *can be separated into two disjoint parts* $\mathsf{body}(\rho_i) = \mathsf{body_h}(\rho_i) \cup \mathsf{body_b}(\rho_i)$*, where the following three conditions hold:*

1. $\mathsf{body_h}(\rho_i) \cap \mathsf{body_b}(\rho_i) = \emptyset$*;*
2. $\bigcap_{j=i+1}^{n} \mathsf{body_h}(\rho_i) = \emptyset$*,*
3. $\mathsf{null}\big(\mathsf{body_h}(\rho_i)\big) \neq \emptyset$*.*

Let us take a closer look at Definition 10. Firstly, Type I simply specifies LR TGDs, so the class of GLR TGDs properly contains the class of LR TGDs. Type II looks similar to Type I, but it differs from Type I in the following way: for the body part of ρ_i containing the recursive atom in the loop pattern, i.e., $\alpha_{i+1} \in \mathsf{body_b}(\rho_i)$, its variables are not in common with variables occurring in the head α_i and the other part of the body $\mathsf{body_h}(\rho_i)$. This indicates that recursion embedded in the underlying loop pattern will not actually happen due to the lack of shared variables.

Type III, on the other hand, says that for each rule ρ_i in every loop pattern, all variables occurring in ρ_i are guarded by each atom in ρ_i's body. Type IV concerns the shared variables occurring in both recursive and non-recursive atoms in the body of rule ρ_i in a loop pattern, i.e., $\mathsf{var}(\alpha_{i+1}) \cap \mathsf{var}(\beta)$[5]. It requires that all such shared variables must be passed on to all following rules in the loop pattern. Finally, Type V ensures that no cycle occurs in Σ's graph of rule dependencies.

Theorem 8. *The class of GLR TGDs satisfies BDTDP.*

According to Theorem 5, we know that the class of GLR TGDs satisfying BDTDP also satisfies BDDP, and hence the following corollary holds.

Corollary 9. *The class of GLR TGDs is first-order rewritable.*

Theorem 10. *Consider the BCQA problem for a given set of GLR TDGs. Its data complexity is in* AC^0*, and its combined complexity is* 2-ExpTime *complete.*

Theorem 11. *Deciding whether a set of TGDs is generalised loop restricted is* PSpace *complete.*

[5]Remember that $\alpha_{i+1} \in \mathsf{body}(\rho_i)$, and the derivation is backward from ρ_{i+1} to ρ_i in loop pattern L.

6 Relationship to Other First-order Rewritable Classes

Generalised loop restricted pattern defined in Definition 10 is a useful notion and concept to capture a broader class of first-order rewritable TGDs. In fact, as we will show in this section, the class of LR TGDs properly captures many currently known such classes.

We first briefly introduce these existing TGDs classes, which are known to be first-order rewritable. A TGD of the form (1):

$$\sigma : \forall \mathbf{X}\mathbf{Y}\varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\psi(\mathbf{X}, \mathbf{Z}),$$

is called *linear* if $\varphi(\mathbf{X}, \mathbf{Y})$ is an atom. σ is *multi-linear* if each atom in φ contains all the universally quantified variables of σ [10]. A set Σ of TGDs is linear or multi-linear if each TGD in Σ is linear or multi-linear, respectively.

Σ is said to be *acyclic* if for each $\sigma \in \Sigma$, the hypergraph of the relations on the left hand-side of σ is acyclic, as well as the hypergraph of the right hand-side of σ, considering only existentially quantified variables $\mathbf{Z}$. In particular, let (V, E) be the hypergraph of σ's left hand-side relations, where vertices in V are the variables of $\mathbf{X} \cup \mathbf{Y}$, and edges in E are relation atoms of $\varphi(\mathbf{X}, \mathbf{Y})$. The hypergraph (V', E') of σ's right hand-side relations is defined in the same way, except whose edges are the relation atoms of $\psi(\mathbf{X}, \mathbf{Z})$ containing at least one variable from $\mathbf{Z}$. Then σ is *acyclic* if both (V, E) and (V', E') are acyclic [5, 29].

Rule σ of the form (1) is called *domain-restricted* if each atom of $\psi(\mathbf{X}, \mathbf{Z})$ contains all or none of the variables in $\varphi(\mathbf{X}, \mathbf{Y})$ [4]. Moreover, we say that σ is called *proper domain-restricted* if $\mathbf{X} \neq \emptyset$. As in practice, the case that $\mathbf{X} = \emptyset$ is of much less interest, from here on, unless stated otherwise, we will assume that all mentions of domain-restricted are proper domain-restricted. Then finally, we say that Σ is (proper) *domain-restricted* if each $\sigma \in \Sigma$ is (proper) domain-restricted.

Σ is said to have the *sticky property* if for each σ in Σ, all variables occurring in $\mathsf{b}ody(\sigma)$ more than once also appear in $\mathsf{head}(\sigma)$, and furthermore, also appear in every atom obtained from some chase derivation which involves $\mathsf{head}(\sigma)$, that is, *stick* to all such atoms [11]. The *sticky-join property*, on the other hand, is less restricted than sticky property, where it only requires to stick certain variables occurring more than once in $\mathsf{body}(\sigma)$ based on certain joinless condition. It has been showed that the sticky-join class captures both the sticky and linear classes, but is incomparable with multi-linear class [11].

Σ is *aGRD* if Σ's rule dependency graph contains no cycle [3, 4].

Let us use LR, ML, AC, DR, SJ and aGRD to denote the classes of loop restricted, mulit-linear acyclic, domain-restricted, sticky-join, aGRD, and domain restricted, respectively. Then we have the following result.

Proposition 3. *Let GLR be the class of generalised loop restricted TGDs defined in Definition 10. Then we have that: (1)* RL $\subsetneq$ GLR*; (2)* ML $\subsetneq$ GLR*; (3)* AC $\subsetneq$ GLR*; (4)* SJ $\subsetneq$ GLR*; (5)* aGRD $\subsetneq$ GLR*; (6)* DR $\subsetneq$ GLR.

Intuitively, in GLR class, the subclasses of Type I, II, III, IV and V properly contain LR class, AC class, ML class, SJ class, and aGRD class, respectively, while class DR is properly contained by LR class and hence by Type I subclass of GLR.

It has been shown in [14] that the LOA class captures all existing known first-order rewritable classes. The following result, however, shows that LOA and GLR are actually two incomparable first-order rewritable classes.

Proposition 4. *We have that* GLR $\not\subseteq$ LOA *and* LOA $\not\subseteq$ GLR.

Proof. ("GLR $\not\subseteq$ LOA") From the proof of Theorem 5 in [15], we consider a set Σ of simple TGDs comprising of the following two rules:

$$\mathsf{s}(X, Y, Z, V) \rightarrow \mathsf{r}(X, Y, Z), \tag{6}$$

$$\mathsf{t}(X, W) \wedge \mathsf{r}(X, W, Y) \rightarrow \exists Z\, \mathsf{s}(X, Y, Z, W). \tag{7}$$

Then we can show that Σ is not in the GLR class of simple TGDs.

("LOA $\not\subseteq$ GLR") Consider again the set of TGDs $\Sigma_{\mathsf{inCharge}}$ in Example 4:

$$\sigma_1: \quad \mathsf{r}(X, W) \wedge \mathsf{t}(X, Y) \wedge \mathsf{t}(Y, Z) \rightarrow \mathsf{s}(X, Z),$$
$$\sigma_2: \quad \mathsf{s}(X, Z) \rightarrow \exists U \mathsf{r}(X, U).$$

Then it can be checked that $\Sigma_{\mathsf{inCharge}}$ is not LOA because we will have a cycle $\langle \sigma_1, \sigma_2 \rangle$, $\langle \sigma_2, \sigma_1 \rangle$ in the *labeled oblivious STGD graph* [14] of Σ and where the edge $\langle \sigma_1, \sigma_2 \rangle$ will have both an m and s labels. On the other hand, we have that Σ is LR as we outlined in Example 5. □

7 Concluding Remarks

Loops have been an important concept in the study for traditional Datalog programs, and then have been employed and extended in Answer Set Programming research in recent years, e.g., [13, 25, 30, 31]. In this paper, through a

series of novel definitions of derivation paths, derivation trees and loop patterns, we are able to discover new decidable classes of TGDs for ontology based query answering using a very different idea from previous approaches.

As we have showed, the class of GLR TGDs properly contains most of other first-order rewritable TGDs classes, we believe that our results presented in this paper will be useful in developing efficient OBDA systems for borader application domains.

References

[1] V. Asuncion, Y. Zhang, H. Zhang, and Y. Bai, Loop restricted existential rules and first-order rewritability for query answering (extended abstract). In *Proceedings of KR-2018*, 2018.

[2] F. Baader, M. Bienvenu, C. Lutz, and F. Wolter, Query and predicate emptiness in ontology-based data access. *Journal of Artificial Intelligence Research* 56 (2016) 1-59.

[3] J-F. Baget, Improving the forward chaining algorithm for conceptual graphs rules. In *Proceedings of KR-2004*, pp 407-414, 2004.

[4] J-F. Baget, M. Leclère, M-L. Mugnier, and E. Salvat, On rules with existential variables: Walking the decidability line. *Artificial Intelligence* 9-10 (2011) 1620-1654.

[5] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, and M. Yannakakis, Properties of acyclic database. In *Proceedings of STOC-1981*, 1981.

[6] M. Benedikt, G. Konstantinidis, G. Mecca, B. Motik, P. Papotti, D. Santoro, and E. Tsamoura, Benchmarking the chase. In *Proceedings of ACM SIGMOD-SIGACT-SIGAI-2017*, 2017.

[7] M. Bienvenu, Ontology-mediated query answering: Harnessing knowledge to get more from data. In *Proceedings of IJCAI-2016*, pp 4058-4061, 2016.

[8] P. Bourhis, M. Leclre, M-L. Mugnier, S. Tison, F. Ulliana, and L. Gallois, Oblivious and semi-oblivious boundedness for existential rules. In *Proceedings of IJCAI-2019*, pp 1581- 1587, 2019.

[9] A. Calì, G. Gottlob, and M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints. In *Proceedings of the 21st International Workshop on Description Logics (DL2008)*, 2016.

[10] A. Calì, G. Gottlob, and T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics* 14 (2012) 57-83.

[11] A. Calì, G. Gottlob, and A. Pieris, Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence* 193 (2012) 87-128.

[12] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning* 39(2007) 385429.

[13] Y. Chen, F. Lin, Y. Zhang, and Y. Zhou, Loop-separable programs and their first-order definability. *Artificial Intelligence* 175 (2011) 809-913.

[14] C. Civili, *Processing Tuple-Generating Dependencies for Ontological Query Answering and Query Explanation*, PhD Thesis, 2015.

[15] C. Civili, and R. Rosati, A broad class of first-order rewritable tuple-generating dependencies. In *Proceedings of the 2nd International Conference on Datalog in Academia and Industry (Datalog-2012)*, pp 68-80, 2012.

[16] A. Deutsch, A. Nash, and J-B. Remmel, The chase revisited. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS-2008)*, pp 149-158, 2008.

[17] T. Eiter, T. Lukasiewicz, and L. Predoiu, Generalized consistent query answering under existential rules. In *Proceedings of KR-2016*, pp 359-368, 2016.

[18] R. Fagin, P-G. Kolaitis, R-J. Miller, and L. Popa, Data exchange: semantics and query answering. *Theoretical Computer Science* 336 (2005) 89-124.

[19] G. Gottlob, M. Manna, and A. Pieris, Combining decidability paradigms for existential rules. *Theory and Practice of Logic Programming* 16 (2013) 877-892.

[20] B-C. Grau, I. Horrocks, M. Krotzsch, C. Kupke, D. Magka, B. Motik, and Z. Wang, Acyclicity notions for existential rules and their application to query answering in ontologies. *Journal of Artificial Intelligence Research* 47 (2013) 741-808.

[21] M. Kaminski, Y. Nenov, and B-G. Grau, Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *Proceedings of KR-2014*, pp 76-91, 2014.

[22] R. Kontchakov, M. Rodriguez-Muro, Ontology-based data access with databases: A short course. In *Reasoning Web*, pp 194-229, 2013.

[23] M. Krötzsch, and S. Rudolph, Extending decidable existential rules by joining acyclicity and guardedness. In *Proceedings of IJCAI-2011*, pp 963-968, 2011.

[24] N. Leone, M. Manna, G. Terracina, and P. Veltri, Efficiently computable datalog$\exists$ programs. In *Proceedings of KR-2012*, 2012.

[25] F. Lin, and Y. Zhou, ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence* 157 (2004) 115-137.

[26] C. Nikolaou, E-V. Kostylev, G. Konstantinidis, M. Kaminski, B-C. Grau, and I. Horrocks, The Bag semantics of ontology-based data access. In *Proceedings of IJCAI-2017*, pp 1224-1230, 2017.

[27] C-H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.

[28] P-F. Petel-Schneider, and I. Horrocks, A comparison of two modelling paradigms in the semantic web. *Journal of Web Semantics* 5 (2007) 240-250.

[29] P. Senellart, and G. Gottlob, On the complexity of deriving schema mappings from database instances. In *Proceedings of PODS-2008*, 2008.

[30] Y. Zhang, and Y. Zhou, On the progression semantics and boundedness of answer set programs. In *Proceedings of KR-2010*, pp 518-527, 2010.

[31] Y. Zhou and Y. Zhang, A progression semantics for first-order logic programs. *Artificial Intelligence* 250 (2017) 58-79.

