Multi-head Guarded Existential Rules Over Fixed Signatures

Georg Gottlob[1] , **Marco Manna**[2] , **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

Guarded existential rules form a robust rule-based language for modelling ontologies. The central problem of ontology-based query answering, as well as the notion of polynomial combined rewritability, have been extensively studied during the last years for this formalism. However, the relevant setting where the underlying signature is fixed is far from being well-understood. All the existing results on ontology-based query answering and polynomial combined rewritability implicitly assume rule-heads with one atom, whereas existential rules in real ontologies are typically coming with multi-heads consisting of several atoms. We aim to fill this gap.

1 Introduction

In ontology-based query answering (OBQA), ontologies are used to enrich incomplete data with domain knowledge, enabling more complete answers to queries, typically conjunctive queries (CQs). A notable range of ontology formalisms for OBQA, which vary in syntax, expressivity and complexity, has been developed during the last decade. Two prominent families of languages, obtained from this extensive effort, are description logics (DL) (Baader et al. 2017), and existential rules (a.k.a. tuple-generating dependencies and Datalog$^{\pm}$ rules) (Baget et al. 2011; Calì et al. 2010). Many of the existing DL-based and rule-based ontology languages guarantee good computational and model-theoretic properties by posing restrictions on the use of quantifiers. They are essentially defined through the relativisation of quantifiers by atomic formulas, similar to the guarded fragment of first-order logic (Andréka, van Benthem, and Németi 1998). It is generally agreed that guardedness is a paradigm that leads to reasonably expressive and robust ontology languages.

The main rule-based ontology language that originated from this paradigm, which is the main concern of this work, is the class of guarded existential rules (Calì, Gottlob, and Kifer 2013). It consists of sets of first-order sentences 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 (the body) and ψ (the head) are conjunctions of relational atoms, and ϕ has an atom that contains all the universally quantified variables.

Guarded Existential Rules. The problem of OBQA has been rigorously studied for guarded existential rules during the last years. The two main research directions were:

1. to pinpoint its computational complexity, and
2. to understand whether it can be solved by relying on standard database technology via query rewriting.

It is fair to claim that the above aspects of the problem are rather well-understood. Concerning the first one, OBQA for guarded existential rules is complete for PTIME in data complexity (i.e., when the ontology and the query are fixed), for NP when the underlying signature (or schema) is fixed with the additional implicit assumption that rule-heads have only one atom (further details about the latter assumption are given below), for EXPTIME over signatures of fixed arity, and for 2EXPTIME in combined complexity (Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Kifer 2013).

Concerning query rewriting, the PTIME-hardness in data complexity immediately implies that the problem in question is not first-order rewritable. On the other hand, we know that it is Datalog rewritable (Bárány, Benedikt, and ten Cate 2013; Gottlob, Rudolph, and Simkus 2014). At this point, let us stress that the above results refer to the pure approach to query rewriting, where the rewriting phase does not depend on any database. With the aim of obtaining first-order rewritings, and thus being able to use a standard relational engine for OBQA purposes, the work (Gottlob, Manna, and Pieris 2014) adopted the more refined approach to query rewriting known as the combined approach, originally introduced in the context of DLs (Lutz, Toman, and Wolter 2009), which allows us to rewrite also the database in a query-independent way. It was shown that indeed OBQA for guarded existential rules is combined first-order rewritable. The really interesting result of that work, though, is that in the fixed signature case the rewriting process takes only polynomial time with the additional implicit assumption that rule-heads mention only one atom. Note that beyond fixed signatures, the combined rewriting process is unlikely to be polynomial.

Multi-head Rules and Fixed Signatures. Despite the thorough analysis of the OBQA problem for guarded existential rules, there is a striking gap, which we reveal below, that passed unnoticed until recently when it was brought to our attention by a colleague of ours (Benedikt 2018).

