Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Finite Model Reasoning in Hybrid Classes of Existential Rules

Georg Gottlob[1], **Marco Manna**[2] **and Andreas Pieris**[3]

[1]Department of Computer Science, University of Oxford
[2]Department of Mathematics and Computer Science, University of Calabria
[3]School of Informatics, University of Edinburgh

georg.gottlob@cs.ox.ac.uk, manna@mat.unical.it, apieris@inf.ed.ac.uk

Abstract

Two paradigmatic restrictions that have been studied for ensuring the decidability of query answering under existential rules are guardedness and stickiness. With the aim of consolidating these restrictions, a flexible condition, called tameness, has been proposed a few years ago, which relies on hybrid reasoning, i.e., a combination of forward and backward procedures. The complexity of query answering under this hybrid class of existential rules is by now well-understood. However, the complexity of finite query answering, i.e., query answering under finite models, has remained an open problem. Closing this problem is the main goal of this work.

1 Introduction

Rule-based languages lie at the core of knowledge representation and databases. In knowledge representation they are used for declarative problem solving, and, more recently, to model and reason about ontological knowledge, while in database applications they usually serve as expressive query languages that go beyond standard first-order queries. A prominent rule-based formalism is *Datalog* [Abiteboul *et al.*, 1995]. Even though Datalog is quite powerful, with a variety of different applications, it is widely agreed that its inability to infer the existence of new values that are not already in the input database is a crucial limitation [Patel-Schneider and Horrocks, 2007]. *Existential rules* (a.k.a. *tuple-generating dependencies* and *Datalog$^{\pm}$ rules*), overcome this limitation by enriching Datalog with existential quantification in rule-heads. However, this leads to the undecidability of the main algorithmic tasks, and, in particular, of conjunctive query (CQ) answering [Beeri and Vardi, 1981; Calì *et al.*, 2013], i.e., the problem of checking whether a CQ is entailed by every model of an extensional database and a set of existential rules.

This negative outcome has led to a flurry of research activity for identifying restrictions on existential rules that ensure the decidability of query answering. Two such restrictions, which are of central importance for the present work, are *guardedness* [Baget *et al.*, 2011; Calì *et al.*, 2013] and *stickiness* [Calì *et al.*, 2012]. It is well-known that guardedness is well-suited for forward reasoning, which in turn relies on the fact that guarded rules admit tree-like universal models. However, stickiness does not enjoy the above tree model property, and instead it relies on backward reasoning.

As one might expect, there are useful statements that can be expressed via guarded rules, but not with sticky rules, and vice versa. For example, using guarded rules we can state that the supervisor of a senior employee is also a senior employee, which is provably not expressible via sticky rules:

$$\text{SeniorEmp}(x), \text{HasSupervisor}(x, y) \rightarrow \text{SeniorEmp}(y).$$

This rule is guarded since it has an atom in its body, called a guard, that contains x and y. However, using guarded rules we cannot say, e.g., that senior employees earn more money than junior employees. This is expressible via the sticky rule:

$$\text{SeniorEmp}(x), \text{JuniorEmp}(y) \rightarrow \text{MoreThan}(x, y).$$

The goal of stickiness is to capture joins that are not expressible via guarded rules. This is done by forcing the join variables to be propagated to the inferred atoms. The above rule is trivially sticky since there are no join variables in its body.

It is a natural question to ask whether the above two inherently different classes of existential rules can be consolidated into a single formalism. This has been thoroughly investigated in [Gottlob *et al.*, 2013]. It has been observed that the naive combination of guardedness and stickiness leads to the undecidability of query answering. This led to the notion of *tameness*, which provides an elegant and flexible way for taming the interaction between guarded and sticky rules. The essence of tameness is as follows: *sticky rules are not allowed to trigger the guard of a guarded rule.* It is easy to verify that the above two rules jointly satisfy the tameness condition.

Conjunctive query answering under the tamed combination of guardedness and stickiness is by now well-understood. A sophisticated hybrid query answering algorithm, which relies on a combination of forward and backward reasoning, has been devised in [Gottlob *et al.*, 2013], which led to optimal complexity results: 2EXPTIME-complete in combined complexity, and PTIME-complete in data complexity. Notice that the analysis performed in [Gottlob *et al.*, 2013] considers unrestricted models. However, in many KR applications the domain of interest is actually finite, and thus it is more realistic

to reason over finite models. This has been already observed in the KR community, and there are several works that focus on finite model reasoning; see, e.g., [Amendola *et al.*, 2017; Calvanese, 1996; Ibáñez-García *et al.*, 2014; Rosati, 2008]. Moreover, for intelligent database applications, the finiteness of the models is a key assumption. The question that comes up is whether *finite query answering*, i.e., query answering focussing on finite models, under tameness remains decidable, and, if this is the case, what is the exact complexity. Closing this non-trivial open problem is the main goal of this work.

The main outcome of our analysis is that finite query answering under tameness has the same complexity as (unrestricted) query answering. This is obtained by showing that the tamed combination of guardedness and stickiness enjoys *finite controllability*, i.e., finite and unrestricted query answering coincide. This exploits the fact that both guardedness and stickiness enjoy finite controllability [Bárány *et al.*, 2014; Gogacz and Marcinkowski, 2017]. At this point, let us say that, in general, combining guardedness with unguarded classes of existential rules that are finitely controllable, by following the same approach as tameness, does not guarantee finite controllability. As we show later, there exist finitely controllable classes that their tamed combination with guardedness leads to classes that are *not* finitely controllable. Thus, to establish the above result, we need to perform a detailed model-theoretic analysis based on guardedness and stickiness. Our results can be summarized as follows:

▶ We first focus on a subclass of tame rules, obtained by posing a stratification condition on guarded and sticky rules, and show that is finitely controllable.

▶ We then establish that a tame set of guarded and sticky rules can be rewritten as a stratified one that preserves finite answers, and thus tameness ensures finite controllability. This result immediately implies that finite query answering under tame guarded and sticky rules is 2ExpTime-complete in combined complexity, and PTime-complete in data complexity. The stratification relies on the fact that sticky rules are UCQ-rewritable. To establish the correctness of the stratification, we isolate a well-behaved family of finite models, called *sticky-supported*, that is a universal finite model set.

▶ Finally, triggered by the above construction, we study the relative expressive power among stratification and tameness. Interestingly, tame rules are *more expressive* than stratified rules w.r.t. the program expressive power [Arenas *et al.*, 2014]. This essentially says that there is a tame set of guarded and sticky rules that cannot be expressed as a stratified set that gives the same answer for every database and CQ.