A Proofs of Theorems

A.1 Proof of Theorem 2

Theorem 2 *Let q be a BCQ over $\mathcal{R}$ and Σ a set of TGDs over $\mathcal{R}$. Then the following results hold:*

1. *There exists a* LOGSPACE *construction of an atomic BCQ q' and a set of TGDs Σ' of schema $\mathcal{R}' \supseteq \mathcal{R}$, where $|\mathsf{head}(\sigma')| = 1$ for each $\sigma' \in \Sigma'$, such that for all database D, $D \cup \Sigma \models q$ iff $D \cup \Sigma' \models q'$ [11].*

2. *If Σ' satisfies BDDP then Σ also satisfies BDDP.*

Proof. We use similar ideas to the construction as to that used in the proof of Lemma A.1 in [11]. Indeed, for the first step, we let q' be an atomic BCQ such that $\mathsf{body}(q') = r^*(X_1, \ldots, X_n)$, where r^* is a "fresh" predicate symbol and $\{X_1, \ldots, X_n\}$ coincides with $\mathsf{var}(q)$, i.e., assuming that q is the BCQ $\exists X_1, \ldots, X_n \, \widehat{\mathsf{body}(q)}$, where $\widehat{\mathsf{body}(q)}$ is the conjunction of the atoms in $\mathsf{body}(q)$, then q' is the atomic BCQ $\exists X_1, \ldots, X_n \, r^*(X_1, \ldots, X_n)$. Then we set $\Sigma^* = \Sigma \cup \big\{ r^*(X_1, \ldots, X_n) \leftarrow \widehat{\mathsf{body}(q)} \, \big\}$.

Then for the next step, we obtain Σ' from Σ^* by applying the following procedure: for each TGD $\sigma = \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Y} r_1(\mathbf{Y}), \ldots, r_k(\mathbf{Y}) \in \Sigma^*$, where $k > 1$ and $\mathbf{Y}$ the set of variables mentioned in $\mathsf{head}(\sigma)$, replace σ with the set of TGDs $\big\{\varphi(\mathbf{X}) \rightarrow r_\sigma(\mathbf{Y})\big\} \cup \big\{r_\sigma(\mathbf{Y}) \rightarrow r_1(\mathbf{Y})\big\} \cup \bigcup_{i \in \{1,\ldots,k-1\}} \big\{r_\sigma(\mathbf{Y}) \wedge r_i(\mathbf{Y}) \rightarrow r_{i+1}(\mathbf{Y})\big\}$, where r_σ is an $|\mathbf{Y}|$-ary new relation symbol in $\mathcal{R}'$. Then it follows from [11] that $D \cup \Sigma \models q$ iff $D \cup \Sigma' \models q'$. In addition, since it follows from the construction of Σ' that for each $i \geq 0$ there exists some $j \geq 0$ such that $\mathsf{chase}^{[i]}(D, \Sigma) \subseteq \mathsf{chase}^{[i+j]}(D, \Sigma')$, where j depends only on the size of the introduced auxiliary predicates (i.e., only depends on the size of Σ), then we further have that Σ' satisfies BDDP implies Σ also satisfies BDDP.

Here, we note that our transformation Σ' from Σ^* is slightly different from the one proposed in [11] in that we also add "$r_i(\mathbf{Y})$" in the body of rules "$r_\sigma(\mathbf{Y}) \rightarrow r_{i+1}(\mathbf{Y})$" (i.e., to get "$r_\sigma(\mathbf{Y}) \wedge r_i(\mathbf{Y}) \rightarrow r_{i+1}(\mathbf{Y})$") since this will allow us to retain information about the conjunction in the heads as they are considered in the loop pattern. It can be showed that the correctness of the transformation is preserved. □

A.2 Proof of Proposition 1

Proposition 1. *Let Σ be a set of TGDs. There exists a natural number N such*

that for every derivation path P of the form (5), if $|P| > N$ then there exist i, j ($1 \leq i < j \leq |P|$) such that $(\alpha_i, \rho_i) \sim (\alpha_j, \rho_j)$.

Proof. Let $K = \mathsf{max}\{\ |\mathbf{XYZ}| \mid \text{there exists } \text{“}\varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\psi(\mathbf{X}, \mathbf{Z})\text{”} \in \Sigma\ \} \cdot |\Sigma|$. Now let $\text{ARGPRM}(K)$ denote the set of $(5K \cdot |\mathsf{const}(\Sigma)| - 1)$-length permutations of the set

$$\{\overline{\mathbf{1}}^{c}, \overline{\mathbf{1}}^{V}, \overline{\mathbf{1}}^{\mathsf{n}}, |_1, \overline{\mathbf{2}}^{c}, \overline{\mathbf{2}}^{V}, \overline{\mathbf{2}}^{\mathsf{n}}, \ldots, |_{K-1}, \overline{\mathbf{K}}^{c}, \overline{\mathbf{K}}^{V}, \overline{\mathbf{K}}^{\mathsf{n}}\ \mid \\ c \in \mathsf{const}(\Sigma)\}.$$

Intuitively, the elements “$\overline{\mathbf{i}}^{x}$” ($1 \leq i \leq K$) where $x \in \{c, V, \mathsf{n} \mid c \in \mathsf{const}(\Sigma)\}$, are the argument positions of the tuple of constants, variables and nulls of a TGD in Σ. Here: (1) $x \in \mathsf{const}(\Sigma)$ denotes that the position i contains a constant; (2) $x = V$ denotes it contains a variable; and (3) $x = \mathsf{n}$ denotes it contains a labeled null. We say that “x” is the *type* of the argument $\overline{\mathbf{i}}^{x}$. The elements “$|_i$” act as a kind of “separator” such that if a tuple

$$\overline{\mathbf{i_1}}^{\mathsf{n}}\ |_{i_2}|_{i_3} \ldots |_{i_j}\ \overline{\mathbf{i_{j+1}}}^{V}\ \overline{\mathbf{i_{j+2}}}^{V}\ \overline{\mathbf{i_{j+3}}}^{V}\ |_{i_{j+4}} \ldots |_{i_{2k+1}}$$

is in $\text{ARGPRM}(K)$, then we view the consecutive series of arguments “$\overline{\mathbf{i_{j+1}}}^{V}\ \overline{\mathbf{i_{j+2}}}^{V}\ \overline{\mathbf{i_{j+3}}}^{V}$” as one group, and where we view arguments within a group as being “equal.” Then by $\text{PROPARGPRM}(K)$, denote the following set of tuples:

$$\Big\{\ \mathbf{e}\ \mid \text{there exists some } \mathbf{e}' \in \text{ARGPRM}(K) \text{ such that } \mathbf{e} \subseteq \mathbf{e}' \\ \text{and :} \\ \text{(1) } |\mathbf{e}| = 2K - 1; \qquad (8) \\ \text{(2) } \mathbf{e}[0] \text{ and } \mathbf{e}[|\mathbf{e}|] \text{ is not equal to “}|_k\text{”} \\ \quad (\text{ for } k \in \{1, \ldots, K-1\}); \qquad (9) \\ \text{(3) If } \mathbf{e}[i] = |_j\ (\text{ for } j \in \{1, \ldots, K-1\}) \text{ then :} \qquad (10) \\ \quad \text{(a) } i > 0 \text{ implies } \mathbf{e}[i-1] = \overline{\mathbf{k}}^{x} \\ \quad\quad (\text{for } k \in \{1, \ldots, K\} \text{ and } x \in \{c, V, \mathsf{n}\}); \\ \quad \text{(b) } i < K \text{ implies } \mathbf{e}[i+1] = \overline{\mathbf{k}}^{x} \\ \quad\quad (\text{for } k \in \{1, \ldots, K\} \text{ and } x \in \{c, V, \mathsf{n}\});$$

(4) If $\mathbf{e}[i] = \overline{\mathbf{j}}^{x}$ (for $j \in \{1, \ldots, K\}$ and $x \in \{c, V, \mathsf{n}\}$)
then : (11)
(a) $i > 0$ and $\mathbf{e}[i-1] = \overline{\mathbf{k}}^{y}$(for $k \in \{1, \ldots, K\}$
and $y \in \{c, V, \mathsf{n}\}$) implies $x = y$;
(b) $i < K$ and $\mathbf{e}[i+1] = \overline{\mathbf{k}}^{y}$(for $k \in \{1, \ldots, K\}$
and $y \in \{c, V, \mathsf{n}\}$) implies $x = y$;
(5) For each $i \in \{1, \ldots, K\}$, there exists some
$j \in \{1, \ldots, |\mathbf{e}|\}$ s.t. $\mathbf{e}[j] = \overline{\mathbf{i}}^{x}$ and $x \in \{c, V, \mathsf{n}\}$ $\}$. (12)

Intuitively the *proper argument permutation tuples*, as denoted "PROPARGPRM(K), " captures the intended meaning of equivalence $\mathbf{t}_1 \sim \mathbf{t}_2$ assuming that $\mathbf{t}_1 \cap \mathbf{t}_2 = \emptyset$. Indeed, we have that (8) specifies that no arguments are repeated on different groups; (9) specifies that the ends of the tuple are not delimited by "$|_k$"; (10) specifies that only one separator (i.e., the "$|_i$" element) acts for each group; (11) specifies that each group are of the same types; and lastly, (12) specifies that each position $i \in \{1, \ldots, K\}$ is mentioned in at least some group. Clearly, we have that $|\text{PROPARGPRM}(K)| \leq |\text{ARGPRM}(K)| \leq (5K \cdot |\mathsf{const}(\Sigma)| - 1)!$.

With a slight abuse of notation, given some element $\mathbf{e} \in \text{PROPARGPRM}(K)$ and some K-length tuple $\mathbf{t}$, we say that $\mathbf{t}$ is in the equivalence class of $\mathbf{e}$, denoted $\mathbf{t} \sim \mathbf{e}$, if for each $i, j \in \{1, \ldots, K\}$, we have that $\mathbf{t}[i] = \mathbf{t}[j]$ iff $\overline{\mathbf{i}}^{x}$ and $\overline{\mathbf{j}}^{y}$ belongs to the same group in $\mathbf{e}$. Thus, to extend to the case where $\mathbf{t}_1 \cap \mathbf{t}_2 \neq \emptyset$, we define the mapping $f : \text{PROPARGPRM}(K) \longrightarrow \mathbb{N}$ such that for each $\mathbf{e} \in \text{PROPARGPRM}(K)$, $\iota(\mathbf{e})$ denotes the size of the following set:

$$S_{\mathbf{e}} = \big\{(\mathbf{t}_1, \mathbf{t}_2) \mid \mathbf{t}_1, \mathbf{t}_2 \in T^K, \mathbf{t}_1 \sim \mathbf{e}, \mathbf{t}_2 \sim \mathbf{e}, \mathbf{t}_1 \cap \mathbf{t}_2 \neq \emptyset \text{ and } \mathbf{t}_1 \not\sim \mathbf{t}_2\big\} \quad (13)$$

(i.e., $f(\mathbf{e}) = |S_{\mathbf{e}}|$), and where T is the following set of distinct constants, variables and labeled nulls: $\text{CONST}(\Sigma) \cup \{X_1, \ldots, X_{2K}\} \cup \{\mathsf{n}_1, \ldots, \mathsf{n}_{2K}\}$. Clearly, we have that $|S_{\mathbf{e}}|$ is clearly defined since $S_{\mathbf{e}}$ is finite for each $\mathbf{e} \in \text{PROPARGPRM}(K)$. Then finally, we define $N = \sum_{\mathbf{e} \in \text{PROPARGPRM}(K)} f(\mathbf{e})$.

Now on the contrary, assume that P is a derivation path (α_1, ρ_1), ..., (α_N, ρ_N), $(\alpha_{N+1}, \rho_{N+1})$, ..., $(\alpha_{N+k}, \rho_{N+k})$ such that $k > 0$ (i.e., $|P| > N$) and $(\alpha_i, \rho_i) \not\sim (\alpha_j, \rho_j)$ for $1 \leq i < j \leq N + k$. Now consider $(\alpha_{N+i}, \rho_{N+i})$ for some $i \in \{1, \ldots, k\}$. Then since $N > |\text{PROPARGPRM}(K)|$, we have that for some $\mathbf{e} \in |\text{PROPARGPRM}(K)|$ and $j \in \{1, \ldots, N\}$, $(\alpha_j, \rho_j) \sim_{\mathbf{e}} (\alpha_{N+i}, \rho_{N+i})$, where assuming that $\rho_j = \sigma[\mathbf{XYZ}/\mathbf{T_1 T_2 T_3}]$ and $\rho_{N+i} = \sigma[\mathbf{XYZ}/\mathbf{T'_1 T'_2 T'_3}]$

(for some $\sigma \in \Sigma$), $(\alpha_j, \rho_j) \sim_{\mathbf{e}} (\alpha_{N+i}, \rho_{N+i})$ denotes that $\mathbf{T_1 T_2 T_3} \sim \mathbf{e}$ and $\mathbf{T'_1 T'_2 T'_3} \sim \mathbf{e}$. (Note that by the assumption that $(\alpha_j, \rho_j) \not\sim (\alpha_{N+i}, \rho_{N+i})$, we have that $\mathbf{T_1 T_2 T_3} \not\sim \mathbf{T'_1 T'_2 T'_3}$ as well.) Now there can only be one of the two possibilities, either (1) $\mathbf{T_1 T_2 T_3} \cap \mathbf{T'_1 T'_2 T'_3} = \emptyset$, or (2) $\mathbf{T_1 T_2 T_3} \cap \mathbf{T'_1 T'_2 T'_3} \neq \emptyset$. If we assume the first case (1), then it contradicts the assumption that $\mathbf{T_1 T_2 T_3} \not\sim \mathbf{T'_1 T'_2 T'_3}$ because $(\alpha_j, \rho_j) \sim_{\mathbf{e}} (\alpha_{N+i}, \rho_{N+i})$. On the other hand, if we consider the latter case (2), then since we have that $N = \sum_{\mathbf{e} \in \text{PROPARGPRM}(K)} f(\mathbf{e})$ with $f(\mathbf{e})$ the size of the finite set (13), then this will be a contradiction as well. □

A.3 Proof of Theorem 3

Theorem 3. *Let Σ be a set of TGDs, D a database over schema $\mathcal{R}$, and q a BCQ query $\exists \mathbf{Z} p(\mathbf{Z})$. Then $\mathsf{chase}(D, \Sigma) \models q$ iff there exist an instantiation $T(D, \Sigma)$ for some derivation tree $T(\Sigma)$ and a substitution θ, such that $T(D, \Sigma) \models p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of terms from Γ of the same length as $\mathbf{Z}$, and $\mathbf{t}\theta = \mathbf{Z}$.*

Proof. ("$\Longrightarrow$") We prove this direction by first providing the following lemma. Firstly, for the below proof, we just assume for convenience and readability that $\mathsf{chase}^{[N]}(D, \Sigma)$ is parameterized by some sequence ζ, i.e., "$\mathsf{chase}^{[N]}(D, \Sigma)$" to mean "$\mathsf{chase}^{[N]}_{\zeta}(D, \Sigma)$" (please see Section 2).