Recall that an existential rule is typically defined as a first-order sentence $\forall\bar{x}\forall\bar{y}\big(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z})\big)$, where ϕ and ψ are conjunctions of relational atoms. For OBQA purposes, the fact that in the head we can have a conjunction of atoms is usually seen as syntactic sugar. The reason is because we can always convert a set Σ of existential rules into a set Σ_1 of existential rules with only atom in the head such that Σ and Σ_1, although not logically equivalent, are equivalent for OBQA. This, in fact, relies on a very simple transformation that replaces each existential rule $\sigma \in \Sigma$ of the form

$$\forall \bar{x} \forall \bar{y} \big(\phi(\bar{x}, \bar{y}) \ \rightarrow \ \exists \bar{z} \, R_1(\bar{x}_1, \bar{z}_1) \wedge \cdots \wedge R_n(\bar{x}_n, \bar{z}_n)\big),$$

where $\bar{x}_i \subseteq \bar{x}$ and $\bar{z}_i \subseteq \bar{z}$, with the set of existential rules

$$\begin{array}{rcl} \forall \bar{x} \forall \bar{y} \big(\phi(\bar{x}, \bar{y}) & \rightarrow & \exists \bar{z} \, \mathrm{Aux}_\sigma(\bar{x}, \bar{z})\big) \\ \forall \bar{x} \forall \bar{z} (\mathrm{Aux}_\sigma(\bar{x}, \bar{z}) & \rightarrow & R_i(\bar{x}_i, \bar{z}_i)), \text{ for } i \in \{1, \ldots, n\}, \end{array}$$

with Aux_σ being a fresh relation not occurring in Σ. Notice further that if Σ is guarded, then also Σ_1 is guarded.

Due to the above transformation, all the known complexity results on OBQA for guarded existential rules, as well as the result on polynomial combined first-order rewritability, have been shown for existential rules with only one atom in the head. The reason was purely technical, that is, to simplify the technical definitions and proofs. Although this simplifying assumption can be made, in general, without affecting the generality of the results, this is *not* true in the relevant setting of fixed signatures. This is because, even if we start from a set of guarded existential rules over a fixed signature, the obtained set of single-head guarded rules after the transformation mentions an unbounded number of new relations each of unbounded arity (see the auxiliary predicates).

As mentioned above, fixing the signature is a relevant setting, which is close in spirit to data complexity. Typically, the size of the signature is much smaller than the size of the database, and therefore, it can be productively assumed to be fixed. Furthermore, since the typical purpose of an ontology is to model a certain domain of interest, the schema is actually predetermined by the application domain in question, which is another justification for considering the signature fixed. The above, together with the fact that it is more convenient to design ontologies using multi-head existential rules, it suggests that the gap in the analysis of OBQA for guarded existential rules described above is a fundamental one, which undoubtedly deserves our attention.

Research Challenges. From the above discussion, we are left with the following key questions concerning the OBQA problem for guarded existential rules over fixed signatures:

1. Does the NP-completeness result for single-head rules carry over to multi-head rules?
2. Does the polynomial combined first-order rewritability result for single-head rules carry over to multi-head rules?

The goal of this work is to provide answers to the above questions, which will advance our knowledge on the important problem of OBQA for guarded existential rules.

Contrary to what one might think, the fact that we need to directly deal with multi-heads causes non-trivial complications, which were not an issue for single-head TGDs, that require novel ideas. To better understand the different nature of the two settings, let us stress that in the case of single-head guarded existential rules, by fixing the underlying signature we implicitly fix the whole ontology. The number of different guarded non-isomorphic rule-bodies that can be formed over a fixed signature is constant, which in turn implies that we can only have a constant number of different single-head guarded rules (up to variable renaming). This is far from being true for multi-head guarded existential rules since we can have an unbounded number of different rules with the same guarded body due to the unguarded multi-heads.

Our Results. The main results of this work are as follows:

1. Before delving into the analysis that will provide answers to our main research questions, we ask ourselves whether multi-head guarded rules are strictly more expressive than single-head ones. We show that, for each schema $\mathbf{S}$ (with at least one binary relation), there exists a set of multi-head guarded existential rules over $\mathbf{S}$ that cannot be equivalently rewritten as a set of single-head guarded existential rules over $\mathbf{S}$ for OBQA purposes. In simple words, if we are not allowed to introduce new relation symbols, multi-heads are not merely syntactic sugar but come with additional expressive power.
2. We then show that OBQA for multi-head guarded existential rules over fixed signatures is NP-complete. The non-trivial result is the upper bound; the lower bound is inherited from CQ evaluation. Towards showing the upper bound, we obtain several results of independent interest: (i) instance checking for multi-head guarded existential rules over fixed signatures is in PTIME, (ii) OBQA for multi-head guarded existential rules can be polynomially reduced to multi-head linear existential rules (i.e., rules with only one atom in the body) without increasing the arity of the signature, and (iii) OBQA for multi-head linear existential rules is NP-complete for signatures of fixed arity. The above are known for single-head rules, but cannot be directly transferred to multi-head ones.
3. Finally, by exploiting the machinery introduced for establishing the NP upper bound, we can show that OBQA for multi-head guarded existential rules over fixed signatures is polynomially combined first-order rewritable; in fact, the target query language is existential positive first-order queries. Towards this end, we also show the same for multi-head linear rules over signatures of fixed arity. The latter is known for single-head linear rules, but it cannot be straightforwardly transferred to multi-head ones.