2 Preliminaries

Basics. Let $\mathbf{C}$, $\mathbf{N}$ and $\mathbf{V}$ be disjoint countably infinite sets of *constants*, *(labeled) nulls*, and *(regular) variables*, respectively. The elements of $(\mathbf{C} \cup \mathbf{N} \cup \mathbf{V})$ are also called *terms*. An *atom* is an expression of the form $R(\bar{t})$, where R is an n-ary predicate, and $\bar{t} = (t_1, \ldots, t_n)$ are terms. A *fact* is an atom that contains only constants from $\mathbf{C}$. A *homomorphism* from a set of atoms A to a set of atoms B is a mapping h from the terms in A to the terms in B, which is the identity on $\mathbf{C}$, such that $R(\bar{t}) \in A$ implies $h(R(\bar{t})) = R(h(\bar{t})) \in B$.

Databases and conjunctive queries. An *instance* is a (possibly infinite) set of atoms with constants and nulls, while a *database* is a finite set of facts. The *active domain* of an instance I, denoted $\mathsf{dom}(I)$, is the set of all terms in I.

A *conjunctive query* (CQ) is a formula of the form $q(\bar{x}) := \exists \bar{y}\big(R_1(\bar{v}_1) \wedge \cdots \wedge R_m(\bar{v}_m)\big)$, where each $R_i(\bar{v}_i)$ is an atom without nulls, each variable mentioned in the $\bar{v}_i$'s appears either in $\bar{x}$ or $\bar{y}$, and $\bar{x}$ are the free variables of q. If $\bar{x}$ is empty, then q is a *Boolean CQ*. As usual, the evaluation of CQs is defined in terms of homomorphisms. Let I be an instance, and $q(\bar{x})$ a CQ as above. The *evaluation of* $q(\bar{x})$ *over* I, denoted $q(I)$, is defined as the set of tuples $\bar{c} \in \mathbf{C}^{|\bar{x}|}$ for which there is a homomorphism h such that $h(q(\bar{x})) \subseteq I$ and $h(\bar{x}) = \bar{c}$. Notice that, by abuse of notation, we sometimes treat a conjunction of atoms as a set of atoms. A *union of conjunctive queries* (UCQ) is a formula $q(\bar{x}) := \bigvee_{1 \leq i \leq n} q_i(\bar{x})$, where each $q_i(\bar{x})$ is a CQ. The *evaluation of* $q(\bar{x})$ *over* I, denoted $q(I)$, is the set of tuples of constants $\bigcup_{1 \leq i \leq n} q_i(I)$.

Tuple-generating dependencies. A *tuple-generating dependency* (TGD) (a.k.a. *existential rule*) is a sentence of the form $\forall \bar{x} \forall \bar{y}\big(\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})\big)$, where ϕ, ψ are (non-empty) conjunctions of atoms without constants and nulls. We write this TGD as $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for joining atoms. We call ϕ and ψ the *body* and *head* of the TGD, respectively. An instance I satisfies the TGD σ above, written $I \models \sigma$, if, whenever there is a homomorphism h such that $h(\phi(\bar{x}, \bar{y})) \subseteq I$, then there is h' that agrees with h on $\bar{x}$ such that $h'(\psi(\bar{x}, \bar{z})) \subseteq I$. The instance I satisfies a set Σ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$. Let $\mathbb{TGD}$ be the class of finite sets of TGDs. Let us clarify that in the rest of the paper we work only with finite sets of TGDs.

Query answering under TGDs. For a database D and a set Σ of TGDs, a *model* of D and Σ is an instance $I \supseteq D$ such that $I \models \Sigma$. Let $\mathsf{fmods}(D, \Sigma)$ and $\mathsf{mods}(D, \Sigma)$ be the set of finite and unrestricted models of D and Σ. The *finite certain answers* to a CQ q w.r.t. D and Σ is $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) = \bigcap_{I \in \mathsf{fmods}(D, \Sigma)} q(I)$. We also define the *certain answers* to q w.r.t. D and Σ, denoted $\mathsf{cert}(q, D, \Sigma)$, where we consider models from $\mathsf{mods}(D, \Sigma)$. Recall that $\mathsf{cert}(q, D, \Sigma)$ coincides with the evaluation of q over $\mathsf{chase}(D, \Sigma)$, i.e., the (possibly infinite) instance constructed by the well-known *chase procedure* on D and Σ; see, e.g., [Calì *et al.*, 2013].

Our main problem in this work is to compute the finite certain answers to a CQ w.r.t. a database and a set of TGDs that falls in a certain class $\mathbb{C}$, i.e., $\mathbb{C} \subseteq \mathbb{TGD}$; concrete classes of TGDs are given below. As is customary when studying the complexity of this problem, we focus on its decision version:

PROBLEM :	$\mathsf{FinQAns}(\mathbb{C})$
INPUT :	A database D, a set $\Sigma \in \mathbb{C}$ of TGDs, a CQ $q(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{\lvert\bar{x}\rvert}$.
QUESTION :	Does $\bar{c} \in \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$?

We also refer to $\mathsf{QAns}(\mathbb{C})$, which is defined as above with the difference that we ask whether $\bar{c}$ belongs to $\mathsf{cert}(q, D, \Sigma)$. For both problems, we use the standard complexity measures, i.e., *combined complexity* and *data complexity*.

Concrete classes of TGDs. It is well-known that both $\mathsf{FinQAns}(\mathbb{TGD})$ and $\mathsf{QAns}(\mathbb{TGD})$ are undecidable problems. This has led to a flurry of activity for identifying restrictions on TGDs that make the above problems decidable. Here, we concentrate on a class obtained by combining two of the main decidability paradigms, i.e., guardedness and stickiness.

Guardedness: A TGD is called *guarded* if it has a body atom, called *guard*, that contains all the body variables [Calì *et al.*, 2013]. Let $\mathbb{G}$ be the class of sets of guarded TGDs.

Stickiness: The goal of stickiness is to capture joins that are not expressible via guarded TGDs. The key idea can be described as follows: variables that appear more than once in the body of a TGD should be inductively propagated (or "stick") to the head atoms. The definition is based on an inductive procedure that marks the variables that violate the above property. Consider a set Σ of TGDs. During the base step, a variable that appears in the body of a TGD of Σ but not in its head is marked. Now, if a variable x in the head of a TGD $\sigma \in \Sigma$ appears at position $R[i]$ (i.e., the i-th attribute of the predicate R), and there exists some $\sigma' \in \Sigma$ that has in its body at position $R[i]$ a marked variable, then x in the body of σ is marked. We say that Σ is sticky if every marked variable appears only once in the body of a TGD. For more details see [Calì *et al.*, 2012]. Let $\mathbb{S}$ be the class of sticky sets of TGDs.