Lemma 1. *Given an instantiated derivation tree $T(D, \Sigma) = (N, E, \lambda)$ with nodes N, edges E and labeling function λ, of Σ under a database D, there exists a homomorphism $\mu : \mathsf{nodes}(T(D, \Sigma)) \longrightarrow \mathsf{chase}^{[N]}(D, \Sigma)$, where $N \leq |\mathsf{nodes}(T(D, \Sigma))| - |\mathsf{leafNodes}(T(D, \Sigma))|$, such that the following conditions are satisfied:*

$$\begin{aligned} &\textit{1. For each } v \in \mathsf{leafNodes}(T(D, \Sigma)) \textit{ such that } \lambda(v) = (\alpha, \alpha), \\ &\mu(v) = \alpha \in D; \end{aligned} \tag{14}$$

$$\begin{aligned} &\textit{2. For each } v \in \mathsf{nodes}(T(D, \Sigma)) \textit{ such that } \lambda(v) = (\alpha, \rho), \\ &\mathsf{child}(v) = \{v_1, \ldots, v_1\},\ \rho = \sigma\theta \textit{ for some substitution } \theta, \\ &\textit{and } \sigma = \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} p(\mathbf{X}, \mathbf{Z}) \in \Sigma, \textit{ there exists a} \\ &\textit{homorphism } h \textit{ such that } h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq \{\mu(v_1), \ldots, \mu(v_n)\} \\ &\textit{and extension } h' \textit{ of } h\restriction_{\mathbf{X}} \textit{ where } \mu(v) = h'(p(\mathbf{X}, \mathbf{Z})); \end{aligned} \tag{15}$$

3. For each $v \in \mathsf{nodes}(T(D,\Sigma))$ *such that* $\lambda(v) = (\alpha, \rho)$ *and* $\alpha = p(\mathbf{t}),$ *if* $\mu(v) = q(\mathbf{t}')$ *then we have that* $p(\mathbf{t})\theta = q(\mathbf{t}')$ *for some substitution* θ. (16)

4. For each $v_1, v_2 \in \mathsf{nodes}(T(D,\Sigma))$ *such that* $\lambda(v_1) = \lambda(v_2)$ *(i.e.,* v_1 *and* v_2 *have the same label),* *then we also have that* $\mu(v_1) = \mu(v_2)$. (17)

Proof. We show the existence of such a homomorphism μ by induction on the depth of the tree $T(D,\Sigma)$ starting from the leaf nodes (i.e., the nodes labeld by the database facts) going up to the root node labeled by (α, ρ). So towards this purpose, for $i \in \{1, \ldots, \mathsf{depth}(T(D,\Sigma))\}$, denote by $T^i(D,\Sigma)$ as the *forest* made up of the subtrees T' of $T(D,\Sigma)$ that are rooted on some node labeled $(\alpha', \rho') \in \mathsf{nodes}(T(D,\Sigma))$ such that $\mathsf{depth}(T') = i$. In particular, we note that $T^i(D,\Sigma)$ will be exactly $T(D,\Sigma)$ when $i = \mathsf{depth}(T(D,\Sigma))$. Lastly, for some node $v \in \mathsf{nodes}(T(D,\Sigma))$, denote by T_v as the subtree of $T(D,\Sigma)$ that is rooted in v.

Basis: When $i = 1$, then each nodes $v \in \in \mathsf{nodes}(T^1(D,\Sigma))$ labeled with (α, ρ) are such that $\rho = \alpha$ and $\alpha \in D$, i.e., α is database fact. Therefore, we simply define $\mu : \mathsf{nodes}(T^1(D,\Sigma)) \longrightarrow \mathsf{chase}(D,\Sigma)$ by setting $\mu(v) = \alpha \in D \subseteq \mathsf{chase}(D,\Sigma)$, for each $v \in \mathsf{nodes}(T^1(D,\Sigma))$. In particular, we note that Conditions (14)-(17) above are already satisfied.

Inductive step: Assume there exists a homomorphism $\mu : \mathsf{nodes}(T^k(D,\Sigma)) \longrightarrow \mathsf{chase}^{[N]}(D,\Sigma)$, for some $k \geq 1$ and $N \leq |\mathsf{nodes}(T^k(D,\Sigma)) \setminus \mathsf{leafNodes}(T^k(D,\Sigma))|$, that satisfies Conditions (14), (15), (16) and (17) above.

Now consider a node $v \in \mathsf{nodes}(T^{k+1}(D,\Sigma)) \setminus \mathsf{nodes}(T^k(D,\Sigma))$ such that $\lambda(v) = (\alpha, \rho)$, $\mathsf{body}(\rho) = \{\alpha_1, \ldots, \alpha_n\}$, $\rho = \sigma\theta$ and $\sigma = \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z p(\mathbf{X}, \mathbf{Z})$. Then by the definition of the instantiated derivation tree $T(D,\Sigma)$, assume that $\mathsf{child}(v) = \{v_1, \ldots, v_n\}$ such that $\lambda(v_1) = (\alpha_1, \rho_1), \ldots, \lambda(v_n) = (\alpha_n, \rho_n)$. Then further assuming that $\alpha_1 = p_1(\mathbf{t}_1), \ldots, \alpha_n = p_n(\mathbf{t}_n)$ and $\mu(v_1) = q_1(\mathbf{t'}_1), \ldots, \mu(v_n) = q_n(\mathbf{t'}_n)$, $p_i(\mathbf{t}_i)\theta_i = q_i(\mathbf{t'}_i)$ for some substitution θ_i (ind. hyp.), then we have that $\{\mu(v_1), \ldots, \mu(v_n)\} \subseteq \mathsf{chase}^{[N]}(D,\Sigma)$ (ind. hyp.). Therefore, from the

fact that homomorphism μ satisfies Condition (17) above, then it follows that we can define a homomorphism τ by setting $\tau = \theta_1 \cup \ldots \cup \theta_n$, such that $\tau(\mathsf{body}(\rho)) \subseteq \{\mu(v_1), \ldots, \mu(v_n)\} \subseteq \mathsf{chase}^{[N]}(D, \Sigma)$. In fact, because $\rho = \sigma\theta$, then we can "directly" define a homomorphism h for σ by setting $h = \tau \circ \theta$ so that $h(\mathsf{body}(\sigma)) \subseteq \{\mu(v_1), \ldots, \mu(v_n)\} \subseteq \mathsf{chase}^{[N]}(D, \Sigma)$. Then from the definition of $\mathsf{chase}^{[N]}(D, \Sigma)$, it follows that σ will be applicable to $\mathsf{chase}^{[N]}(D, \Sigma)$ under the homomorphism h, i.e., there exists some chase step $I_i \xrightarrow{\sigma,h} I_{i+1}$ such that $\{\lambda(v_1), \ldots, \lambda(v_n)\} \subseteq I_i$, for some $0 \leq i \leq N$. Then based on this fact, there will exists some $q(\mathbf{t}') \in \mathsf{chase}^{[N+1]}(D, \Sigma)$ such that for some extension h' of $h \restriction_{\mathbf{X}}$, we have that $q(\mathbf{t}') = h'(\mathsf{head}(\sigma))$. Therefore, we can define μ for the node $v \in \mathsf{nodes}(T^{k+1}(D, \Sigma)) \setminus \mathsf{nodes}(T^k(D, \Sigma))$ by setting $\mu(v) = q(\mathbf{t}')$. In particular, assuming that $\alpha = p(\mathbf{t})$ (i.e., recall that $\lambda(v) = (\alpha, \rho)$), then we note from the definition of the extension h' of $h \restriction_{\mathbf{X}}$ that $p(\mathbf{t})\theta = q(\mathbf{t}')$ for some substitution θ. Therefore, it follows that μ is a homomorphism that can be extended from $\mathsf{nodes}(T^{k+1}(D, \Sigma))$ to $\mathsf{chase}^{[N+M]}(D, \Sigma)$, where $M = |\mathsf{nodes}(T^{k+1}(D, \Sigma)) \setminus \mathsf{nodes}(T^k(D, \Sigma))| - |\mathsf{leafNodes}(T(D, \Sigma))|$.

□

Then from Lemma 1, since $T(D, \Sigma) \models p(\mathbf{t})$, then assuming that $\lambda\big(\mathsf{root}(T(D, \Sigma))\big) = (\alpha, \rho)$ such that $\alpha = r(\mathbf{s})$, we have from the definition of "instantiated tree supportedness" of an atom that $h(r(\mathbf{t})) = r(\mathbf{s})$ for some homomorphism $h : \mathbf{t} \longrightarrow \mathbf{s}$. Then because we have that $\mu(r(\mathbf{s})) = r(\mathbf{t}')$ for some atom $r(\mathbf{t}') \in \mathsf{chase}^{[N]}(D, \Sigma)$, where $N \leq |\mathsf{nodes}(T(D, \Sigma))| - |\mathsf{leafNodes}(T(D, \Sigma))|$ and $\mu : \mathsf{nodes}(T(D, \Sigma)) \longrightarrow \mathsf{chase}(D, \Sigma)$ the "bounding number" and homomorphism defined in Lemma 1, respectively, then we also have from Lemma 1 that $r(\mathbf{s})\theta = r(\mathbf{t}')$ for some substitution θ. Therefore, with $h' = \theta \circ h$, then we have that $h'(p(\mathbf{t})) = q(\mathbf{t}') \in \mathsf{chase}^{[N]}(D, \Sigma)$, which implies that $\mathsf{chase}^{[N]}(D, \Sigma) \models p(\mathbf{t})$.

("$\Longleftarrow$") Assume $\mathsf{chase}^{[N]}(D, \Sigma) \models p(\mathbf{t})$ for some atom $p(\mathbf{t})$ and $N \geq 1$. Then by the definition of $\mathsf{chase}^{[N]}(D, \Sigma) \models p(\mathbf{t})$, there exists some atom $p(\mathbf{t}')$ $\in \mathsf{chase}^{[N]}(D, \Sigma)$ and homomorphism $h : \mathbf{t}' \longrightarrow \mathbf{t}$ such that $h(p(\mathbf{t})) = p(\mathbf{t}')$. Thus, there exists some finite chase sequence $I_0 \xrightarrow{\sigma_0,h_0} I_1, \ldots, I_{N-1} \xrightarrow{\sigma_N,h_N} I_N$ such that $p(\mathbf{t}') \in I_N$. Let us assume without loss of generality that for $i \in \{1, \ldots, N-1\}$, there does not exists another atom $p(\mathbf{t}'') \in I_i$ such that $h(p(\mathbf{t}))$ $= p(\mathbf{t}'')$. Then based on the sequences of TGDs σ_i and homomorphisms h_i that made σ_i applicable to I_i, we can construct an instantiated derivation tree

$T(D, \Sigma)$ as follows:

1. Let $\mathsf{root}(T(D, \Sigma))$ be labeled with $(p(\mathbf{t}'), \sigma_N \theta_N)$, where θ_N is the corresponding substitution for h_N and its extension h'_N;

2. For each atom $\alpha \in \mathsf{chase}^{[N]}(D, \Sigma)$ either :
- add a node v with label (α, α), if $\alpha \in D$, otherwise
- add a node v with label (α, ρ), where $\alpha = \mathsf{head}(\rho)$, $\rho = \sigma_i \theta_i$ and θ_i the corresponding substitution for h_i and its "extension" h'_i.

3. For each node v with label (α, ρ) such that $\rho = \sigma\theta$, for some $\sigma \in \Sigma$ and substitution θ, and $\mathsf{body}(\rho) = \{\alpha_1, \ldots, \alpha_n\}$ then for $i \in \{1, \ldots, n\}$, add an edge (v, v_i) such that either:

- v_i is labled with (α_i, α_i), if $\alpha_i \in D$, otherwise
- v_i is labled with (α_i, ρ_i), such that $\alpha_i = \mathsf{head}(\rho_i)$, $\rho_i = \sigma_j \theta_j$, θ_j the corresponding subtitution for h_j (and corresponding extension h'_j) and $I_j \xrightarrow{\sigma_j, h_j} I_{j+1}$ is the first chase step that derived α_i.

Then it is not too difficult to see that the above construction for $T(D, \Sigma)$ is in fact an instantiated derivation tree and where $N \leq |\mathsf{nodes}(T(D, \Sigma))| - |\mathsf{leafNodes}(T(D, \Sigma))|$. (i.e., recall that $p(\mathbf{t}') \in I_N$ such that $I_{N-1} \xrightarrow{\sigma_N, h_N} I_N$ is the first chase step that derived $p(\mathbf{t}')$). Therefore, because $h(p(\mathbf{t})) = p(\mathbf{t}')$ for some homomorphism $h : \mathbf{t} \longrightarrow \mathbf{t}'$ (i.e., recall that $\mathsf{chase}(D, \Sigma) \models p(\mathbf{t})$ and $p(\mathbf{t}') \in \mathsf{chase}$ such that $h(p(\mathbf{t})) = p(\mathbf{t}')$) and since, assuming that $(\alpha, \rho) = \mathsf{root}(T(D, \Sigma))$, we have that $p(\mathbf{t}') = \alpha$ from the construction of $T(D, \Sigma)$, then we clearly have that $T(D, \Sigma) \models p(\mathbf{t})$ through the same "witnessing" homomorphism h. □

A.4 Proof of Proposition 2

Proposition 2. *Given a finite set Σ of TGDs, Σ only has a finite number of loop patterns under the equivalence relation $\sim$.*

Proof. From Proposition 1, there is a number N such that for any derivation path P of the form (5), for each $N+1 \leq j \leq |P|$, there exists some $1 \leq i \leq N$ such that $(\alpha_i, \rho_i) \sim (\alpha_j, \rho_j)$. Therefore, it follows that one only has to check each derivation path P if it is a loop pattern. □

A.5 Proof of Theorem 4

Theorem 4. *The class of loop restricted TGDs satisfies the BDTDP.*

Before proving Theorem 4, we first introduce the notion of subsumption between two derivation trees.