2 Preliminaries

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

Relational Instances. A *(relational) schema* $\mathbf{S}$ is a finite set of relation symbols (or predicates) with associated arity. We write $\mathsf{ar}(R)$ for the arity of a predicate R, and $\mathsf{ar}(\mathbf{S})$ for the arity of $\mathbf{S}$, i.e., the number $\max_{R \in \mathbf{S}}\{\mathsf{ar}(R)\}$. An *atom* over $\mathbf{S}$ is an expression of the form $R(\bar{t})$, where $R \in \mathbf{S}$ and $\bar{t}$ is a tuple of terms. An *instance* over $\mathbf{S}$ is a (possibly infinite) set of atoms over $\mathbf{S}$ with constants and nulls, while a *database* over $\mathbf{S}$ is a finite instance over $\mathbf{S}$ with only constants. Given a set of of terms T, let $\mathsf{B}(T, \mathbf{S})$ be the set of atoms that can be formed using terms of T and predicates of $\mathbf{S}$. We write

$\mathsf{dom}(I)$ for the set of terms in an instance I; this notation naturally extends to sets of atoms.

Homomorphisms. A *homomorphism* from a set of atoms A to a set of atoms B is a function $h : \mathsf{dom}(A) \rightarrow \mathsf{dom}(B)$ that is the identity on $\mathbf{C}$ with $R(h(\bar{t})) \in B$ for every $R(\bar{t}) \in A$. We write $A \rightarrow B$ for the fact that there is a homomorphism from A to B. For a set of terms S, we say that A and B are S*-isomorphic*, denoted $A \simeq_S B$, if there is a 1-1 homomorphism from A to B such that h^{-1} maps B to A.

Queries. A *conjunctive query* (CQ) over a schema $\mathbf{S}$ 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*. 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 the set of tuples $\bar{c} \in \mathbf{C}^{|\bar{x}|}$ such that $q(\bar{c}) \rightarrow I$; we may treat a conjunction of atoms as a set.

Tuple-generating dependencies. A *tuple-generating dependency* (TGD) σ over a schema $\mathbf{S}$ is a first-order 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 and ψ are (non-empty) conjunctions of atoms over $\mathbf{S}$ that mention only variables. We write σ 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 σ, denoted $\mathsf{body}(\sigma)$ and $\mathsf{head}(\sigma)$, respectively. An instance I satisfies σ, written $I \models \sigma$, if, whenever $\phi(\bar{x}, \bar{y}) \rightarrow I$ via h, then $\psi(\bar{x}, \bar{z}) \rightarrow I$ via h' that agrees with h on $\bar{x}$. 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 family of all finite sets of TGDs. A class $\mathbb{C}$ of TGDs is a subset of $\mathbb{TGD}$. We write $\mathbb{C}[\mathbf{S}]$ for the class $\{\Sigma \in \mathbb{C} \mid \Sigma \text{ is over } \mathbf{S}\}$.

Ontological Query Answering. Given a database D and a set Σ of TGDs, a *model* of D and Σ is a (possibly infinite) instance $I \supseteq D$ such that $I \models \Sigma$. We write $\mathsf{mods}(D, \Sigma)$ for the set of models of D and Σ. The *certain answers* to a CQ q w.r.t. D and Σ is the set $\mathsf{cert}(q, D, \Sigma) = \bigcap_{I \in \mathsf{mods}(D,\Sigma)} q(I)$. The main problem that we study in this work follows:

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

We are interested in the complexity of the problem when the schema is fixed, i.e., the complexity of $\mathsf{OQA}(\mathbb{C}[\mathbf{S}])$ for a fixed schema $\mathbf{S}$. We adopt the usual convention that when we talk about the complexity of $\mathsf{OQA}(\mathbb{C})$ for fixed schemas, we say that it is $\mathcal{C}$-complete for a complexity class $\mathcal{C}$ if, for each schema $\mathbf{S}$, $\mathsf{OQA}(\mathbb{C}[\mathbf{S}])$ is in $\mathcal{C}$, and there is a schema $\mathbf{S}$ such that $\mathsf{OQA}(\mathbb{C}[\mathbf{S}])$ is $\mathcal{C}$-hard. Analogously, we can talk about the complexity of $\mathsf{OQA}(\mathbb{C})$ for schemas of fixed arity.

Guardedness and Linearity. A TGD σ is *guarded* if there is an atom in $\mathsf{body}(\sigma)$, called *guard*, that contains all the body variables. By convention, the leftmost body atom of a guarded TGD σ is the guard, denoted $\mathsf{guard}(\sigma)$, and all the other atoms are the *side* atoms of σ. Let $\mathbb{G}$ (resp., $\mathbb{G}_1$) be the class of finite sets of guarded TGDs (resp., with one head atom). A subclass of $\mathbb{G}$, which is crucial for our work, is the class of *linear* TGDs, denoted $\mathbb{L}$, which collects all the sets of TGDs with one body atom. We also write $\mathbb{L}_1$ for the class of sets of linear TGDs with one head atom.

The Chase Procedure. The chase procedure is a useful tool when reasoning with TGDs. Let us first define a single chase application. A *trigger* for a set Σ of TGDs on an instance I is a pair (σ, h), where $\sigma \in \Sigma$ and h is a homomorphism from $\mathsf{body}(\sigma)$ to I. An *application* of (σ, h) to I returns an instance $J = I \cup h'(\mathsf{head}(\sigma))$, where h' extends h in such a way that (i) for each existentially quantified variable z of σ, $h'(z) \in \mathbf{N} \setminus \mathsf{dom}(I)$, and (ii) for distinct existentially quantified variables z and w of σ, $h'(z) \neq h'(w)$. Such a trigger application is denoted as $I\langle\sigma, h\rangle J$.

The main idea of the chase is, starting from a database D, to exhaustively apply distinct triggers for the given set Σ of TGDs on the instance constructed so far, and keep doing this until a fixpoint is reached. This is formalized as follows:

- A finite sequence of instances $(I_i)_{0 \leq i \leq n}$, with $D = I_0$ and $n \geq 0$, is a *chase derivation* of D w.r.t. Σ if: (i) for each $0 \leq i < n$, there is a trigger (σ, h) for Σ on I_i such that $I_i\langle\sigma, h\rangle I_{i+1}$; (ii) for each $0 \leq i < j < n$, assuming that $I_i\langle\sigma_i, h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j, h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j$ implies $h_i \neq h_j$, and (iii) there is no trigger (σ, h) for Σ on I_n such that $(\sigma, h) \notin \{(\sigma_i, h_i)\}_{0 \leq i < n}$. In this case, the result of the chase is the (finite) instance I_n.
- An infinite sequence of instances $(I_i)_{i \geq 0}$ is a *chase derivation* of D w.r.t. Σ if: (i) for each $i \geq 0$, there exists a trigger (σ, h) for Σ on I_i such that $I_i\langle\sigma, h\rangle I_{i+1}$; (ii) for each $i, j > 0$ such that $i \neq j$, assuming that $I_i\langle\sigma_i, h_i\rangle I_{i+1}$ and $I_j\langle\sigma_j, h_j\rangle I_{j+1}$, $\sigma_i = \sigma_j$ implies $h_i \neq h_j$; and (iii) for each $i \geq 0$, and for every trigger (σ, h) for Σ on I_i, there exists $j \geq i$ such that $I_j\langle\sigma, h\rangle I_{j+1}$; this is the *fairness condition*, and guarantees that all the triggers eventually will be applied. The result of the chase is $\bigcup_{i \geq 0} I_i$.