Guardedness + Stickiness: The notion of *tameness* has been introduced in [Gottlob *et al.*, 2013], with the aim of taming the interaction between guarded and sticky TGDs. The idea is to allow guarded and sticky TGDs to co-exist as long as none of the sticky TGDs triggers the guard of a guarded TGD. Before we give the formal definition, we need to recall two auxiliary notions. A set Σ of TGDs belongs to the *union* of $\mathbb{G}$ and $\mathbb{S}$, denoted $\mathbb{G}|\mathbb{S}$, if there is a partition $\{\Sigma_g, \Sigma_s\}$ of Σ such that $\Sigma_g \in \mathbb{G}$ and $\Sigma_s \in \mathbb{S}$. A *guard function* of a set $\Sigma \in \mathbb{G}$ is a function g from Σ to the set of atoms in Σ such that: for each $\sigma \in \Sigma$, $g(\sigma)$ is an atom in the body of σ that contains all its body variables. In other words, a guard function specifies which body atom plays the role of the guard.

We can now recall tameness. A set Σ of TGDs is *GS-tame* if $\Sigma \in \mathbb{G}|\mathbb{S}$, and there exists a partition $\{\Sigma_g, \Sigma_s\}$ of Σ, with $\Sigma_g \in \mathbb{G}$ and $\Sigma_s \in \mathbb{S}$, such that: there exists a guard function g of Σ_g such that, for every $\sigma \in \Sigma_g$, the predicate of $g(\sigma)$ does not occur in the head of a TGD of Σ_s. In simple words, Σ is GS-tame if it can be partitioned into a guarded component Σ_g and a sticky component Σ_s, and, in addition, we can choose the guards of Σ_g in such a way that none of their predicates appears in the head of a sticky TGD. The obtained class is denoted $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$, where $|_{\mathsf{t}}$ denotes the fact that we consider the union of $\mathbb{G}$ and $\mathbb{S}$, but with tamed interaction. It is known that $\mathsf{QAns}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$ is 2ExpTime-complete in combined complexity, and PTime-complete in data complexity [Gottlob *et al.*, 2013]. However, the complexity of $\mathsf{FinQAns}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$ is still open. Closing this problem is the main goal of this work.

3 Finite Controllability

A key notion for our analysis is finite controllability [Rosati, 2011]. We say that a class $\mathbb{C}$ of TGDs is *finitely controllable* if, for every database D, set $\Sigma \in \mathbb{C}$ of TGDs, and CQ q, $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma)$. This suggests that, for pinpointing the complexity of $\mathsf{FinQAns}(\mathbb{C})$, where $\mathbb{C}$ is finitely controllable, it suffices to focus on $\mathsf{QAns}(\mathbb{C})$. As already said, the complexity of $\mathsf{QAns}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$ is known. Therefore, our goal is to show that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ is finitely controllable, which will immediately give us the complexity of $\mathsf{FinQAns}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$. To this end, we are going to exploit the fact that both $\mathbb{G}$ and $\mathbb{S}$ are finitely controllable. For $\mathbb{G}$, this has been shown in [Bárány *et al.*, 2014], while for $\mathbb{S}$ in [Gogacz and Marcinkowski, 2017].

Tameness vs. finite controllability. At this point, one may be tempted to think that, since both $\mathbb{G}$ and $\mathbb{S}$ are finitely controllable, it is straightforward to show that also $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ is finitely controllable. In other words, one may claim that tameness preserves finite controllability, no matter which finitely controllable class of TGDs we consider in the place of stickiness. We proceed to show that this is not the case.

It should be clear by now that tameness provides a generic way for combining guardedness with other unguarded classes of TGDs. Thus, in the same way as $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$, we can define $\mathbb{G}|_{\mathsf{t}}\mathbb{C}$, where $\mathbb{C}$ is some arbitrary class of TGDs. We now show that there is a finitely controllable class $\mathbb{C}$ of TGDs such that $\mathbb{G}|_{\mathsf{t}}\mathbb{C}$ is *not* finitely controllable. To achieve this, we are going to consider the class of *full* TGDs, i.e., TGDs without existentially quantified variables; we denote this class by $\mathbb{F}$. It is immediate that $\mathbb{F}$ is finitely controllable, since the chase procedure terminates; see, e.g., [Fagin *et al.*, 2005]. However:

Proposition 1 *$\mathbb{G}|_{\mathsf{t}}\mathbb{F}$ is not finitely controllable.*

Proof (sketch). Let $D = \{P(a, b), R(a, b)\}$, and $\Sigma \in \mathbb{G}|_{\mathsf{t}}\mathbb{F}$ the set of TGDs consisting of

$$\begin{aligned} \sigma_1 &= P(x, y), R(x, y) \rightarrow \exists z P(y, z), R(y, z) \\ \sigma_2 &= R(x, y), R(y, w), R(z, w) \rightarrow R(x, z). \end{aligned}$$

Every $I \in \mathsf{fmods}(D, \Sigma)$ contains an atom of the form $R(t, a)$, where t is some term. However, $\mathsf{chase}(D, \Sigma)$ does not contain such an atom. Therefore, assuming that q is the Boolean CQ $\exists x R(x, a)$, $() \in \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$ and $() \notin \mathsf{cert}(q, D, \Sigma)$; $()$ is the empty tuple. The claim follows. □

The above result suggests that using blindly the fact that both $\mathbb{G}$ and $\mathbb{S}$ are finitely controllable, without exploiting any additional property, it is not enough for establishing that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ is finitely controllable. This indicates that a detailed analysis is required. The next two sections are devoted to performing this analysis, and showing that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ is finitely controllable.

4 The Stratified Case

We first focus on a simpler class of TGDs and show that is finitely controllable. This result is interesting in its own right, but, more crucially, it provides a useful tool that we are going to exploit for showing that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ is finitely controllable.

This simpler class of TGDs is obtained by limiting the interaction between guardedness and stickiness via *stratification*. A set Σ of TGDs is *GS-stratified* if $\Sigma \in \mathbb{G}|\mathbb{S}$, and there exists a partition $\{\Sigma_g, \Sigma_s\}$ of Σ, where $\Sigma_g \in \mathbb{G}$ and $\Sigma_s \in \mathbb{S}$, such that: for every $\sigma \in \Sigma_g$, none of the predicates in the body of σ occurs in the head of a TGD of Σ_s. We write $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ for the obtained class. Clearly, $\mathbb{G}|_{\mathsf{s}}\mathbb{S} \subset \mathbb{G}|_{\mathsf{t}}\mathbb{S}$. We show that:

Theorem 2 $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ *is finitely controllable.*

Before giving the proof, we need to recall that $\mathbb{S}$ is *UCQ-rewritable* [Calì *et al.*, 2012]. This means that, for a set $\Sigma \in \mathbb{S}$ of TGDs and a CQ q, we can always construct a (finite) UCQ q_Σ such that, for every database D, $\mathsf{cert}(q, D, \Sigma) = q_\Sigma(D)$.