Definition 11 (Derivation tree subsumption). *Let Σ be a set of TGDs, and $T_1(\Sigma)$ and $T_2(\Sigma)$ be two derivation trees of Σ. Then we say that $T_2(\Sigma)$* subsumes *$T_1(\Sigma)$ if the following conditions are satisfied: (1) $\mathsf{root}(T_2(\Sigma)) = \mathsf{root}(T_1(\Sigma))$; and (2) $\mathsf{leafNodes}(T_2(\Sigma)) \subset \mathsf{leafNodes}(T_1(\Sigma))$.*

Proof. Given a set Σ of LR TGDs. Let $\mathcal{T}(\Sigma)$ be the set of all derivation trees of Σ. We consider the set $\mathbb{T}(\Sigma)$ of all derivation trees that are distinct under $\sim$ and their tree depths are not larger than N, where N is the integer mentioned in Proposition 1[6]. Then it is clear that $\mathbb{T}(\Sigma) \subseteq \mathcal{T}(\Sigma)$ and is a finite set. Now we can prove the following important result:

Lemma 2. *Let $T(\Sigma) \in \mathcal{T}(\Sigma)$ (note Σ is LR). Then for every database D and every atom $p(\mathbf{t})$, $T(D, \Sigma) \models p(\mathbf{t})$ iff there exists some $T'(\Sigma) \in \mathbb{T}(\Sigma)$ such that $T'(D, \Sigma) \models p(\mathbf{t})$.*

Then the theorem follows directly from Lemma 2, by setting the bound to be the maximal depth of trees in $\mathbb{T}(\Sigma)$. The key idea of proving Lemma 2 is based on the fact that for any tree $T(\Sigma)$ in $\mathcal{T}(\Sigma)$, there is a corresponding tree $T'(\Sigma)$ in $\mathbb{T}(\Sigma)$ which can replace $T(\Sigma)$ without affecting $T(\Sigma)$'s derivations. Without loss of generality, consider a tree $T(\Sigma)$ in $\mathcal{T}(\Sigma)$, where a path P in $T(\Sigma)$ is longer than N. Then from Proposition 1, there must exist a loop pattern $L = (w_i, \cdots, w_j)$ in path P, such that the depth of node w_i is within the

[6] A complete proof of Proposition 1 is given in the full version of this paper, in which N is presented.

bound N, and the depth of node w_j is beyond N. Since $w_i \sim w_j$ and L is loop restricted and from the conditions presented in Definition 8, then using similar ideas from [13], we can prove that the subtree underneath the node $body_b(\rho_i)$ in $T(\Sigma)$ can be replaced by the subtree underneath the node $body_b(\rho_j)$. That is, the loop pattern fragment $(w_i, \cdots, w_j)$ in path P is replaced by a new node $w_i^* : (\alpha_i, [body_b(\rho_j), body_h(\rho_i) \rightarrow \alpha_i])$. According to Proposition 2, Σ only has a finite number of loop patterns under $\sim$. So by doing this *folding* for all paths in $T(\Sigma)$, we eventually transform $T(\Sigma)$ into a $T'(\Sigma)$ whose depth is bounded by N, that is, $T'(\Sigma) \in \mathbb{T}(\Sigma)$. □

A.6 Proof of Theorem 5

Theorem 5. *The class of LR TGDs satisfies BDDP.*

Proof. If a set of TGDs Σ satisfies the BDTDP property, then there exists some number K such that for every database D, we have that $D \cup \Sigma \models \exists \mathbf{Z} p(\mathbf{Z})$ iff there exists some atom $p(\mathbf{n})$, where $\mathbf{n} \in \Gamma_N^{|\mathbf{n}|}$, such that $T(D, \Sigma) \models p(\mathbf{n})$ and where $\mathsf{depth}(T(D, \Sigma)) \leq K$. Then since by Theorem 3, we have that $\mathsf{chase}^{[N]}(D, \Sigma) \models p(\mathbf{n})$, where N is bounded by $|\mathsf{nodes}(T(D, \Sigma))|$, then it follows that Σ also satisfies the BDDP property. □

A.7 Proof of Theorem 6

Theorem 6. *Consider the BCQA problem for a given set of LR TDGs. Its data complexity is in* AC^0*, and its combined complexity is* 2-EXPTIME*-complete.*

Proof. We only prove here the 2-EXPTIME-complete combined complexity since the AC^0 data complexity directly follows from Theorem 5 and from the fact that first-order rewritable implies AC^0 in data complexity [11].

(*Membership*) Let Σ be a LR set of TGDs. Then we have by Theorem 4 that Σ satisfies BDTDP. That is, there exists a number K such that for any database D and query $q = \exists \mathbf{Z} p(\mathbf{Z})$, we have that there exists a derivation tree $T(\Sigma) = (N, E, \lambda)$ such that $T(\Sigma) \models p(\mathbf{t})$ and where:

- $\mathbf{t}$ is a tuple of terms from Γ of the same length as $\mathbf{Z}$;
- $\mathbf{t}\theta = \mathbf{Z}$;
- $\mathsf{depth}(T(D, \Sigma)) \leq K$.

Then enumerating through all those derivation trees can be done in time $O(\mathsf{maxbody}(\Sigma, q)^K)$, where:

$$\mathsf{maxbody}(\Sigma, q) \;=\; \mathsf{max}\Big\{\,\big|\mathsf{body}(\sigma)\big|\;\big|\;\sigma \in \Sigma\,\Big\}.$$

On the other hand, since the number K is sufficient to be of size

$$O(2^{p(|\Sigma|\cdot \mathsf{maxart}(\Sigma,q))}),$$

for some polynomial function $p(x)$ where $\mathsf{maxart}(\Sigma, q)$ denotes the maximum arity of an atom in $\mathsf{atoms}(\Sigma \cup \{q\})$ (by Proposition 2 and Theorem 4), then it follows that enumerating though all those utmost K-depth derivation trees can be done in time

$$\begin{aligned} & O(\mathsf{maxbody}(\Sigma, q)^{2^{p(|\Sigma|\cdot \mathsf{maxart}(\Sigma,q))}}) \\ \leq\; & O(2^{\lceil \log(\mathsf{maxbody}(\Sigma,q)) \rceil \cdot 2^{p(|\Sigma|\cdot \mathsf{maxart}(\Sigma,q))}}). \end{aligned}$$

Then finally, further considering the checking if any of these utmost K-depth derivation trees can be instantiated under the database D can be done in time:

$$\begin{aligned} & O(|D| \cdot 2^{\lceil \log(\mathsf{maxbody}(\Sigma,q)) \rceil \cdot 2^{p(|\Sigma|\cdot \mathsf{maxart}(\Sigma,q))}}) \\ \leq\; & O(2^{\lceil \log(|D|) \rceil + \lceil \log(\mathsf{maxbody}(\Sigma,q)) \rceil \cdot 2^{p(|\Sigma|\cdot \mathsf{maxart}(\Sigma,q))}}). \end{aligned}$$

(*Hardness*) We reduce the CQA problem under *weak-acyclic* (WA) TGDs into the CQA under LR TGDs. Here, we note that CQA under WA TGDs known to be 2-ExpTime-complete [11]. So towards this purpose, for a given database D, set of LR TGDs Σ, and BCQ q, we denote the transformation $\text{LR}_{D,\Sigma}(\Sigma)$ such that $\text{LR}_{D,\Sigma}(\Sigma)$ is the following set of TGDs:

$$\Big\{\; p_1^*(\mathbf{t}_1, \mathbf{Y}_1) \wedge \ldots \wedge p_n^*(\mathbf{t}_n, \mathbf{Y}_n) \rightarrow q^*(\mathbf{u}, \mathbf{Z})$$

$$\Big|\; \text{“}p_1(\mathbf{t}_1) \wedge \ldots \wedge p_n(\mathbf{t}_n) \rightarrow q(\mathbf{u})\text{”} \in \Sigma \cup \{q\} \text{ and there exists a}$$

number l such that :

(1) for each $\mathbf{X} \in \{\mathbf{Y}_1, \ldots, \mathbf{Y}_n, \mathbf{Z}\}$:

(a) $|\mathbf{X}| \;=\; \text{MAXART}(\Sigma) \cdot \Big\lceil \log\big(\,\big|\text{CONST}(D)\big| + \big|\text{ARG}(\Sigma)\big|\,\big)\Big\rceil$;

(2) $\mathbf{Z} \;=\; X_1, \ldots, X_l, 1, X_{l+1}, \ldots, X_{|\mathbf{Z}|}$ and $\mathbf{Y}_i, \mathbf{Y}_j \in \{\mathbf{Y}_1, \ldots, \mathbf{Y}_n\}$:

(a) $\mathbf{Y}_i \;=\; X_1, \ldots, X_l, 0, X_{l+1}, \ldots, X_{|\mathbf{Y}_i|}$;

(b) $\mathbf{Y}_j \;=\; X_1, \ldots, X_l, 0, X_{l+1}, \ldots, X_{|\mathbf{Y}_j|}\;\Big\}$,

and $\text{LR}_{D,\Sigma}(D)$ is the following set of atoms:

$$\text{LR}_{D,\Sigma}(D) \;=\; \Big\{\, p^*(\mathbf{t},\mathbf{0}) \,\big|\, p(\mathbf{t}) \;\in\; D \,\Big\},$$

where $|\mathbf{0}| = |0,\ldots,0| = \text{MAXART}(\Sigma) \cdot \Big\lceil \log\big(\,\big|\text{CONST}(D)\big| + \big|\text{ARG}(\Sigma)\big|\,\big)\Big\rceil$. In particular, we note that for each predicate symbol $p \in \mathsf{Pred}(\Sigma)$, we have that p^* is a new (auxiliary) predicate symbol. Then it follows that $\text{LR}_{D,\Sigma}(\Sigma)$ is LR TGDs. Moreover, we have that:

$$\mathsf{chase}(D,\Sigma) \;\models\; q \;\;\text{iff}\;\; \mathsf{chase}\big(\text{LR}_{D,\Sigma}(D), \text{LR}_{D,\Sigma}(\Sigma)\big) \;\models\; \text{LR}_{D,\Sigma}(q).$$

□

A.8 Proof of Theorem 7

Theorem 7. *Deciding whether a set of TGDs is loop restricted is* PSPACE*-complete.*

Proof. (*Membership*) To show membership, we provide a non-deterministic PSPACE algorithm called NOTLR, as we will described in Algorithm 1. Therefore, since the complexity class PSPACE is closed under nondeterminism and complementation (i.e., PSPACE, NPSPACE and coPSPACE and are all equivalent complexity classes) then the result follows. Firstly, for the following Algorithm 1, we assume without loss of generality that all the existentially quantified variables in each head of a TGD in Σ has already been eliminated via appropriate substitutions of labeled nulls from Γ_N. Thus, we further assume that $\sigma_1, \sigma_2 \in \Sigma$ and $\sigma_1 \neq \sigma_2$ implies $\mathsf{varNull}(\sigma_1) \cap \mathsf{varNull}(\sigma_2) = \emptyset$, i.e., all the universally quantified variables and labeled nulls in each pair of unique TGDs are pairwise disjoint.

Now we further define some necessary notions. Given an atom α, a TGD σ such that $\mathsf{varNull}(\alpha) \cap \mathsf{varNull}(\sigma) = \emptyset$ and some atom $\beta \in \mathsf{body}(\sigma)$ (and thus, $\mathsf{varNull}(\alpha) \cap \mathsf{varNull}(\beta) = \emptyset$ as well), we denote by $\theta_{\langle\beta/\alpha,\sigma\rangle}$ (as will be used in Line 19 of Algorithm 1) as a substitution on σ that satisfies the following properties:

- $\beta\theta_{\langle\beta/\alpha,\sigma\rangle} = \alpha$;
- for each $t \in (\mathsf{varNull}(\sigma) \setminus \mathsf{varNull}(\beta))$, we have that $\theta_{\langle\beta/\alpha,\sigma\rangle}(t) = t$, i.e., identity on terms not in $\mathsf{varNull}(\beta)$.

Intuitively, given that $\mathsf{varNull}(\alpha) \cap \mathsf{varNull}(\sigma) = \emptyset$, we have that $\theta_{\langle \beta/\alpha,\sigma\rangle}$ is the "minimal substitution" that is required to unify the atom α into the TGD σ through the atom $\beta \in \mathsf{body}(\sigma)$. Given again a TGD σ and two finite sets $V_1, V_2 \subset (\Gamma_V \cup \Gamma_N)$, we denote by $\vartheta_{[V_1 \mapsto V_2]}$ (which will be used in Line 31 of Algorithm 1) as a renaming function $\vartheta_{[V_1 \mapsto V_2]} : \mathsf{varNull}(\sigma) \longrightarrow V_2$ such that the following properties are satisfied:

- $t \in (V_1 \cap \mathsf{varNull}(\sigma))$ implies $\vartheta_{[V_1 \mapsto V_2]}(t) \in V_2$;
- $t_1, t_2 \in \mathsf{varNull}(\sigma)$ implies $t_1 = t_2$ iff $\vartheta_{[V_1 \mapsto V_2]}(t_1) = \vartheta_{[V_1 \mapsto V_2]}(t_2)$;
- $t \in (\mathsf{varNull}(\sigma) \setminus V_1)$ or $t \in (V_1 \cap V_2)$ implies $\vartheta_{[V_1 \mapsto V_2]}(t) = t$.

Intuitively, $\vartheta_{[V_1 \mapsto V_2]}$ is an injective renaming function from the variables and labeled nulls of σ mentioned in V_1 into those in V_2. Lastly, we also assume the two finite sets S_V and S_N, where: (1) $S_V \subset \Gamma_V$ and $S_N \subset \Gamma_N$; (2) $|S_V| = |S_N| = \mathsf{maxArt}(\Sigma)$; and (4) $(S_V \cup S_N) \cap \mathsf{varNull}(\Sigma) = \emptyset$.