We write $\mathsf{chase}_\delta(D, \Sigma)$ for the result of a (finite or infinite) chase derivation δ of D w.r.t. Σ. Here is the key property:

Proposition 1 *Consider a database D, a set Σ of TGDs, and a chase derivation δ of D w.r.t. Σ. For every instance $I \in \mathsf{mods}(D, \Sigma)$, it holds that $\mathsf{chase}_\delta(D, \Sigma) \rightarrow I$.*

In the rest of the paper, we will silently use a crucial consequence of the above result, namely a tuple $\bar{c}$ belongs to $\mathsf{cert}(q, D, \Sigma)$ iff there exists a prefix $(I_i)_{0 \leq i \leq n}$, for $n \geq 0$, of a chase derivation of D w.r.t. Σ such that $\bar{c} \in q(I_n)$.

We conclude our discussion by defining a useful relation over the result of chase derivations. Consider a chase derivation $\delta = (I_i)_{i \geq 0}$ of a database D w.r.t. a set Σ of TGDs, and assume that for each $i \geq 0$, $I_i\langle\sigma_i, h_i\rangle I_{i+1}$, i.e., I_{i+1} is obtained from I_i via the application of the trigger (σ_i, h_i) to I_i. The *parent relation* of δ, denoted $\prec_\delta^p$, is a binary relation over $\mathsf{chase}_\delta(D, \Sigma)$ such that $\alpha \prec_\delta^p \beta$ iff there is $i \geq 0$ such that $\alpha \in h_i(\mathsf{body}(\sigma_i))$ and $\beta \in I_{i+1} \setminus I_i$. Let $\prec_\delta^{p,+}$ be the transitive closure of $\prec_\delta^p$. Note that $\prec_\delta^p$ is acyclic, i.e., it forms a directed acyclic graph, whereas in the case of linear TGDs it forms a forest, with the atoms of D being the roots.

3 Relative Expressiveness

We start with the question concerning the expressive power of multi-head guarded TGDs. We adopt the notion of program expressive power, introduced in (Arenas, Gottlob, and Pieris 2018), which captures the essence of our question: having available a schema $\mathbf{S}$, is it the case that a set $\Sigma \in \mathbb{G}[\mathbf{S}]$ can be rewritten as a set $\Sigma' \in \mathbb{G}_1[\mathbf{S}]$ that can answer the exact same CQs? As we shall see below, the answer to this question is negative. Let us first properly introduce the adopted notion of program expressive power.[1]

The *expressive power* of a set Σ of TGDs over a schema $\mathbf{S}$, denoted $\mathsf{ep}(\Sigma)$, is the set of triples $(D, q(\bar{x}), \bar{c})$, where D is a database over $\mathbf{S}$, $q(\bar{x})$ is a CQ over $\mathbf{S}$, and $\bar{c} \in \mathsf{dom}(D)^{|\bar{x}|}$, such that $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$. The *program expressive power* of a class of TGDs $\mathbb{C}$ is defined as the family of triples

$$\mathsf{pep}(\mathbb{C}) \;=\; \{\mathsf{ep}(\Sigma) \mid \Sigma \in \mathbb{C}\}.$$

For two classes of TGDs $\mathbb{C}_1, \mathbb{C}_2$, we say that $\mathbb{C}_2$ is *more expressive* than $\mathbb{C}_1$, written $\mathbb{C}_1 \leq \mathbb{C}_2$, if $\mathsf{pep}(\mathbb{C}_1) \subseteq \mathsf{pep}(\mathbb{C}_2)$. In addition, we say that $\mathbb{C}_2$ is *strictly more expressive* than $\mathbb{C}_2$, written $\mathbb{C}_1 < \mathbb{C}_2$, if $\mathbb{C}_1 \leq \mathbb{C}_2$ and $\mathbb{C}_2 \not\leq \mathbb{C}_1$. The next result provides a definite answer to our first question:

Theorem 2 *For a schema* $\mathbf{S}$ *with* $\mathsf{ar}(\mathbf{S}) > 1$, $\mathbb{G}_1[\mathbf{S}] < \mathbb{G}[\mathbf{S}]$.