Proof. Consider a set $\Sigma \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$. We need to show that, for every database D and CQ q, $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma)$. It is clear that $\mathsf{cert}(q, D, \Sigma) \subseteq \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$; thus, it remains to show that $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) \subseteq \mathsf{cert}(q, D, \Sigma)$.

Let $\{\Sigma_g, \Sigma_s\}$, where $\Sigma_g \in \mathbb{G}$ and $\Sigma_s \in \mathbb{S}$, be a partition of Σ that witnesses the fact that $\Sigma \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$. We are going to show that, for an arbitrary tuple $\bar{c}$ of constants, $\bar{c} \notin \mathsf{cert}(q, D, \Sigma)$ implies $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$. Before we proceed further, let us state an auxiliary lemma, which establishes a useful connection between the models of Σ_g and Σ_s:

Lemma 3 *Fix an instance $I \in \mathsf{mods}(D, \Sigma_g)$. For every $J \in \mathsf{mods}(I, \Sigma_s)$, there is $K \subseteq J$ such that $K \in \mathsf{mods}(D, \Sigma)$.*

We are now ready to complete the proof of Theorem 2:

$\bar{c} \notin \mathsf{cert}(q, D, \Sigma)$
(*by hypothesis*)

$\Rightarrow$ there exists a UCQ q_{Σ_s} such that $\bar{c} \notin \mathsf{cert}(q_{\Sigma_s}, D, \Sigma_g)$
(*by UCQ-rewritability of $\mathbb{S}$, and GS-stratification*)

$\Rightarrow$ $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q_{\Sigma_s}, D, \Sigma_g)$
(*by finite controllability of $\mathbb{G}$*)

$\Rightarrow$ there exists $I \in \mathsf{fmods}(D, \Sigma_g)$ such that $\bar{c} \notin q_{\Sigma_s}(I)$

$\Rightarrow$ $\bar{c} \notin \mathsf{cert}(q, I, \Sigma_s)$
(*since q_{Σ_s} is a UCQ-rewriting of Σ_s and q*)

$\Rightarrow$ $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, I, \Sigma_s)$
(*by finite controllability of $\mathbb{S}$*)

$\Rightarrow$ there exists $J \in \mathsf{fmods}(I, \Sigma_s)$ such that $\bar{c} \notin q(J)$

$\Rightarrow$ there exists $K \subseteq J$ such that $K \in \mathsf{fmods}(D, \Sigma)$
(*by Lemma 3*)

$\Rightarrow$ $\bar{c} \notin q(K)$
(*by monotonicity of CQs*)

$\Rightarrow$ $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$,

as needed. This completes our proof. □

It is interesting to observe that in the above proof, we only use the fact that $\mathbb{G}$ and $\mathbb{S}$ are finitely controllable, and also the fact that $\mathbb{S}$ is UCQ-rewritable. This suggests that actually Theorem 2 can be extended to any class of TGDs $\mathbb{C}_1|_{\mathsf{s}}\mathbb{C}_2$ (defined in the same way as $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$) as long as $\mathbb{C}_1$ and $\mathbb{C}_2$ are finitely controllable, and $\mathbb{C}_2$ is UCQ-rewritable. This result is of independent interest, which can be a useful tool for identifying even more expressive finitely controllable classes.

5 The Tamed Case

We now proceed with our main technical result:

Theorem 4 $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ *is finitely controllable.*

As a corollary to Theorem 4 we obtain the following result:

Corollary 5 $\mathsf{FinQAns}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$ *is complete for* 2ExpTime *in combined complexity, and* PTime *in data complexity*

The rest of this section is devoted to explaining the proof of Theorem 4. Fix $\Sigma \in \mathbb{G}|_{\mathsf{t}}\mathbb{S}$. We need to show that, for every database D and CQ q, $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma)$. Since $\mathsf{cert}(q, D, \Sigma) \subseteq \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$ holds trivially, it remains to show that $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) \subseteq \mathsf{cert}(q, D, \Sigma)$. Our plan of attack for showing the latter containment is as follows. We are going to convert Σ into a set $\Sigma^\star \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$, i.e., into a GS-stratified set of TGDs, and q into a CQ $q^\star$, such that:

1. $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) \subseteq \mathsf{cert}_{\mathsf{fin}}(q^\star, D, \Sigma^\star)$.
2. $\mathsf{cert}(q^\star, D, \Sigma^\star) \subseteq \mathsf{cert}(q, D, \Sigma)$.

It is then easy, by exploiting also the fact that $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ is finitely controllable, to establish the desired containment:

$$\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) \overset{(1)}{\subseteq} \mathsf{cert}_{\mathsf{fin}}(q^\star, D, \Sigma^\star) \overset{\text{(Theorem 2)}}{\subseteq} \mathsf{cert}(q^\star, D, \Sigma^\star) \overset{(2)}{\subseteq} \mathsf{cert}(q, D, \Sigma).$$

In the rest of this section, we explain how $\Sigma^\star$ and $q^\star$ are constructed, and show the containments (1) and (2) above.

Constructing $\Sigma^\star$ and $q^\star$

The construction of the set $\Sigma^\star$ proceeds in three main steps. Let $\{\Sigma_g, \Sigma_s\}$, where $\Sigma_g \in \mathbb{G}$ and $\Sigma_s \in \mathbb{S}$, be the partition of Σ that witnesses the fact that $\Sigma \in \mathbb{G}|_{\mathsf{t}}\mathbb{S}$:

Enriching step. We enrich the set Σ_g in such a way that we obtain a set $\Sigma_g^+ \in \mathbb{G}$ that is powerful enough to entail all the atoms that are entailed by Σ_s and affect Σ_g. This is the most crucial and complex step in the construction of $\Sigma^\star$, for which details are given below.

Copying step. We construct a set Σ_c of "copy" TGDs, which essentially copies each predicate P occurring in Σ_g^+ into a fresh predicate $P^\star$. Formally, for each n-ary predicate P occurring in Σ_g^+, Σ_c contains the TGD:

$$P(x_1, \ldots, x_n) \rightarrow P^\star(x_1, \ldots, x_n)$$

It is clear that $\Sigma_c \in \mathbb{G}$.