In Algorithm 1, we have that Line 1 nondeterministically guesses a triple $(\alpha_0, \sigma_0, \theta_0)$ and two finite subsets $V \subset \Gamma_V$ and $V^* \subset (\Gamma_V \cup \Gamma_N)$ and where $V \subseteq V^*$. Here, $\sigma_0\theta_0$, as obtained from the tuple $(\alpha_0, \sigma_0, \theta_0)$, denotes the "initial" part of our loop pattern, i.e., informally, if we assume a loop pattern $P = (\alpha_1, \rho_1), \ldots, (\alpha_n, \rho_n)$ then ρ_n would correspond to our initial $\sigma_0\theta_0$. The (boolean) variables "lp, " and "lp_II, " as first mentioned in Lines 10-11, lets us know if a derivation path (as will be produce via the **while** loop in Lines 14-32, which we will explain later) is a *loop pattern* or *loop separable pattern*, respectively. The following Lines 12 and 13 is for the initialization of the **while** loop of Lines 14-32. In particular, the number "n" in Line 13 is nondeterministically a guess of a length of a possible loop pattern that will be derived in the **while** loop and where the value of N is as specified in Proposition 1. Then finally in regards to the **while** loop (Lines 14-32), Line 16 nondeterministically guesses a triple $(\alpha', \sigma', \theta')$ that satisfies the three Conditions (1), (2) and (3) as specified in Lines 17, 18 and 19, respectively. Intuitively, each iteration of the **while** loop extends our initial (one element) derivation path $(\alpha_N, \sigma_N\theta_N)$ (corresponding to the initial triple "$(\alpha_0, \sigma_0, \theta_0)$" as mentioned earlier) to add one more element corresponding to the triple $(\alpha', \sigma', \theta')$ with properties as mentioned in Lines 17-19. Indeed, we note in particular the condition in Line 18 which guarantees that $\alpha \in \mathsf{body}(\sigma'\theta')$ so that each added new (guessed) elements $(\alpha', \sigma'\theta')$ into the derivation path is indeed still a "derivation path." Thus, after (say) k iterations, this would correspond to a derivation path (say) $P' = (\alpha_j, \rho_j), \ldots, (\alpha_N, \rho_N)$ such that $|P'| = k$ and where $\rho_N = \sigma_0\theta_0$.

Algorithm 1: NOTLR

Data: A set of TGDs Σ (*with existentially quantified variables already eliminated as mentioned above*), number N from Proposition 1 and the two sets S_V and S_N previously mentioned above.

Result: "$accept$" if not loop restricted and "$reject$" otherwise

let $(\alpha_0, \sigma_0, \theta_0)$ be a triple and V and V^* be two finite sets such that:
  (1) $\sigma_0 \in \Sigma$;
  (2) θ_0 an applicable substitution for σ_0 that satisfies the following conditions:
    (a) for each $t \in (\mathsf{var}(\sigma_0) \setminus \mathsf{var}(\mathsf{head}(\sigma_0)))$, we have that $\theta_0(t) \in (S_V \cup S_N)$;
    (b) for each $t \in \mathsf{varNull}(\mathsf{head}(\sigma_0))$, we have that $\theta_0(t) \in V^*$;
  (2) $\alpha_0 = \mathsf{head}(\sigma_0\theta_0)$;
  (3) $V^* = \mathsf{varNull}(\sigma_0\theta_0)$;
  (4) $V \subseteq \mathsf{var}(V^*)$;
  (5) $V^* \cap (\mathsf{varNull}(\Sigma) \cup S_V \cup S_N) = \emptyset$.
$lp \longleftarrow$ **false**;
$lp\_\text{II} \longleftarrow$ **true**;
$\rho \longleftarrow \sigma_0\theta_0$;
let $n \in \{1, \ldots, N+1\}$ **and** $i \longleftarrow 1$;
while $i \leq n$ *and* $lp =$ **false do**
  $\alpha \longleftarrow \mathsf{head}(\rho)$;
  let $(\alpha', \sigma', \theta')$ be a triple and $\beta' \in \mathsf{body}(\sigma')$ s.t. the following are satisfied:
    (1) $\sigma' \in \Sigma$;
    (2) $\theta' = \theta_{\langle \beta'/\alpha, \sigma' \rangle}$ a substitution;
    (3) $\alpha' = \mathsf{head}(\sigma'\theta')$.
  if *there exists two sets* $\mathsf{body_h}$ *and* $\mathsf{body_b}$ *such that:*
    (1) $\mathsf{body_h} \subseteq \mathsf{body}(\sigma'\theta')$ ***and*** $\mathsf{body_b} \subseteq \mathsf{body}(\sigma'\theta')$*;*
    (2) $\mathsf{body_h} \cap \mathsf{body_b} = \emptyset$*;*
    (3) $\alpha \in \mathsf{body_b}$.
   then
    if $lp\_II =$ **true** *and* $\mathsf{var}(\{\alpha'\} \cup \mathsf{body_h}) \cap \mathsf{var}(\mathsf{body_b}) \neq V$ **then**
      $lp\_\text{II} \longleftarrow$ **false**;
  $V^* \longleftarrow V^* \cap \mathsf{varNull}(\alpha')$;
  $\rho \longleftarrow \vartheta_{[(\mathsf{varNull}(\sigma'\theta') \setminus V^*) \mapsto (S_V \cup S_N)]}(\sigma'\theta')$;
  if $\rho \sim \sigma_0\theta_0$ **then**
    $lp \longleftarrow$ **true**;
  else
    $i \longleftarrow i + 1$;
if $\rho \sim \sigma_0\theta_0$ **then**
  if *the following condition:*
    $lp\_II =$ **false** ***and*** $\mathsf{var}(V^*) = V$*,*
  holds **then**
    return $accept$;
return $reject$;

A key thing to observe in the **while** loop (Lines 14-32) is that we do not store each of the triples $(\alpha', \sigma', \theta')$ but are overwritten in the next iteration (see Lines 15 and 30). As such, the space needed by Algorithm 1 will never go beyond PSPACE although the actual length of the corresponding loop pattern is exponential. In fact, the total space that will be used cannot be more than $O(|\Sigma| \cdot |\mathsf{atoms}(\Sigma)| \cdot \mathsf{maxArt}(\Sigma))$. This is actually the reason for the application of the relabeling function in Line 28 so that assignments are restricted only to the variables and labeled nulls to the set $\mathsf{varNull}(\Sigma) \cup S_V \cup S_N \cup V^*$. Most importantly, the crucial information that we keep from each iteration are the values of the (boolean) variables (or flags) "lp and "lp_II which, as set in Lines 26 and 30, tells us when the derivation path is already a loop pattern (Line 30) or loop separable pattern (Line 26). In regards to the loop separable pattern, we note that our initial nondeterministic guess of the set (of variables) "V" in Line 1 is to denote the final intersection of the variables of all the α_i's in the derivation path and so that a *v*iolation of loop separable pattern condition is detected in Line 35 only if it is the case that $\mathsf{var}(V^*) = V$, and where V^* is the actual computed intersection of the α_i's variables/nulls as computed on Line 27 of each iteration of the **while** loop. Finally, we have in Line 33 that if the termination of the **while** loop of Lines 14-32 corresponded to a loop pattern, then further satisfaction of the condition in Lines 35 tells us that the loop pattern is not "loop separable," in which case Algorithm 1 returns "$accept$. "

Lemma 3. *For a set of TGDs Σ, we have that Σ is not LR iff there exists some computation of* NOTLR(Σ) *such that* NOTLR(Σ) *returns "$accept$."*

Proof. ("$\Longrightarrow$") Then let $P' = (\alpha'_1, \rho'_1)$, ..., (α'_n, ρ'_n) be the loop pattern that is not loop restricted. Then we can make each of the pair $(\alpha'_{n-i}, \rho'_{n-i})$ (for $i \in \{1, \ldots, n-1\}$) correspond to each iterations of the **while** loop in Lines 14-32 of Algorithm 1. Moreover, the fact that P' is not loop restricted and the fact that $\bigcap_{i \in \{1,\ldots,n\}} \mathsf{var}(\alpha'_i) = \mathsf{var}(V^*)$ further implies that we can make the entire computation to be an accepting computation of NOTLR(Σ).

("$\Longleftarrow$") Assume without loss of generality that the (nondeterministic) choice for $n \in \{1, \ldots, N\}$ in Line 13 of Algorithm 1 is k. Then we can construct a loop pattern $P' = (\alpha'_1, \rho'_1)$, ..., (α'_n, ρ'_n) such that $|P'| = k + 1$ inductively as follows (note that our induction will be from $i = n$ to $i = 1$):

Basis: Let (α'_n, ρ'_n) be the pair such that $\rho'_n = \sigma_0\theta_0$ with $\sigma_0\theta_0$ the TGD and assignment corresponding to the triple $(\alpha_0, \sigma_0, \theta_0)$ in Line 1 of Algorithm 1. In addition, also let $(\alpha'_{n-1}, \rho'_{n-1})$ be the pair such that $\rho'_{n-1} = \sigma'\theta'$ with $\sigma'\theta'$ corresponding to the triple $(\alpha', \sigma', \theta')$ of the first iteration (i.e., when $i = 2$ in the **while** loop of Lines 14-32) in Line 16

of Algorithm 1. Moreover, about the assignment $\theta' = \theta_{\langle\beta'/\alpha,\sigma'\rangle}$ as set in Line 18, we further assume that $\theta'(t) \notin \mathsf{varNull}(\sigma_0\theta_0)$ for each $t \in (\mathsf{var}(\sigma') \setminus \mathsf{var}(\beta'))$. Intuitively speaking, the extra assertions about the assignment θ' simply enforces the condition that the other variables not mentioned in β' is mapped by θ' to a set disjoint from $\mathsf{varNull}(\sigma_0\theta_0)$. Note that the aforementioned condition about θ' can possibly introduce an exponential number of variables which we allow since we are now constructing a "particular" loop pattern and not a membership problem. Clearly, we have that $\alpha'_n \in \mathsf{body}(\rho'_{n-1})$.

Inductive step: Assume that we had already defined $(\alpha'_{n-i}, \rho'_{n-i}), \ldots, (\alpha'_n, \rho'_n)$ for $i \in \{1, \ldots, k\}$ which corresponds to the first i-iterations of the **while** loop of Lines 14-32 of Algorithm 1. In addition, also assume that $\alpha'_{n-j} \in \mathsf{body}(\rho'_{n-(j+1)})$ for $j \in \{1, \ldots, i-1\}$.

Then we set $(\alpha'_{n-(i+1)}, \rho'_{n-(i+1)})$ as the pair such that:

1. $\rho'_{n-(i+1)} = \sigma'\theta^*$, where:
   (a) σ' is the TGD corresponding to the $(i+1)^{\text{th}}$-iteration of the **while** loop (Lines 14-32) as mentioned in Line 16;
   (b) θ^* is similar to $\theta' = \theta_{\langle\beta'/\alpha,\sigma'\rangle}$ as in Line 18 but where this time, we also assume that $\theta^*(t) \notin \mathsf{varNull}(\rho'_{n-i})$ for each $t \in (\mathsf{var}(\sigma') \setminus \mathsf{var}(\beta'))$ (similarly to the **basis** above) and where $\alpha = \alpha'_{n-i}$ and $\beta' \in \mathsf{body}(\sigma')$ (as in Line 16).
2. $\alpha'_{n-(i+1)} = \mathsf{head}(\rho'_{n-(i+1)})$.

Lemma 4. *With ρ as defined in Line 29 of Algorithm 1, we have that $\rho \sim \sigma_0\theta_0$ iff $\rho'_{n-(i+1)} \sim \rho'_n$.*

Proof. ("$\Longrightarrow$") First, the "renaming function" $\vartheta_{[(\mathsf{varNull}(\sigma'\theta')\setminus V^*)\mapsto(S_V\cup S_N)]}$ as invoked in Line 28 and the computation of the set V^* in Line 27 enforces $\mathsf{varNull}(\rho) \cap \mathsf{varNull}(\sigma_0\theta_0) = V^*$. Therefore, we have that $\rho\restriction_{V^*} \sim \sigma_0\theta_0\restriction_{V^*}$. Therefore, since this is congruent with the fact that $\rho'_{n-(i+1)}\restriction_{V'} \sim \rho'_n\restriction_{V'}$, where $V' = \mathsf{varNull}(\rho'_{n-(i+1)}) \cap \mathsf{varNull}(\rho'_n)$, then it follows that $\rho'_{n-(i+1)} \sim \rho'_n$ as well.

("$\Longleftarrow$") This direction is similar to the previous one and follows from the fact that $\rho'_{n-(i+1)}\restriction_{V'} \sim \rho'_n\restriction_{V'}$ (where $V' = \mathsf{varNull}(\rho'_{n-(i+1)}) \cap \mathsf{varNull}(\rho'_n)$) is congruent to the fact that $\rho\restriction_{V^*} \sim \sigma_0\theta_0\restriction_{V^*}$ with V^* as computed in Line 27 of Algorithm 1. □

Therefore, it follows from Lemma 4 that P' is indeed a loop pattern. Moreover, since $\text{NOTLR}(\Sigma)$ is an accepting computation and thus, satisfies the condition in Lines 35 of Algorithm 1, then it follows that P' is a loop pattern of Σ that is not loop restricted. This completes the proof of Lemma 3. $\square$

(*Hardness*) We prove hardness from "first principles." Thus, let L be an arbitrary decision problem in PSPACE. Then from the definition of complexity class PSPACE [27], there exists some *deterministic Turing machine* M such that for any string $\mathbf{s}$, $\mathbf{s} \in L$ iff M accepts $\mathbf{s}$ in at most $p(|\mathbf{s}|)$-space. Thus, assume the Turing machine M to be the tuple $\langle Q, \Gamma, \square, \Sigma, \delta, q_0, F\rangle$ such that: (1) $Q \neq \emptyset$ is a finite set of states; (2) $\Gamma \neq \emptyset$ is a finite set of alphabet symbols; (3) $\square \in \Gamma$ is the "blank" symbol; (4) $\Sigma \subseteq \Gamma \setminus \{\square\}$ is the set of input symbols; (5) $\delta : (Q \setminus F) \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function; (6) $q_0 \in Q$ is the initial state; and lastly, (7) $F \subseteq Q$ is the set of final/accepting states.