It is not difficult to verify that to establish Theorem 2 (in fact, $\mathbb{G}[\mathbf{S}] \not\leq \mathbb{G}_1[\mathbf{S}]$ since $\mathbb{G}_1[\mathbf{S}] \leq \mathbb{G}[\mathbf{S}]$ holds trivially), we simply need to exhibit a set of TGDs $\Sigma \in \mathbb{G}[\mathbf{S}]$ such that, for every $\Sigma_1 \in \mathbb{G}_1[\mathbf{S}]$, there exist a database D and a CQ q, both over the schema $\mathbf{S}$, with $\mathsf{cert}(q, D, \Sigma) \neq \mathsf{cert}(q, D, \Sigma_1)$. By assumption, $\mathbf{S}$ contains at least one binary predicate R. We define Σ as the singleton set consisting of the TGD

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

Towards a contradiction, assume that there exists a set $\Sigma_1 \in \mathbb{G}_1[\mathbf{S}]$ such that, for every database D and CQ q, both over $\mathbf{S}$, it holds that $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D, \Sigma_1)$. Let $D_R = \{R(a, b)\}$, and consider also the Boolean CQs

$$\begin{aligned}
\mathsf{line}_2 &= \exists x \exists y \exists z\, (R(x, y) \wedge R(y, z)) \\
\mathsf{line}_3 &= \exists x \exists y \exists z \exists w\, (R(x, y) \wedge R(y, z) \wedge R(z, w)),
\end{aligned}$$

and for each $P \in \mathbf{S} \setminus \{R\}$, the Boolean CQ

$$\mathsf{other}_P \;=\; \exists x_1 \cdots \exists x_{\mathsf{ar}(P)}\, P(x_1, \ldots, x_{\mathsf{ar}(P)}).$$

Clearly, $\mathsf{cert}(\mathsf{line}_2, D_R, \Sigma) \neq \varnothing$, while $\mathsf{cert}(\mathsf{line}_3, D_R, \Sigma)$ and $\mathsf{cert}(\mathsf{other}_P, D_R, \Sigma)$, for each $P \in \mathbf{S} \setminus \{R\}$, are empty. Hence, the same holds for Σ_1. It is an easy exercise to verify that the following hold; otherwise, it will contradict one of the above statements concerning certain answers under Σ_1:

Fact 3 *For every chase derivation* δ *of* D_R *w.r.t.* Σ_1*:*

1. *There is no atom of the form* $R(t, t)$ *in* $\mathsf{chase}_\delta(D_R, \Sigma_1)$.
2. *There are no atoms of the form* $R(t, t'), R(t', t)$*, with* $t \neq t'$*, in* $\mathsf{chase}_\delta(D_R, \Sigma_1)$.
3. *There is no atom of the form* $P(\bar{t})$*, for a predicate* $P \in \mathbf{S}$ *such that* $P \neq R$*, in* $\mathsf{chase}_\delta(D_R, \Sigma_1)$.

[1]A set of TGDs is also called program, influenced by logic programming, and hence the name program expressive power.

The above, together with the fact that Σ_1 is guarded, allow us to conclude that, for every chase derivation $\delta = (I_i)_{i \geq 0}$ of D w.r.t. Σ with $I_i \langle \sigma_i, h_i \rangle I_{i+1}$, σ_i has the form $R(x, y) \rightarrow \psi$, where ψ is one of the following atomic formulas:

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

But this implies that $\mathsf{cert}(\mathsf{line}_2, D_R, \Sigma_1)$ is empty, which is a contradiction, and the claim follows.

4 Complexity Analysis

$\mathsf{OQA}(\mathbb{G}_1)$ is NP-complete for fixed schemas (Calì, Gottlob, and Kifer 2013). The goal of this section is to show that this result carries over to multi-head TGDs:

Theorem 4 $\mathsf{OQA}(\mathbb{G})$ *is* NP*-complete for fixed schemas.*

The lower bound is inherited from CQ evaluation, which is NP-hard even for schemas with one binary relation. The upper bound, on the other hand, is a rather non-trivial result, which cannot be immediately obtained from the fact that $\mathsf{OQA}(\mathbb{G}_1)$ is in NP. Our proof proceeds in three steps:

1. We first focus on the simpler problem of instance checking, and show that it is in PTIME for fixed schemas.
2. We reduce $\mathsf{OQA}(\mathbb{G}[\mathbf{S}])$, for a schema $\mathbf{S}$, to $\mathsf{OQA}(\mathbb{L}[\mathbf{S}'])$, where $\mathbf{S}' \supseteq \mathbf{S}$ with $\mathsf{ar}(\mathbf{S}) = \mathsf{ar}(\mathbf{S}')$, and we further show, by exploiting point (1), that the given reduction is a polynomial time reduction whenever $\mathbf{S}$ is fixed.
3. We then show that $\mathsf{OQA}(\mathbb{L})$ is NP-complete for schemas of fixed arity. This result, which is of independent interest, together with the above reduction, establishes the desired NP upper bound claimed in Theorem 4.