Stratification step. We construct the set of TGDs $\Sigma^\star$ by taking the union $\Sigma_g^+ \cup \Sigma_c \cup \Sigma_s^\star$, where $\Sigma_s^\star$ is obtained from Σ_s by renaming each predicate P to $P^\star$.

It is clear that $\Sigma^\star$ falls in $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ since none of the "star" predicates in the head of a TGD of $\Sigma_s^\star \in \mathbb{S}$ occurs in the body of a TGD of $(\Sigma_g^+ \cup \Sigma_c) \in \mathbb{G}$. Now, having $\Sigma^\star$ in place, it should be clear that $q^\star$ is the CQ obtained from q by renaming each predicate P to $P^\star$. Let us now formalize the enriching step.

Formalization of the enriching step. This step relies on the notion of *embedding* of Σ_s into a TGD $\sigma \in \Sigma_g$, which hinges on the fact that $\mathbb{S}$ is UCQ-rewritable. Roughly, we see the body of σ as a CQ q^σ, and we consider the UCQ-rewriting of q^σ with Σ_s. Each partial rewriting gives rise to a new TGD, which in turn can be transformed into linearly many guarded TGDs. The obtained set of guarded TGDs is the embedding of Σ_s into σ, denoted $\mathsf{Embed}(\Sigma_s, \sigma)$. Having this in place, we can then construct the enriched version of Σ_g by embedding Σ_s into every TGD $\sigma \in \Sigma_g$, i.e.,

$$\Sigma_g^+ = \bigcup_{\sigma \in \Sigma_g} \mathsf{Embed}(\Sigma_s, \sigma).$$

It remains to formalize the embedding of Σ_s into σ. Let us assume that the TGD $\sigma \in \Sigma_g$ is of the form $R(\bar{x}, \bar{y}), \varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$, where $R(\bar{x}, \bar{y})$ is the guard of σ provided by the guard function g of Σ_g. Let $q^\sigma(\bar{x})$ be the CQ

$$\exists \bar{y}(R(\bar{x}, \bar{y}) \wedge \varphi(\bar{x}, \bar{y})).$$

By employing the resolution-based query rewriting algorithm from [Gottlob *et al.*, 2014], we can construct a finite UCQ-rewriting of q^σ and Σ_s, denoted $q^\sigma_{\Sigma_s}$, such that, each CQ q of $q^\sigma_{\Sigma_s}$ enjoys two useful syntactic properties: (i) q is *answer-guarded*, i.e., it has an atom that contains all the free variables of q, while the answer-guard is $\rho_q(R(\bar{x}, \bar{y}))$, where ρ_q is some mapping that maps the variables in q to variables in q, and (ii) for each variable x in q, if x does not occur in q^σ, then x occurs only once in q. Having $q^\sigma_{\Sigma_s}$ in place, we define

$$T_\sigma \ := \ \bigcup_{q \in q^\sigma_{\Sigma_s}} \{q \ \rightarrow \ \exists \bar{z} \, \psi(\rho_q(\bar{x}), \bar{z})\},$$

i.e., each CQ q in $q^\sigma_{\Sigma_s}$ becomes the body of a TGD τ_q, while the head of τ_q is the head of the input TGD σ after applying the mapping ρ_q to the variables of $\bar{x}$. Notice that the TGDs of T_σ are not guarded. However, since each q in $q^\sigma_{\Sigma_s}$ is answer-guarded, with $\rho_q(R(\bar{x}, \bar{y}))$ being the answer-guard, τ_q enjoys the following property: the body atom $\rho_q(R(\bar{x}, \bar{y}))$ of τ_q contains all the variables that appear both in the body and in the head of τ_q, while each other variable that appears in the body but not in the head, occurs only once in the body of τ_q. Notice that the set of body variables of a TGD that occur also in the head is known as the *frontier* of the TGD. Thus, in what follows, we refer to the atom $\rho_q(R(\bar{x}, \bar{y}))$ in the body of τ_q as the *frontier-guard* of τ_q. The above property of τ_q allows us to convert it into linearly many guarded TGDs.

Consider a TGD $\tau \in T_\sigma$, and an atom $\alpha = S(x_1, \ldots, x_n)$ in its body with $x_{i_1}, \ldots, x_{i_k}$ being the variables in α that occur also in the frontier-guard of τ. We write $p(\alpha)$ for the atom $S^\alpha_\tau(x_{i_1}, \ldots, x_{i_k})$, where S^α_τ is an auxiliary predicate, and $p(\tau)$ for the TGD obtained from τ after replacing each body atom α of τ, other than the frontier-guard, with $p(\alpha)$. We also denote by σ^α_τ the TGD

$$S(x_1, \ldots, x_n) \ \rightarrow \ S^\alpha_\tau(x_{i_1}, \ldots, x_{i_k}),$$

which simply computes the projection of S over $i_1, \ldots, i_k$. For brevity, we say that an atom in the body of a TGD $\tau \in T_\sigma$ is a *side-atom* if it is not the frontier-guard of τ. We can now easily convert T_σ into a set of guarded TGDs, which gives us the embedding of Σ_s into σ:

$$\mathsf{Embed}(\Sigma_s, \sigma) \ := \ \bigcup_{\tau \in T_\sigma} p(\tau) \ \cup \ \bigcup_{\substack{\tau \in T, \\ \alpha \text{ is a side-atom of } \tau}} \sigma^\alpha_\tau.$$

This completes the formalization of the enriching step.

Establishing the crucial containments

We proceed to show the crucial containments that, together with Theorem 2, allow us to conclude Theorem 4.

Containment 1: $\mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma) \ \subseteq \ \mathsf{cert}_{\mathsf{fin}}(q^\star, D, \Sigma^\star)$

Consider a tuple of constants $\bar{c}$. We proceed to show that if $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q^\star, D, \Sigma^\star)$, then $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$. Our plan of attack for showing this is as follows:

- We isolate a class of finite models of D and $\Sigma^\star$, the so-called *sticky-supported* (or *s-supported*), and show that it forms a *universal finite model set*, i.e., for every $I \in \mathsf{fmods}(D, \Sigma^\star)$, there exists $J \in \mathsf{fmods}(D, \Sigma^\star)$ that is s-supported and can be homomorphically mapped to I.
- Since $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q^\star, D, \Sigma^\star)$, there exists an s-supported $I \in \mathsf{fmods}(D, \Sigma^\star)$ such that $\bar{c} \notin q^\star(I)$. We finally show that the instance J obtained from I by renaming each predicate $P^\star$ into P, is a finite model of D and Σ such that $\bar{c} \notin q(J)$. Thus, $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$.

We proceed to formalize the above description.