Now before proceeding to our actual reduction, we first introduce the following helpful notions, let $\mathbf{V}_0 = XXY$ and $\mathbf{V}_1 = YXX$. Intuitively, $\mathbf{V}_0$ encodes the digit $\overline{0}$ and $\mathbf{V}_1$ the digit $\overline{1}$. For example, under the said notions, we have that $\mathbf{V}_0\mathbf{V}_0\mathbf{V}_1$ stands for the tuple $XXYXXYYXX$ since $\underbrace{XXY}_{\mathbf{V}_0}\underbrace{XXY}_{\mathbf{V}_0}\underbrace{YXX}_{\mathbf{V}_1}$. Intuitively, such an encoding scheme will allow us to encode bit-patterns (e.g., as in "00100011," "00100010," "00100110," etc.) to represent both the linear ordering and the current tape cell contents of the Turing machine M and where it also enjoys the propert $\mathbf{V}_0 \not\sim \mathbf{V}_1$. By $\mathbf{Z}$, we denote the three times repetition of the variable Z, i.e., $\mathbf{Z} = ZZZ$. As will be seen later on, this will allow us to determine when a final state is reached.

Let $\mathbf{s} = a_0 \dots a_{|\mathbf{s}|} \in \Gamma^{|\mathbf{s}|}$ be the input string to the machine M. Then define $\Sigma^{\text{MOVE}}_{M(\mathbf{s})}$ as the set of TGDs such that:

$$
\begin{aligned}
&\Sigma^{\text{MOVE}}_{M(\mathbf{s})} \\
&\quad = \big\{\, cf\big(\mathbf{T}_{\langle i\rangle}, \mathsf{stt}(q), \mathsf{num}(k), \mathbf{X}_1, \dots, \mathbf{X}_{k-1}, \mathsf{alp}(a), \mathbf{X}_{k+1}, \dots, \mathbf{X}_{p(|\mathbf{s}|)}\big) \\
&\qquad \rightarrow cf\big(\mathbf{T}_{\langle i\rangle} + 1, \mathsf{stt}(q'), \mathsf{num}(k+1), \\
&\qquad\qquad \mathbf{X}_1, \dots, \mathbf{X}_{k-1}, \mathsf{alp}(b), \mathbf{X}_{k+1}, \dots, \mathbf{X}_{p(|\mathbf{s}|)}\big) \\
&\qquad \mid\ \delta(q,a) = (q', b, R) \text{ and } i \in \{1, \dots, n-1\},\ \text{where} \\
&\qquad\ n = \big\lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|)\big\rceil \text{ and } k \in \{1, \dots, p(|\mathbf{s}|)\}\,\big\}
\end{aligned}
\tag{18}
$$

$$\cup \Big\{\, cf\big(\mathbf{T}_{\langle i\rangle}, \mathsf{stt}(q), \mathsf{num}(k), \mathbf{X}_1, \ldots, \mathbf{X}_{k-1}, \mathsf{alp}(a), \mathbf{X}_{k+1}, \ldots, \mathbf{X}_{p(|\mathbf{s}|)}\big)$$
$$\to cf\big(\mathbf{T}_{\langle i\rangle} + 1, \mathsf{stt}(q'), \mathsf{num}(k-1),$$
$$\mathbf{X}_1, \ldots, \mathbf{X}_{k-1}, \mathsf{alp}(b), \mathbf{X}_{k+1}, \ldots, \mathbf{X}_{p(|\mathbf{s}|)}\big)$$
$$\mid\ \delta(q,a) = (q', b, L) \text{ and } i \in \{1, \ldots, n-1\},\ \text{where}$$
$$n = \lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|)\rceil \text{ and } k \in \{1, \ldots, p(|\mathbf{s}|)\}\ \Big\}, \tag{19}$$

where:

- $\mathbf{T}_{\langle i\rangle}$ $\big($for $i \in \{1, \ldots, n-1\}$ where $n = \lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|)\rceil\big)$ is a tuple of variable *triples* such that $\mathbf{T}_{\langle i\rangle} = \underbrace{\mathbf{T}_1 \ldots \mathbf{T}_i \mathbf{V}_0 \mathbf{V}_1 \ldots \mathbf{V}_1}_{n\text{-times}}$ and $\mathbf{T}_j = T_{j1}T_{j2}T_{j3}$ for $j \in \{1, \ldots, i\}$. In particular, we note that $|\mathbf{T}_{\langle i\rangle}| = 3 \cdot n$ for any $i \in \{1, \ldots, n-1\}$ (with n as previously defined above);

- With $\mathbf{T}_{\langle i\rangle}$ as previously defined above, $\mathbf{T}_{\langle i\rangle}{+}1 = \underbrace{\mathbf{T}_1 \ldots \mathbf{T}_i \mathbf{V}_1 \mathbf{V}_0 \ldots \mathbf{V}_0}_{n\text{-times}}$, (i.e., loosely speaking, $\mathbf{T}_{\langle i\rangle} + 1$ is the successor of $\mathbf{T}_{\langle i\rangle}$ under the binary representation);

- $\mathsf{stt} : Q \longrightarrow \{\mathbf{V}_0, \mathbf{V}_1, \mathbf{Z}\}^n$, where $n = \lceil \log(|Q|)\rceil$, such that assuming an ordering $Q \setminus F = \{q_1, \ldots, q_{|Q|}\}$ of the states in $Q \setminus F$, then we have that for each $i \in \{1, \ldots, |Q|\}$, if $b_0 \ldots b_n$ is the n-length binary string representation of the number i, then $\mathsf{stt}(s_i) = \mathbf{V}_{b_0} \ldots \mathbf{V}_{b_n}$. On the other hand, we have that $\mathsf{stt}(q) = \underbrace{\mathbf{Z} \ldots \mathbf{Z}}_{n\text{-times}}$ for each $q \in F$;

- $\mathsf{num} : \{1, \ldots, 2^{p(|\mathbf{s}|)}\} \longrightarrow \{\mathbf{V}_0, \mathbf{V}_1\}^n$, where $n = p(|\mathbf{s}|)$, and so that if the n-length binary string representation of some number $k \in \{1, \ldots, 2^{p(|\mathbf{s}|)}\}$ is $b_0 \ldots b_n$ $\big($i.e., $b_i \in \{1, 0\}$ for $i \in \{1, \ldots, n\}\big)$, then $\mathsf{num}(k) = \mathbf{V}_{b_0} \ldots \mathbf{V}_{b_n}$;

- $\mathsf{alp} : \Gamma \longrightarrow \{\mathbf{V}_0, \mathbf{V}_1\}^n$, where $n = \lceil \log(|\Gamma|)\rceil$, such that assuming an ordering $\Gamma = \{a'_1, \ldots, a'_{|\Gamma|}\}$ of the alphabets Γ, we have that for each $i \in \{1, \ldots, |\Gamma|\}$, if $b_0 \ldots b_n$ is the n-length binary string representation of the number i, then $\mathsf{alp}(a'_i) = \mathbf{V}_{b_0} \ldots \mathbf{V}_{b_n}$;

- For $i \in \{1, \ldots, k-1, k+1, \ldots, p(|\mathbf{s}|)\}$, $\mathbf{X}_i = X_{i1} \ldots X_{in}$ is an n-length tuple of distinct variables, where (as above) $n = \lceil \log(|\Gamma|)\rceil$.

Intuitively, the set of TGDs $\Sigma^{\text{MOVE}}_{M(\mathbf{s})}$ as previously described above simulates the right and left movements of the head of the machine M. Here, the relational symbol cf (i.e., "cf" for *configuration*) as mentioned in (18) and (19), is of arity:

$$3 \cdot \big\lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|)\big\rceil + \lceil \log(|Q|)\rceil + p(|\mathbf{s}|)\lceil \log(|\Gamma|)\rceil + p(|\mathbf{s}|) \cdot \lceil \log(|\Gamma|)\rceil,$$

and where the arguments of cf is explained via the following illustration:

$$cf\big(\underbrace{\mathbf{T}_{\langle i\rangle}}_{\text{time/step}}\ ,\ \underbrace{\mathsf{stt}(q)}_{\text{current state}},\ \overbrace{\mathsf{num}(k)}^{\text{current head tape position}},\ \underbrace{\mathbf{X}_1,\ldots,\mathbf{X}_{k-1},\ \overbrace{\mathsf{alp}(a)}^{\text{character in cell } k},\ \mathbf{X}_{k+1},\ldots,\mathbf{X}_{p(|\mathbf{s}|)}}_{\text{current tape configuration}}\big).$$

Additionally, to also incorporate the starting and accepting configurations of the machine M on the input string $\mathbf{s}$, we further define the set of TGDs $\Sigma^{\text{INIT}}_{M(\mathbf{s})}$ as follows:

$$\begin{aligned}
&\Sigma^{\text{INIT}}_{M(\mathbf{s})} = \\
&\big\{\, r(X,Z) \\
&\quad \rightarrow cf\big(\mathbf{T}_0, \mathsf{stt}(q_0), \mathsf{num}(0), \mathsf{alp}(a_0), \ldots, \mathsf{alp}(a_{|\mathbf{s}|}), \mathsf{alp}(\square), \ldots, \mathsf{alp}(\square)\big), \qquad (20)\\
&\quad cf\big(\mathbf{T}_0, \mathsf{stt}(q_0), \mathsf{num}(0), \mathsf{alp}(a_0), \ldots, \mathsf{alp}(a_{|\mathbf{s}|}), \mathsf{alp}(\square), \ldots, \mathsf{alp}(\square)\big) \\
&\quad \rightarrow cf\big(\mathbf{T}_0 + 1, \mathsf{stt}(q), \mathsf{num}(1), \mathsf{alp}(b), \ldots, \mathsf{alp}(a_{|\mathbf{s}|}), \mathsf{alp}(\square), \ldots, \mathsf{alp}(\square)\big) \qquad (21)\\
&\quad \mid\ \delta(q_0, a_0) = (q, b, R)\ \big\},
\end{aligned}$$

where $\mathbf{T}_0 = \underbrace{\mathbf{V}_0 \ldots \mathbf{V}_0 \mathbf{V}_0}_{|\mathbf{T}|\text{-times}}$ and $\mathbf{T}_0 + 1 = \underbrace{\mathbf{V}_0 \ldots \mathbf{V}_0 \mathbf{V}_1}_{|\mathbf{T}|\text{-times}}$. In particular, we note from the previous notions that all the variables mentioned in the rule (20) will only be from the set $\{X, Y\}$ since they are simply combinations of the tuples $\mathbf{V}_0 = XXY$ and $\mathbf{V}_1 = YXX$. Then finally to complete our encoding, further define $\Sigma^{\text{ACCEPT}}_{M(\mathbf{s})}$ as follows:

$$\begin{aligned}
&\Sigma^{\text{ACCEPT}}_{M(\mathbf{s})} = \\
&\big\{\, cf\big(\mathbf{T}_{\langle i\rangle}, \mathsf{stt}(q), \mathsf{num}(k), \mathbf{X}_1, \ldots, \mathbf{X}_{k-1}, \mathsf{alp}(a), \mathbf{X}_{k+1}, \ldots, \mathbf{X}_{p(|\mathbf{s}|)}\big) \\
&\quad \rightarrow r(X,Z) \qquad (22)\\
&\quad \mid\ q \in F \text{ and } i \in \{1,\ldots,n-1\},\ \text{where } n = \big\lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|)\big\rceil, \\
&\quad\ k \in \{1,\ldots,p(|\mathbf{s}|)\} \text{ and } a \in \Gamma\ \big\}.
\end{aligned}$$

Lastly, for the rest of the proof, we set $\Sigma_{M(\mathbf{s})} = \Sigma^{\text{INIT}}_{M(\mathbf{s})} \cup \Sigma^{\text{MOVE}}_{M(\mathbf{s})} \cup \Sigma^{\text{ACCEPT}}_{M(\mathbf{s})}$.

Lemma 5. *M accepts $\mathbf{s}$ iff $\Sigma_{M(\mathbf{s})}$ is not LR.*

Proof. ("$\Longrightarrow$") Then we can construct a loop pattern of $\Sigma_{M(\mathbf{s})}$ that is not loop separable. Indeed, such a k-steps accepting computation of $M(\mathbf{s})$ corresponds to a sequence:

$$cf(\mathsf{num}(0), \mathsf{stt}(q_0), \mathsf{num}(0), \mathbf{X}_0), \ldots, cf(\mathsf{num}(k), \mathsf{stt}(q), \mathsf{num}(n_k), \mathbf{X}_k),$$

where $q \in F$ (i.e., the set of accepting states) and each of the $\mathbf{X}_i$ (for $i \in \{1, \ldots, k\}$) corresponds to the tape configuration of the machine M at the i^{th}-step. Then we can map such a sequence into a derivation path $P = (\alpha_1, \rho_1)$, (α_2, ρ_2), ..., (α_n, ρ_n) such that:

- (α_n, ρ_n) is a pair such that $\alpha_n = cf(\mathsf{num}(0), \mathsf{stt}(q_0), \mathsf{num}(0), \mathbf{X}_0)$ and ρ_n the rule (20), i.e.,

$$\rho_n = r(X, Z) \rightarrow cf\big(\mathbf{T}_0, \mathsf{stt}(q_0), \mathsf{num}(0), \mathsf{alp}(a_0), \ldots, \mathsf{alp}(a_{|\mathbf{s}|}), \mathsf{alp}(\square), \ldots, \mathsf{alp}(\square)\big);$$

- (α_2, ρ_2) is a pair such that $\alpha_2 = r(X, Z)$ and ρ_2 the rule (22), i.e.,

$$\rho_2 = cf\big(\mathbf{T}_{\langle i \rangle}, \mathsf{stt}(q), \mathsf{num}(k), \mathbf{X}_1, \ldots, \mathbf{X}_{k-1}, \mathsf{alp}(a), \mathbf{X}_{k+1}, \ldots, \mathbf{X}_{p(|\mathbf{s}|)}\big) \rightarrow r(X, Z),$$

  for some $i \in \{1, \ldots, \lceil p(|\mathbf{s}|) \cdot \log(|\Gamma|) \rceil\}$ and $q \in F$ (the accepting states);