4.1 Step 1: Instance Checking

The problem that we study in this section, which is actually a special case of ontological query answering, is as follows:

PROBLEM :	$\mathsf{IC}(\mathbb{C})$
INPUT :	A database D, a set $\Sigma \in \mathbb{C}$ of TGDs, a CQ $R(\bar{x})$, and $\bar{c} \in \mathsf{dom}(D)^{\mathsf{ar}(R)}$.
QUESTION :	Does $\bar{c} \in \mathsf{cert}(R(\bar{x}), D, \Sigma)$?

The goal is to show the following complexity result:

Theorem 5 $\mathsf{IC}(\mathbb{G})$ *is in* PTIME *for fixed schemas.*

$\mathsf{IC}(\mathbb{G}_1)$ is in PTIME for fixed schemas (Calì, Gottlob, and Kifer 2013), which has been shown via a sophisticated alternating algorithm that uses logarithmic space. However, this result cannot be straightforwardly transferred to multi-head TGDs. We proceed to establish Theorem 5 via a novel alternating algorithm that is designed to operate on multi-head TGDs. But first we need to introduce some auxiliary results.

Guarded Types. A key notion when reasoning with guarded TGDs is the type of an atom. Consider a database D, and a set $\Sigma \in \mathbb{G}$ of TGDs. Let $\delta = (I_i)_{i \geq 0}$ be a (finite or infinite) chase derivation of D w.r.t. Σ with $I_i \langle \sigma_i, h_i \rangle I_{i+1}$. Given an atom $\alpha \in \mathsf{chase}_\delta(D, \Sigma)$, its *$\delta$-type (w.r.t. D and Σ)*, denoted $\mathsf{type}_\delta(\alpha)$, is the set $\{\beta \in \mathsf{chase}_\delta(D, \Sigma) \mid \mathsf{dom}(\beta) \subseteq \mathsf{dom}(\alpha)\}$, i.e., all the atoms in the result of δ that share terms with α. The key property of the type, which has been shown in (Calì, Gottlob, and Kifer 2013) for single-head TGDs, and it can be easily extended to multi-head TGDs, it roughly states that the set of atoms in $\mathsf{chase}_\delta(D,\Sigma)$ that can be derived from α (used as a guard) is determined by the δ-type of α. To make this precise we need some auxiliary notions.

The *guarded parent relation* of δ, denoted $\prec_\delta^{gp}$, is a binary relation over $\mathsf{chase}_\delta(D,\Sigma)$ such that $\alpha \prec_\delta^{gp} \beta$ iff there exists $i \geq 0$ with $\alpha = h_i(\mathsf{guard}(\sigma_i))$ and $\beta \in I_{i+1} \setminus I_i$. Let $\prec_\delta^{gp,+}$ be the transitive closure of $\prec_\delta^{gp}$. Note that $\prec_\delta^{gp}$ forms a forest with the atoms of D being the roots. We can now define the notion of projection of δ, which will allow us to state the key property of the type. Consider an atom $\alpha \in \mathsf{chase}_\delta(D,\Sigma)$, and let $(i_j)_{j>0}$ be the sequence of indices, with $0 \leq i_1 < i_2 < i_3 < \cdots$ such that, for each $\ell \geq 0$, $\ell \in \{i_j\}_{j>0}$ iff $h_\ell(\mathsf{guard}(\sigma_\ell)) = \alpha$ or $\alpha \prec_\delta^{gp,+} h_\ell(\mathsf{guard}(\sigma_\ell))$. Simply stated, $(i_j)_{j>0}$ collects all the applications in δ, in ascending order, that use α or a $\prec_\delta^{gp}$-descendant of α as the guard. The *α-projection of* δ, denoted $\delta[\alpha]$, is the sequence of instances $(J_i)_{i\geq 0}$, with $J_0 = \mathsf{type}_\delta(\alpha)$, and, for $j > 0$,

$$J_j \;=\; J_{j-1} \cup \left\{\beta \in I_{i_j+1} \mid h_{i_j}(\mathsf{guard}(\sigma_{i_j})) \prec_\delta^{gp} \beta\right\}.$$

We can now formally state the key property of the notion of type, which heavily relies on guardedness:

Lemma 6 *Consider a database D, and $\Sigma \in \mathbb{G}$. For a chase derivation δ of D w.r.t. Σ, and an atom $\alpha \in \mathsf{chase}_\delta(D,\Sigma)$, $\delta[\alpha]$ is a chase derivation of* $\mathsf{type}_\delta(\alpha)$ *w.r.t.* Σ.