Sticky-supported models. The definition of s-supported models relies on the disjunctive chase introduced in [Deutsch and Tannen, 2003], an extension of the chase procedure that is able to deal with disjunctive TGDs (DTGD), i.e., TGDs extended with disjunction in the head. Each disjunctive chase step "branches" out several instances, each satisfying one of the disjuncts of the DTGD that is applied. Thus, the result of the disjunctive chase is, in general, a set of instances (and not a single instance as in the classical chase). For a set of DTGDs $\Sigma_\vee$, we write $\mathsf{dchase}(D, \Sigma_\vee)$ for the result of the disjunctive chase of D w.r.t. $\Sigma_\vee$. Before introducing s-supported models, we need some auxiliary terminology.

We assume that the TGDs of $\Sigma^\star_s$ have only one atom in the head, which contains at most one occurrence of an existentially quantified variable. This does not affect the generality of our proof, since the normalization procedure from [Calì *et al.*, 2012], which converts a TGD into linearly many TGDs with the above properties, preserves finite certain answers. For a (finite) set of nulls $N = \{\bot_1, \ldots, \bot_n\} \subset \mathbf{N}$, where $n \geq 1$, we define $\vee_N(\Sigma^\star_s)$ as the set of DTGDs obtained from $\Sigma^\star_s$ by instantiating the existentially quantified variables using nulls from N. For example, if $N = \{\bot_1, \bot_2\}$, the TGD $R^\star(x, y) \rightarrow \exists z \, S^\star(y, z)$ will be transformed into the DTGD $R^\star(x, y) \rightarrow S^\star(y, \bot_1) \vee S^\star(y, \bot_2)$. For $I \in \mathsf{fmods}(D, \Sigma^\star)$,

$$\begin{aligned} \mathsf{G}(I) &= \{R(\bar{t}) \in I \mid R \text{ is a predicate in } \Sigma\} \\ \mathsf{G}^\star(I) &= \{R^\star(\bar{t}) \mid R(\bar{t}) \in \mathsf{G}(I)\} \\ \mathsf{S}^\star(I) &= I \setminus (\mathsf{G}(I) \cup \mathsf{G}^\star(I)). \end{aligned}$$

We now define s-supported models.

Definition 6 *We say that $I \in \mathsf{fmods}(D, \Sigma^\star)$ is* s-supported *if*

$$\mathsf{S}^\star(I) \ \in \ \mathsf{dchase}(\mathsf{G}^\star(I), \vee_N(\Sigma^\star_s))$$

for a set of nulls $N \subset \mathbf{N}$ such that $|N| \geq |\mathsf{dom}(\mathsf{G}^\star(I))|$ and $N \cap \mathsf{dom}(\mathsf{G}^\star(I)) = \varnothing$. ∎

Intuitively speaking, I is s-supported if $\mathsf{S}^\star(I)$ can be obtained starting from $\mathsf{G}^\star(I)$ and executing the TGDs of $\Sigma^\star_s$ in a chase-like manner, but the existentially quantified variables are satisfied by nulls of $\mathsf{dom}(I)$ that do not occur in $\mathsf{G}^\star(I)$. Notice the crucial difference with the standard chase procedure, where an existentially quantified variable is always satisfied by a "fresh" null, which may lead to an infinite instance. We now show the following crucial result:

Lemma 7 *The set $\{I \in \mathsf{fmods}(D, \Sigma^\star) \mid I \text{ is } s\text{-supported}\}$ is a universal finite model set of D and $\Sigma^\star$.*

Proof (sketch). Fix an arbitrary $I \in \mathsf{fmods}(D, \Sigma^\star)$. We have to show that there exists an instance from $\mathsf{fmods}(D, \Sigma^\star)$ that is s-supported, and it can be homomorphically mapped to I. To this end, we first show that there exists an instance J from $\mathsf{dchase}(\mathsf{G}^\star(I), \vee_N(\Sigma_s^\star))$, with $N = \{\bot_1, \ldots, \bot_{|\mathsf{dom}(I)|}\}$ and $N \cap \mathsf{dom}(\mathsf{G}^\star(I)) = \varnothing$, and a homomorphism h such that $h(J) \subseteq (\mathsf{G}^\star(I) \cup \mathsf{S}^\star(I))$. Let $\mathsf{dom}(\mathsf{G}^\star(I)) = \{t_1, \ldots, t_n\}$, and $\mathsf{dom}(I) \setminus \mathsf{dom}(\mathsf{G}^\star(I)) = \{u_1, \ldots, u_m\}$. We define h as

$$\{t_i \mapsto t_i\}_{1 \leq i \leq n} \cup \{\bot_i \mapsto t_i\}_{1 \leq i \leq n} \cup \{\bot_{n+i} \mapsto u_i\}_{1 \leq i \leq m}.$$

We can show, by induction on the number of disjunctive chase steps, that indeed there exists $J \in \mathsf{dchase}(\mathsf{G}^\star(I), \vee_N(\Sigma_s^\star))$ such that $h(J) \subseteq (\mathsf{G}^\star(I) \cup \mathsf{S}^\star(I))$. Observe now that $(\mathsf{G}(I) \cup J) \in \mathsf{fmods}(D, \Sigma^\star)$, is s-supported, and $h(\mathsf{G}(I) \cup J) \subseteq I$. □

The countermodel of D and Σ. Lemma 7 implies that there exists an s-supported $I \in \mathsf{fmods}(D, \Sigma^\star)$ such that $\bar{c} \notin q^\star(I)$. We claim that $\mathsf{GS}(I) = (\mathsf{G}(I) \cup \mathsf{S}(I))$, where $\mathsf{S}(I)$ is obtained from $\mathsf{S}^\star(I)$ after renaming each predicate $P^\star$ to P, is the desired countermodel of D and Σ. More precisely:

Lemma 8 *$\mathsf{GS}(I) \in \mathsf{fmods}(D, \Sigma)$, and $\bar{c} \notin q(\mathsf{GS}(I))$.*

Proof (sketch). It is clear that $\mathsf{GS}(I) \supseteq D$ and $\mathsf{GS}(I) \models \Sigma_s$. Moreover, since q is $q^\star$ after renaming each predicate $P^\star$ to P, $\bar{c} \notin q(\mathsf{GS}(I))$. Thus, it remains to show that $\mathsf{GS}(I) \models \Sigma_g$. Consider an arbitrary TGD $\sigma \in \Sigma_g$ of the form $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and assume there is a homomorphism h such that $h(\varphi(\bar{x}, \bar{y})) \subseteq \mathsf{GS}(I)$. We need to show that there is h' that agrees with h on $\bar{x}$ such that $h'(\psi(\bar{x}, \bar{z})) \subseteq \mathsf{GS}(I)$. Due to tameness, the guard of σ provided by the guard function of Σ_g is mapped by h to $\mathsf{G}(I)$. This implies that $h(\varphi(\bar{x}, \bar{y}))$ contains only terms from $\mathsf{dom}(\mathsf{G}(I))$. Let us consider $h(\varphi(\bar{x}, \bar{y}))$ as a Boolean CQ, denoted q_h^σ; by abuse of terminology, q_h^σ contains null values from $\mathsf{dom}(\mathsf{G}(I))$, which should be interpreted as constants. We proceed to establish a crucial claim:

Claim 1 $() \in \mathsf{cert}\,(q_h^\sigma, \mathsf{G}(I), \Sigma_s)$.