- and lastly, (α_1, ρ_1) is a pair such that $\alpha_1 = cf(\mathsf{num}(0), \mathsf{stt}(q_0), \mathsf{num}(0), \mathbf{X}_0)$ and ρ_1 the rule (20) again, i.e.,

$$\rho_1 = r(X, Z) \rightarrow cf\big(\mathbf{T}_0, \mathsf{stt}(q_0), \mathsf{num}(0), \mathsf{alp}(a_0), \ldots, \mathsf{alp}(a_{|\mathbf{s}|}), \mathsf{alp}(\square), \ldots, \mathsf{alp}(\square)\big). \quad (23)$$

Then since we have that

$$(\mathsf{num}(\bar{\mathbf{i}}), \mathsf{num}(q), \mathsf{num}(\bar{\mathbf{i}}), \mathbf{X}_i) \not\sim (\mathsf{num}(\bar{\mathbf{j}}), \mathsf{num}(q'), \mathsf{num}(\bar{\mathbf{j}}), \mathbf{X}_j)$$

for any $i \neq j$ since $\mathsf{num}(\bar{\mathbf{i}}) \not\sim \mathsf{num}(\bar{\mathbf{j}})$, then we also have that $(\alpha_i, \rho_i) \not\sim (\alpha_j, \rho_j)$. Therefore, since we clearly have that $(\alpha_1, \rho_1) \sim (\alpha_n, \rho_n)$, then it

follows that P is in fact a loop pattern. Thus, it is only left for us to show that P is in fact not loop separable. Indeed, since it follows from a simple induction that $\bigcap_{i\in\{0,\ldots,n-1\}} \mathsf{var}(\alpha_{n-i}) = \{X\}$ while $\mathsf{var}(\rho_{n-i}) = \mathsf{var}(\alpha_{n-i}) = \{X, Y\}$ for $i \in \{1, \ldots, n-3\}$, then since for ρ_2 we have that $\mathsf{body_h}(\rho_2) = \emptyset$, $\mathsf{body_b}(\rho_2) = \{\alpha_3\}$, and $\alpha_2 = r(X, Z)$, then since $\mathsf{var}(\{\alpha_2\} \cup \mathsf{body_h}(\rho_2)) \cap \mathsf{body_b}(\rho_2)) = \{X, Z\} \neq \bigcap_{i\in\{1,\ldots,n\}} \mathsf{var}(\alpha_i) = \{X\}$ (where in particular, we note that since $q \in F$, then $Z \in \mathsf{var}(\mathsf{body_b}(\rho_2))$ because $\mathsf{stt}(q) = \underbrace{\mathbf{Z}\ldots\mathbf{Z}}_{\lceil\log|Q|\rceil\text{-times}}$), then it follows that P is not loop separable.

("$\Longleftarrow$") Firstly, we observe from the construction of $\Sigma_{M(\mathbf{s})} = \Sigma^{\text{INIT}}_{M(\mathbf{s})} \cup \Sigma^{\text{MOVE}}_{M(\mathbf{s})} \cup \Sigma^{\text{ACCEPT}}_{M(\mathbf{s})}$ that due to the linear ordering as enforced by the time argument of cf (i.e., the first tuples $\mathbf{T}_{\langle i\rangle}$ and $\mathbf{T}_{\langle i\rangle} + 1$ mentioned in the rules (18), (19), (20) and (22)) that any loop pattern $P = \langle\alpha_1, \rho_1\rangle, \ldots, \langle\alpha_n, \rho_n\rangle$ of $\Sigma_{M(\mathbf{s})}$, where $\rho_1 = \sigma_1\theta_1$ and $\rho_n = \sigma_n\theta_n$, implies that both σ_1 and σ_n are of the rule (20). This also follows from the fact that assuming $\alpha_i = cf(\mathbf{T}_i, \mathbf{s}_i, \mathbf{p}_i, \mathbf{tp}_i)$ such that $\mathbf{T}_i$, $\mathbf{s}_i$, $\mathbf{p}_i$ and $\mathbf{tp}_i$ are the "time," "state," "tape position" and current "tape configuration," respectively, then we will have that $\mathbf{T}_i \not\sim \mathbf{T}_j$ for each $i, j \in \{1, \ldots, n-1\}$ and $i < j$, but where we have that $\mathbf{T}_1 \sim \mathbf{T}_n$. Thus, using similar arguments above, if we assume that $P = (\alpha_1, \rho_1), \ldots, (\alpha_n, \rho_n)$ is a loop pattern of $\Sigma_{M(\mathbf{s})}$ that is not loop restricted, then it follows that the sequence $(\alpha_2, \rho_2), \ldots, (\alpha_n, \rho_n)$ corresponds to an accepting computation of $M(\mathbf{s})$. □

□

A.9 Proof of Proposition 3

Proposition 3. *With GLR the class of generalized loop restricted TGDs as defined in Definition 10, we have that the following holds:*

- LR $\subsetneq$ GLR*;*
- AC $\subsetneq$ GLR*;*
- ML $\subsetneq$ GLR*;*
- SJ $\subsetneq$ GLR*;*
- aGRD $\subsetneq$ GLR*;*
- DR $\subsetneq$ GLR.

Proof. We prove by considering the individual cases as follows:

("LR $\subsetneq$ GLR"): This follows from the fact that the loop pattern Type I of Definition 10 is actually the loop pattern of Definition 8.

("AC $\subsetneq$ GLR"): On the contrary, assume that there exists some $\Sigma \in$ AC such that $\Sigma \notin$ GLR. Then by Definition 10, there exists some loop pattern $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ such that it is neither of the Types I-V as described in Definition 10. In particular, we have that L is not of the Type II. Then this implies that for all (α_i, ρ_i) $(1 < i \leq n)$, we have that $\mathsf{body}(\rho_i)$ separated into two disjoint body parts $\mathsf{body}(\rho_i) = \mathsf{body_h}(\rho_i) \cup \mathsf{body_b}(\rho_i)$ implies that for all j $(1 \leq j < i)$, one of the following conditions holds:

1. $\mathsf{body_h}(\rho_i) \cap \mathsf{body_b}(\rho_i) \neq \emptyset$, or
2. $\mathsf{var}\big(\{\alpha_j\} \cup \mathsf{body_h}(\rho_i)\big) \cap \mathsf{var}\big(\mathsf{body_b}(\rho_i)\big) \neq \emptyset$.

In particular, if we take $\mathsf{body_h}(\rho_i) = \emptyset$ and $\mathsf{body_b}(\rho_i) = \mathsf{body}(\rho_i)$, for each $i \in \{1, \ldots, n\}$, then since L is a loop pattern (and thus, $\alpha_{i+1} \in \mathsf{body}(\rho_i) = \mathsf{body_b}(\rho_i)$) then we have that Conditions 1 and 2 cannot hold. Therefore, we must have that Condition 3 holds for each (α_i, ρ_i) $(1 \leq i < n)$ (i.e., if we take $\mathsf{body_h}(\rho_i) = \emptyset$ and $\mathsf{body_b}(\rho_i) = \mathsf{body}(\rho_i)$). Then this contradicts the assumption that $\Sigma \in$ aGRD because this implies a cycle in the "firing graph" [3].

("ML $\subsetneq$ GLR"): On the contrary, assume that there exists some $\Sigma \in$ ML such that $\Sigma \notin$ GLR. Then again by Definition 10, there exists some loop pattern $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ such that it is neither of the Types I-V as described in Definition 10. In particular, we have that L is not of the Type III. Then this implies that there exists some (α_i, ρ_i) $(1 \leq i < n)$ such that $\mathsf{var}(\rho_i) \not\subseteq \mathsf{var}(\beta)$, for some $\beta \in \mathsf{body}(\rho_i)$. Therefore, since $\rho_i = \sigma_i\theta_i$, for some $\sigma_i \in \Sigma$ and assignment θ_i, then it follows that there exists some $\beta' \in \mathsf{body}(\sigma_i)$ such that $\mathsf{var}(\sigma_i) \not\subseteq \mathsf{var}(\beta')$. Then this contradicts the assumption that $\Sigma \in$ ML.

("SJ $\subsetneq$ GLR"): On the contrary, assume that there exists some $\Sigma \in$ SJ such that $\Sigma \notin$ GLR. Then again by Definition 10, there exists some loop pattern $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ such that it is neither of the Types I-V as described in Definition 10. In particular, we have that L is not of the Type IV. Then this implies that there exists some pair (α_i, ρ_i) in L $(1 \leq i < n)$ such that $\big(\mathsf{var}(\alpha_i) \cap \mathsf{var}(\beta)\big) \not\subseteq \bigcap_{j=i+1}^{n} \mathsf{var}(\alpha_j)$, for

some $\beta \in \mathsf{body}(\rho_{i+1}) \setminus \{\alpha_i\}$. Then this again contradicts the assumption that $\Sigma \in$ SJ since the "expansion" of Σ [11] (which correspond to the loop pattern) will contain a marked variable that occurs in two different atoms;

("aGRD $\subsetneq$ GLR"): On the contrary, assume that there exists some $\Sigma \in$ aGRD and $\Sigma \notin$ GLR. Then again by Definition 10, there exists some loop pattern $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ such that it is neither of the Types I-V as described in Definition 10. In particular, we have that L is not of the Type V. Then we have from the definition of loop restricted Type V that Σ will have cycle in the rule dependency graph, which contradicts the assumption that $\Sigma \in$ aGRD;

("DR $\subsetneq$ GLR"). This follows from the definition of *domain restricted* TGDs that for each rule σ we have that:

$$\mathsf{var}(\mathsf{body}(\sigma)) \subseteq \mathsf{var}\big(\mathsf{head}(\sigma)\big). \tag{24}$$

Now let $\Sigma \in$ DR and let $L = (\alpha_1, \rho_1) \cdots (\alpha_n, \rho_n)$ be a loop pattern of Σ. Then we have that L is of the loop pattern of Type I of Definition 10. Indeed, we will show that for each $i = 1$ to $i = n$, we have that $\mathsf{body}(\rho_i)$ can be separated into two disjoint parts $\mathsf{body}(\rho_i) = \mathsf{body}_{\mathsf{h}}(\rho_i) \cup \mathsf{body}_{\mathsf{b}}(\rho_i)$ such that the following conditions holds:

1. $\mathsf{body}_{\mathsf{h}}(\rho_i) \cap \mathsf{body}_{\mathsf{b}}(\rho_i) = \emptyset$,
2. $\alpha_{i+1} \in \mathsf{body}_{\mathsf{b}}(\rho_i)$,
3. $\mathsf{var}\big(\{\alpha_i\} \cup \mathsf{body}_{\mathsf{h}}(\rho_i)\big) \cap \mathsf{var}\big(\mathsf{body}_{\mathsf{b}}(\rho_i)\big) = \bigcap_{j=1}^{n} \mathsf{var}(\alpha_j)\, \emptyset$.

Indeed, for each i, we partition $\mathsf{body}(\rho_i)$ into two sets $\mathsf{body}_{\mathsf{h}}(\rho_i)$ and $\mathsf{body}_{\mathsf{b}}(\rho_i)$ such that:

$$\mathsf{body}_{\mathsf{h}}(\rho_i) = \emptyset; \tag{25}$$
$$\mathsf{body}_{\mathsf{b}}(\rho_i) = \mathsf{body}(\rho_i), \tag{26}$$

i.e., we set $\mathsf{body}_{\mathsf{b}}(\rho_i)$ to be all the body atoms of ρ_i while we set $\mathsf{body}_{\mathsf{h}}(\rho_i)$ to be empty. Then it is not too difficult to see that $\mathsf{body}_{\mathsf{h}}(\rho_i)$ and $\mathsf{body}_{\mathsf{b}}(\rho_i)$ is clearly a partition of $\mathsf{body}(\rho_i)$. Moreover, because of Con-

dition (24), we further have that:

$$\begin{aligned}
&\mathsf{var}\big(\{\alpha_i\} \cup \mathsf{body_h}(\rho_i)\big) \cap \mathsf{var}\big(\mathsf{body_b}(\rho_i)\big) \\
=\ &\mathsf{var}(\{\alpha_i\}) \cap \mathsf{var}\big(\mathsf{body_b}(\rho_i)\big) \text{ (i.e., since we set } \mathsf{body_h} = \emptyset) \\
=\ &\bigcap_{j=1}^{n} \mathsf{var}(\alpha_j).
\end{aligned}$$

□

A.10 Proof of Theorem 8

Theorem 8. *The class of generalized loop restricted patterns satisfies BDTDP.*

Proof. ("Types I, II and V"): The proof follows similarly to that as in Proposition 11 and Proposition 12 of [13].

("Type IV"): Consider a loop pattern $L = (w_1, \ldots, w_n)$ of Type IV (as defined in Definition 10). Then by the definition of a loop pattern (see Definition 7), we have that $w_1 \sim w_n$ and $w_i \not\sim w_j$ for any other i, j $(1 \leq i, j \leq n)$. Then assuming that $w_i = (\alpha\theta_i, \sigma\theta_i)$, for each $i \in \{1, \ldots, n\}$, let us consider the following two only possibilities:

Case 1: $\exists \beta_1\theta_1, \beta_2\theta_1 \in \mathsf{body}(\sigma\theta_1)$, where $\beta_1\theta_1 \neq \beta_2\theta_1$, s.t. $\big(\mathsf{var}(\beta_1\theta_1) \cap \mathsf{var}(\beta_2\theta_1)\big) \neq \emptyset$:

Then we can assume without loss of generality that the loop pattern $L = (w_1, \ldots, w_n)$ is such that:

$$w_1 = \big(\alpha\theta_1, \text{“}\beta\theta_1, \widehat{\mathsf{B_b}\theta_1}, \widehat{\mathsf{B_h}\theta_1} \rightarrow \alpha\theta_1\text{”}\big); \tag{27}$$

$$w_2 = \big(\alpha_2\theta_2, \sigma_2\theta_2\big); \tag{28}$$

$$\vdots$$

$$w_{n-1} = \big(\alpha_{n-1}\theta_{n-1}, \sigma_{n-1}\theta_{n-1}\big); \tag{29}$$

$$w_n = \big(\alpha\theta_n, \text{“}\beta\theta_n, \widehat{\mathsf{B_b}\theta_n}, \widehat{\mathsf{B_h}\theta_n} \rightarrow \alpha\theta_n\text{”}\big), \tag{30}$$

where:

- $\sigma =$ “$\beta, \widehat{\mathsf{B_b}}, \widehat{\mathsf{B_h}} \rightarrow \alpha$” is some TGD rule of Σ, where $\mathsf{body}(\sigma) = \{\beta\} \cup \mathsf{B_b} \cup \mathsf{B_h}$, $\beta \notin \big(\mathsf{B_b} \cup \mathsf{B_h}\big)$ and $\widehat{\mathsf{B_b}}$ and $\widehat{\mathsf{B_h}}$ denotes the conjunctions of the atoms in $\mathsf{B_b}$ and $\mathsf{B_h}$, respectively;

- θ_1 and θ_n are assignments on the rule $\sigma \in \Sigma$;
- $\beta'\theta_n \in \mathsf{B_b}\theta_n$ implies $\big(\mathsf{var}(\beta\theta_n) \cap \mathsf{var}(\beta'\theta_n)\big) \neq \emptyset$;
- $\beta'\theta_n \in \mathsf{B_h}\theta_n$ implies $\big(\mathsf{var}(\beta\theta_n) \cap \mathsf{var}(\beta'\theta_n)\big) = \emptyset$;
- $\alpha_2\theta_2 = \beta\theta_1 \in \mathsf{body}(\sigma\theta_1)$.