Proof (sketch). The claim is equivalent to $h(\varphi(\bar{x}, \bar{y})) \subseteq \mathsf{chase}(\mathsf{G}(I), \Sigma_s)$. Let $\varphi^\star(\bar{x}, \bar{y})$ be the conjunction of atoms obtained from $\varphi(\bar{x}, \bar{y})$ by renaming each predicate P to $P^\star$. It is clear that $h(\varphi^\star(\bar{x}, \bar{y})) \subseteq (\mathsf{G}^\star(I) \cup \mathsf{S}^\star(I))$, which implies that there exists $J \in \mathsf{dchase}(\mathsf{G}^\star(I), \vee_N(\Sigma_s^\star))$, where $N = \{\bot_1, \ldots, \bot_{|\mathsf{dom}(I)|}\}$ and $N \cap \mathsf{dom}(\mathsf{G}^\star(I))$ is empty, such that $h(\varphi^\star(\bar{x}, \bar{y})) \subseteq J$. Therefore, there exists a finite sequence of disjunctive chase steps, starting from $\mathsf{G}^\star(I)$ and applying the DTGDs of $\vee_N(\Sigma_s^\star)$, that generates $h(\varphi^\star(\bar{x}, \bar{y}))$. Crucially, none of these steps performs a join over a null of N, i.e., at each step, the variables that occur more than once in the body of the applied DTGD are mapped to terms of $\mathsf{dom}(\mathsf{G}^\star(I))$; otherwise, $\Sigma_s^\star$ is not sticky. This allows us to extract from the above sequence of disjunctive chase steps, a finite sequence of standard chase steps, starting from $\mathsf{G}^\star(I)$ and applying TGDs of $\Sigma_s^\star$, that generates $h(\varphi^\star(\bar{x}, \bar{y}))$. Therefore, $h(\varphi^\star(\bar{x}, \bar{y})) \subseteq \mathsf{chase}(\mathsf{G}^\star(I), \Sigma_s^\star)$, which immediately implies that $h(\varphi(\bar{x}, \bar{y})) \subseteq \mathsf{chase}(\mathsf{G}(I), \Sigma_s)$, as needed. □

The above claim implies that there exists a partial rewriting q of q_h^σ and Σ_s, and a homomorphism μ that extends h, such that $\mu(q) \subseteq \mathsf{G}(I)$. This fact allows us to show that there is $\sigma^+ \in \mathsf{Embed}(\Sigma_s, \sigma)$ such that h maps the body of σ^+ to $\mathsf{G}(I)$. Since $\mathsf{G}(I) \models \mathsf{Embed}(\Sigma_s, \sigma)$, there exists h', which agrees with h on $\bar{x}$, that maps the head of σ^+ to $\mathsf{G}(I)$. Thus, $h'(\psi(\bar{x}, \bar{z})) \subseteq \mathsf{GS}(I)$, and the claim follows. □

It is clear that Lemma 8 implies that $\bar{c} \notin \mathsf{cert}_{\mathsf{fin}}(q, D, \Sigma)$, and containment (1) follows.

Containment 2: $\mathsf{cert}(q^\star, D, \Sigma^\star) \subseteq \mathsf{cert}(q, D, \Sigma)$

Consider a tuple $\bar{c}$ of constants. We proceed to show that, if $\bar{c} \in \mathsf{cert}(q^\star, D, \Sigma^\star)$, then $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$. Although the proof of this implication is not straightforward, it is simpler than the proof of containment (1) since we deal with certain answers (not finite certain answers). This allows us to exploit the soundness of the UCQ-rewritings underlying the construction of $\Sigma^\star$ in a direct way.

Let $\mathsf{chase}(D, \Sigma^\star)^{-\star}$ be the chase instance of D and $\Sigma^\star$ after renaming each predicate $P^\star$ to P. Then:

Lemma 9 *There exists a homomorphism h such that $h(\mathsf{chase}(D, \Sigma^\star)^{-\star}) \subseteq \mathsf{chase}(D, \Sigma)$.*

By hypothesis, $\bar{c} \in q^\star(\mathsf{chase}(D, \Sigma^\star))$, which in turn implies that $\bar{c} \in q(\mathsf{chase}(D, \Sigma^\star)^{-\star})$. By Lemma 9 we get that $\bar{c} \in q(\mathsf{chase}(D, \Sigma))$, and containment (2) follows.

6 Relative Expressive Power

We have seen that, given a set $\Sigma \in \mathbb{G}|_{\mathsf{t}}\mathbb{S}$ and a CQ q, it is always possible to construct a set $\Sigma^\star \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$ and a CQ $q^\star$ such that, for every database D, $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q^\star, D, \Sigma^\star)$. Let us clarify that in this section we focus on (arbitrary) certain answers since $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ and $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ are finitely controllable. At this point, one may be tempted to think that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ and $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ are equally expressive. However, whenever we refer to the expressive power of a class of existential rules, we are usually interested in the so-called program expressive power [Arenas *et al.*, 2014], which aims at the decoupling of the set of TGDs from the CQ. Let us recall the formal definition.

Consider a set Σ of TGDs. The *program expressive power* of Σ, denoted $\mathsf{pep}(\Sigma)$, is the set of triples $(D, q, \bar{c})$, where D is a database, $q(\bar{x})$ is a CQ, and $\bar{c}$ belongs to $\mathsf{cert}(q, D, \Sigma)$. Then, the program expressive power of a class $\mathbb{C}$ is defined as $\mathsf{pep}(\mathbb{C}) = \{\mathsf{pep}(\Sigma) \mid \Sigma \in \mathbb{C}\}$. A class $\mathbb{C}$ is *more expressive* than a class $\mathbb{C}'$, written $\mathbb{C}' \prec \mathbb{C}$, if $\mathsf{pep}(\mathbb{C}') \subset \mathsf{pep}(\mathbb{C})$. It is not difficult to show that $\mathbb{C}$ is more expressive than $\mathbb{C}'$ iff:

1. For every $\Sigma' \in \mathbb{C}'$, there exists $\Sigma \in \mathbb{C}$ such that, for every D and q, $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma')$.
2. There exists $\Sigma \in \mathbb{C}$ such that, for every $\Sigma' \in \mathbb{C}'$, there exists D and q such that $\mathsf{cert}(q, D, \Sigma) \neq \mathsf{cert}(q, D, \Sigma')$.