Then since the loop pattern $L = (w_1, \ldots, w_n)$ is of Type IV, then we have that

$$\bigcup_{\beta'\theta_n \in \mathsf{B_b}\theta_n} \big(\mathsf{var}(\beta\theta_n) \cap \mathsf{var}(\beta'\theta_n)\big) \subseteq \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i). \tag{31}$$

Now let us define the assignment $\theta' : \mathsf{var}(\sigma_n\theta_n) \longrightarrow \mathsf{var}(\sigma_1\theta_1)$ as follows:

$$\begin{aligned}\theta' = &\big\{\, \theta_n(X) \mapsto \theta_n(X) \;\big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \in \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \,\big\} \\ &\cup \big\{\, \theta_n(X) \mapsto \theta_1(X) \;\big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \notin \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \,\big\}.\end{aligned} \tag{32}$$

Then since $\sigma\theta_1 \sim \sigma\theta_n$ and by Condition (31), then it follows that with $w_n^* = \big(\alpha\theta^*, \text{“}\beta\theta^*, \widehat{\mathsf{B_b}\theta^*}, \widehat{\mathsf{B_h}\theta^*} \rightarrow \alpha\theta^*\text{”}\big)$, where $\theta^* = \theta' \circ \theta_n$, we have that $w_n^* = w_1$. Therefore, using the same argument as to the proof of Proposition 12 in [13], we have that any occurrence of the loop pattern $L = (w_1, \ldots, w_n)$ in a derivation tree $T(\Sigma)$ implies another derivation tree $T'(\Sigma)$ such that $T(\Sigma)$ subsumes $T'(\Sigma)$ by “replacing” the derivation path $L = (w_1, \ldots, w_n)$ by the singleton $w^* = \big(\alpha\theta_1, \text{“}\beta\theta^*, \widehat{\mathsf{B_b}\theta_1}, \widehat{\mathsf{B_h}\theta_1} \rightarrow \alpha\theta_1\text{”}\big)$ (see Fig. 2 in the proof of Proposition 12 of [13]).

Case 2: $\forall\beta_1\theta_1, \beta_2\theta_1 \in \mathsf{body}(\sigma\theta_1)$ such that $\beta_1\theta_1 \neq \beta_2\theta_1$ implies that $\big(\mathsf{var}(\beta_1\theta_1) \cap \mathsf{var}(\beta_2\theta_1)\big) = \emptyset$:

The key to proving this case is similar to the ideas of the previous case above. Indeed, let us assume again a loop pattern $L =$

$(w_1, \ldots, w_n)$ such that:

$$
\begin{aligned}
w_1 &= \big(\alpha\theta_1, \text{“}\beta\theta_1, \widehat{\mathsf{B}\theta_1} \rightarrow \alpha\theta_1\text{”}\big); && (33)\\
w_2 &= \big(\alpha_2\theta_2, \sigma_2\theta_2\big); && (34)\\
&\vdots \\
w_{n-1} &= \big(\alpha_{n-1}\theta_{n-1}, \sigma_{n-1}\theta_{n-1}\big); && (35)\\
w_n &= \big(\alpha\theta_n, \text{“}\beta\theta_n, \widehat{\mathsf{B}\theta_n} \rightarrow \alpha\theta_n\text{”}\big), && (36)
\end{aligned}
$$

where:

- $\sigma = \text{“}\beta, \widehat{\mathsf{B}} \rightarrow \alpha\text{”}$ is some TGD rule of Σ, where $\mathsf{body}(\sigma) = \{\beta\} \cup \mathsf{B}$, $\beta \notin \mathsf{B}$ and $\widehat{\mathsf{B}}$ denotes the conjunctions of the atoms in B;
- θ_1 and θ_n are assignments on the rule $\sigma \in \Sigma$;
- $\beta'\theta_n \in \mathsf{B}\theta_n$ implies $\big(\mathsf{var}(\beta\theta_n) \cap \mathsf{var}(\beta'\theta_n)\big) = \emptyset$;
- $\alpha_2\theta_2 = \beta\theta_1 \in \mathsf{body}(\sigma\theta_1)$.

Then we also define the assignment $\theta' : \mathsf{var}(\sigma_n\theta_n) \longrightarrow \mathsf{var}(\sigma_1\theta_1)$ as follows:

$$
\begin{aligned}
\theta' = &\Big\{ \theta_n(X) \mapsto \theta_n(X) \;\Big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \in \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \Big\} \\
&\cup \Big\{ \theta_n(X) \mapsto \theta_1(X) \;\Big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \notin \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \Big\}.
\end{aligned} \qquad (37)
$$

Then similarly to the previous case above, because $\sigma\theta_1 \sim \sigma\theta_n$, then it follows that with $w_n^* = \big(\alpha\theta^*, \text{“}\beta\theta^*, \widehat{\mathsf{B}\theta^*} \rightarrow \alpha\theta^*\text{”}\big)$, where $\theta^* = \theta' \circ \theta_n$, we have that $w_n^* = w_1$. Therefore, using again the same argument as to the proof of Proposition 12 in [13], we have that any occurrence of the loop pattern $L = (w_1, \ldots, w_n)$ in a derivation tree $T(\Sigma)$ implies another derivation tree $T'(\Sigma)$ such that $T(\Sigma)$ subsumes $T'(\Sigma)$ by “replacing” the derivation path $L = (w_1, \ldots, w_n)$ by the singleton $w^* = \big(\alpha\theta_1, \text{“}\beta\theta^*, \widehat{\mathsf{B}\theta_1} \rightarrow \alpha\theta_1\text{”}\big)$ (see Fig. 2 in the proof of Proposition 12 of [13]).

("Type III"): The key to proving this case is also similar to the ideas of the previous two cases above. Let us now assume a loop pattern $L = (w_1, \ldots, w_n)$ of Type III such that:

$$w_1 = \big(\alpha\theta_1, \text{“}\beta\theta_1, \widehat{\mathsf{B}\theta_1} \rightarrow \alpha\theta_1\text{”}\big); \tag{38}$$

$$w_2 = \big(\alpha_2\theta_2, \sigma_2\theta_2\big); \tag{39}$$

$$\vdots$$

$$w_{n-1} = \big(\alpha_{n-1}\theta_{n-1}, \sigma_{n-1}\theta_{n-1}\big); \tag{40}$$

$$w_n = \big(\alpha\theta_n, \text{“}\beta\theta_n, \widehat{\mathsf{B}\theta_n} \rightarrow \alpha\theta_n\text{”}\big), \tag{41}$$

where:

- $\sigma = \text{“}\beta, \widehat{\mathsf{B}} \rightarrow \alpha\text{”}$ is some TGD rule of Σ, where $\mathsf{body}(\sigma) = \{\beta\} \cup \mathsf{B}$, $\beta \notin \mathsf{B}$ and $\widehat{\mathsf{B}}$ denotes the conjunctions of the atoms in B;
- θ_1 and θ_n are assignments on the rule $\sigma \in \Sigma$;
- $\alpha_2\theta_2 = \beta\theta_1 \in \mathsf{body}(\sigma\theta_1)$.

Then we again define the assignment $\theta' : \mathsf{var}(\sigma_n\theta_n) \longrightarrow \mathsf{var}(\sigma_1\theta_1)$ as follows:

$$\begin{aligned}\theta' = &\Big\{ \theta_n(X) \mapsto \theta_n(X) \;\Big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \in \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \Big\} \\ &\cup \Big\{ \theta_n(X) \mapsto \theta_1(X) \;\Big|\; X \in \mathsf{var}(\sigma) \text{ and } \theta_n(X) \notin \bigcap_{i=1}^{i=n} \mathsf{var}(\alpha_i\theta_i) \Big\}.\end{aligned} \tag{42}$$

Then similarly to the previous two case above, because $\sigma\theta_1 \sim \sigma\theta_n$, then it follows that with $w_n^* = \big(\alpha\theta^*, \text{“}\beta\theta^*, \widehat{\mathsf{B}\theta^*} \rightarrow \alpha\theta^*\text{”}\big)$, where $\theta^* = \theta' \circ \theta_n$, we have that $w_n^* = w_1$. Therefore, using again the same argument as to the proof of Proposition 12 in [13], we have that any occurrence of the loop pattern $L = (w_1, \ldots, w_n)$ in a derivation tree $T(\Sigma)$ implies another derivation tree $T'(\Sigma)$ such that $T(\Sigma)$ subsumes $T'(\Sigma)$ by "replacing" the derivation path $L = (w_1, \ldots, w_n)$ by the singleton $w^* = \big(\alpha\theta_1, \text{“}\beta\theta^*, \widehat{\mathsf{B}\theta_1} \rightarrow \alpha\theta_1\text{”}\big)$ (see Fig. 2 in the proof of Proposition 12 of [13]).

Finally, we conclude the proof by showing how a combination of loop pattern Types I-V can be made to "contract" under some derivation path of a derivation tree. Indeed, using again similar ideas from [13], let

$$P = (w_{11}, \ldots, w_{1k_1}, \underbrace{w'_{11}, \ldots, w'_{1l}}_{L_1}, w_{21}, \ldots, w_{2k_2}, \underbrace{w'_{21}, \ldots, w'_{2l}}_{L_2}, w_{31}, \ldots, w_{3k_3})$$

be a derivation path and assume that $L_1 = (w'_{11}, \ldots, w'_{1l})$ and $L_2 = (w'_{21}, \ldots, w'_{2l})$ are loop patterns such that $L_1 \sim L_2$ and both L_1 and L_2 are of Types II and V, i.e., loop patterns L_1 and L_2 are of either Types II or V occurring more than once. Then there exists some $i \in \{1, \ldots, l\}$, such that assuming

$$w'_{li} = \big(\alpha_i\theta_{li},\ \text{“}\widehat{\mathsf{body}_\mathsf{b}(\sigma_i}\theta_{li}), \widehat{\mathsf{body}_\mathsf{h}(\sigma_i}\theta_{li}) \to \alpha_i\theta_{li}\text{”}\big)$$

$\big($for $l \in \{1, 2\}\big)$, we have that

1. $\mathsf{body}_\mathsf{h}(\sigma_i\theta_{li}) \cap \mathsf{body}_\mathsf{b}(\sigma_i\theta_{li}) = \emptyset$;
2. $\alpha_{i+1}\theta_{li+1} \in \mathsf{body}_\mathsf{b}(\sigma_i\theta_{li})$;
3. $\mathsf{var}\big(\{\alpha_i\theta_{li}\} \cup \mathsf{body}_\mathsf{h}(\sigma_i\theta_{li})\big) \cap \mathsf{var}\big(\mathsf{body}_\mathsf{b}(\sigma_i\theta_{li})\big) = \emptyset$.

Then we have that there exists some assignment θ^* such that

$$w'_{1i} = \big(\alpha_i\theta_{1i},\ \text{“}\widehat{\mathsf{body}_\mathsf{b}(\sigma_i}\theta_{2i})\theta^*, \widehat{\mathsf{body}_\mathsf{h}(\sigma_i}\theta_{1i}) \to \alpha_i\theta_{1i}\text{”}\big)$$

i.e., $\mathsf{body}_\mathsf{b}(\sigma_i\theta_{2i})\theta^* = \mathsf{body}_\mathsf{b}(\sigma_i\theta_{1i})$. Then it follows that we can replace the derivation path P with the derivation path

$$P^* = (w_{11}, \ldots, w_{1k_1}, w'_{11}, \ldots, w'_{1i-1}w^*_i, w'_{2i+1}, \ldots, w'_{2l}, w_{31}, \ldots, w_{3k_3}),$$

where

$$w^*_i = \big(\alpha_i\theta_{1i},\ \text{“}\widehat{\mathsf{body}_\mathsf{b}(\sigma_i}\theta_{2i})\theta^*, \widehat{\mathsf{body}_\mathsf{h}(\sigma_i}\theta_{1i}) \to \alpha_i\theta_{1i}\text{”}\big).$$

On the other hand, if we have a derivation path

$$P = (w_{11}, \ldots, w_{1k_1}, \underbrace{w'_1, \ldots, w'_l}_{L}, w_1, \ldots, w_{2k_2})$$

such that $L = (w'_1, \ldots, w'_l)$ is a loop pattern of either Types I, III or IV. Then as we have seen for the cases of "Type I, " "Type IV, " and "Type III" above, if follows that the derivation path P can be replaced with a derivation path

$$P^* = (w_{11}, \ldots, w_{1k_1}, w^*, w_1, \ldots, w_{2k_2}),$$

which is obtained from P by replacing the loop $L = (w'_1, \ldots, w'_l)$ with the singleton w^*.

Finally, because we have from Proposition 2 that there are only a finite number of loop patterns (up to equivalence "$\sim$"), then it follows that GLR TGDs can be characterized by only a finite number of derivation trees of bounded length. $\square$