It is clear that the construction given in the previous section does not show that $\mathbb{G}|_{\mathsf{t}}\mathbb{S}$ and $\mathbb{G}|_{\mathsf{s}}\mathbb{S}$ have the same program expressive power since we modify the CQ (we rename each predicate P to $P^\star$). Even though this is a mild modification, it turned out that is unavoidable. We can show that:

Theorem 10 $\mathbb{G}|_{\mathsf{s}}\mathbb{S} \prec \mathbb{G}|_{\mathsf{t}}\mathbb{S}$.

Proof (sketch). We need to show $\mathsf{pep}(\mathbb{G}|_{\mathsf{s}}\mathbb{S}) \subset \mathsf{pep}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$. Clearly, $\mathsf{pep}(\mathbb{G}|_{\mathsf{s}}\mathbb{S}) \subseteq \mathsf{pep}(\mathbb{G}|_{\mathsf{t}}\mathbb{S})$ since $\mathbb{G}|_{\mathsf{s}}\mathbb{S} \subset \mathbb{G}|_{\mathsf{t}}\mathbb{S}$. It remains to show that there exists $\Sigma \in \mathbb{G}|_{\mathsf{t}}\mathbb{S}$ such that $\mathsf{pep}(\Sigma) \notin$

$\mathsf{pep}(\mathbb{G}|_{\mathsf{s}}\mathbb{S})$, or, equivalently, for every $\Sigma' \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$, there exists D and q such that $\mathsf{cert}(q, D, \Sigma) \neq \mathsf{cert}(q, D, \Sigma')$. Let Σ be the set consisting of the TGDs

$$\begin{aligned} P(x, y, z), R(x, y) &\rightarrow R(y, z), S(y), S(z) \\ S(x), S(y) &\rightarrow R(x, y). \end{aligned}$$

Assume there exists $\Sigma' \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$ such that $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma')$ for every D and q. If $\{\Sigma'_g, \Sigma'_s\}$, where $\Sigma'_g \in \mathbb{G}$ and $\Sigma'_s \in \mathbb{S}$, is the partition of Σ' that witnesses the fact that $\Sigma' \in \mathbb{G}|_{\mathsf{s}}\mathbb{S}$, we can show that $\Sigma'_s \notin \mathbb{S}$, which is a contradiction. To show this, we employ the database

$$D = \{R(c_1, c_2), P(c_1, c_2, c_3), P(c_2, c_3, c_4), P(c_3, c_4, c_5)\}$$

and the CQ $q = R(c_2, c_5)$. The proof exploits the fact that, during the chase, guarded TGDs cannot propagate in an atom two constants that are not already together in a database atom, which implies that $R(c_2, c_5)$ is generated by a TGD of Σ'_s. We also use the fact that $\mathbb{S}$ is UCQ-rewritable, and, in particular, the fact that in each partial rewriting of q and Σ'_s, "fresh" variables occur only once due to stickiness. □

7 Future Work

Although the problem of checking whether a tuple is a certain answer has attracted considerable attention, the problem of checking whether a tuple is *not* a certain answer has received far less attention. By exploiting the ideas underlying s-supported models, we are planning to devise algorithms that will try to refute a candidate certain answer, assuming that the input set of TGDs is finitely controllable, by first exploring finite models with only one null, and then incrementally explore finite models with more nulls. If a candidate is not certain, then this procedure will recognize it since it explores a universal finite model set. Such an algorithm can be executed in parallel with the chase procedure, which can be used for checking whether a candidate answer is entailed. Therefore, we obtain a sound and complete procedure for query answering under finitely controllable classes of TGDs. We believe that such a parallel procedure can lead to practical algorithms since in real-life scenarios the entailment or refutation of a candidate answer can be recognized quickly without introducing a prohibitively large number of nulls.

Acknowledgements

Gottlob and Pieris have been supported by the EPSRC Programme Grant "VADA" EP/M025268/. Manna has been partially supported by the Italian Ministry for Economic Development under PON project "S2BDW" (n. F/050389/01-03/X32), and by Regione Calabria under POR project "DLV Large Scale" (CUP J28C17000220006).

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Amendola *et al.*, 2017] Giovanni Amendola, Nicola Leone, and Marco Manna. Finite model reasoning over existential rules. *TPLP*, 17(5-6):726–743, 2017.

[Arenas *et al.*, 2014] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. Expressive languages for querying the semantic web. In *PODS*, pages 14–26, 2014.

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[Bárány *et al.*, 2014] Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014.

[Beeri and Vardi, 1981] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *ICALP*, pages 73–85, 1981.

[Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[Calì *et al.*, 2013] Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[Calvanese, 1996] Diego Calvanese. Finite model reasoning in description logics. In *KR*, pages 292–303, 1996.

[Deutsch and Tannen, 2003] Alin Deutsch and Val Tannen. Reformulation of XML queries and constraints. In *ICDT*, pages 225–241, 2003.

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

[Gogacz and Marcinkowski, 2017] Tomasz Gogacz and Jerzy Marcinkowski. Converging to the chase - A tool for finite controllability. *J. Comput. Syst. Sci.*, 83(1):180–206, 2017.

[Gottlob *et al.*, 2013] Georg Gottlob, Marco Manna, and Andreas Pieris. Combining decidability paradigms for existential rules. *TPLP*, 13(4-5):877–892, 2013.

[Gottlob *et al.*, 2014] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Query rewriting and optimization for ontological databases. *ACM Trans. Database Syst.*, 39(3):25:1–25:46, 2014.

[Ibáñez-García *et al.*, 2014] Yazmin Angélica Ibáñez-García, Carsten Lutz, and Thomas Schneider. Finite model reasoning in horn description logics. In *KR*, 2014.

[Patel-Schneider and Horrocks, 2007] Peter F. Patel-Schneider and Ian Horrocks. A comparison of two modelling paradigms in the semantic web. *J. Web Semantics*, 5(4):240–250, 2007.

[Rosati, 2008] Riccardo Rosati. Finite model reasoning in DL-Lite. In *ESWC*, pages 215–229, 2008.

[Rosati, 2011] Riccardo Rosati. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.*, 77(3):572–594, 2011.