Pivotal Atoms. Our alternating algorithm exploits the existence of some special atoms, called pivotal, for guarded subsets of the result of a derivation. Roughly, to check whether a guarded set of atoms Q is in the result of some chase derivation δ, it suffices to check whether Q is in the result of the α-projection of δ with α being a pivotal atom for Q.

Consider a database D, and a set $\Sigma \in \mathbb{G}$ of TGDs, both over a schema $\mathbf{S}$. A finite set of atoms $Q \subseteq \mathsf{B}(\mathbf{C} \cup \mathbf{N}, \mathbf{S})$ is *guarded* if it has an atom α such that $\mathsf{dom}(\alpha) = \mathsf{dom}(Q)$. We say that Q is *ungrounded* if each of its atoms contains at least one null of $\mathbf{N}$. Let $\mathsf{null}(Q)$ be the set of nulls in Q. Consider now a chase derivation $\delta = (I_i)_{i\geq 0}$ of D w.r.t. Σ with $I_i\langle\sigma_i, h_i\rangle I_{i+1}$, and a set of atoms $Q \subseteq \mathsf{B}(\mathbf{C} \cup \mathbf{N}, \mathbf{S})$ that is guarded and ungrounded. An atom $\alpha \in \mathsf{chase}_\delta(D,\Sigma)$ is *δ-pivotal for Q* if $\alpha = h_i(\mathsf{guard}(\sigma_i))$ for some $i \geq 0$ such that $\mathsf{null}(Q) \not\subseteq \mathsf{dom}(I_i)$, while $\mathsf{null}(Q) \subseteq \mathsf{dom}(I_{i+1})$; i.e., a δ-pivotal atom for Q is an atom of $\mathsf{chase}_\delta(D,\Sigma)$ in which the nulls in Q occur together for the first time according to δ. The key lemma concerning pivotal atoms follows:

Lemma 7 *Consider a database D, and a set $\Sigma \in \mathbb{G}$, both over a schema $\mathbf{S}$. Let δ be a chase derivation of D w.r.t. Σ, and $Q \subseteq \mathsf{B}(\mathbf{C} \cup \mathbf{N}, \mathbf{S})$ be a guarded and ungrounded set of atoms. The following are equivalent:*

1. $Q \subseteq \mathsf{chase}_\delta(D,\Sigma)$.
2. *There is an atom $\alpha \in \mathsf{chase}_\delta(D,\Sigma)$ that is δ-pivotal for Q such that $Q \subseteq \mathsf{chase}_{\delta[\alpha]}(\mathsf{type}_\delta(\alpha),\Sigma)$.*
3. *There is* exactly one *atom $\alpha \in \mathsf{chase}_\delta(D,\Sigma)$ that is δ-pivotal for Q, and $Q \subseteq \mathsf{chase}_{\delta[\alpha]}(\mathsf{type}_\delta(\alpha),\Sigma)$.*

The direction $(3) \Rightarrow (2)$ holds trivially, whereas $(2) \Rightarrow (1)$ holds since $\mathsf{chase}_{\delta[\alpha]}(\mathsf{type}_\delta(\alpha),\Sigma) \subseteq \mathsf{chase}_\delta(D,\Sigma)$. It remains to show $(1) \Rightarrow (3)$. Let $\gamma \in Q$ be the atom that contains all the terms of $\mathsf{dom}(Q)$, and let $\alpha, \beta \in \mathsf{chase}_\delta(D,\Sigma)$ be atoms such that $\alpha \prec_\delta^{gp} \beta$, $\beta \prec_\delta^{gp,+} \gamma$, $\mathsf{null}(\alpha) \not\subseteq \mathsf{null}(\beta)$, and $\mathsf{null}(\beta) \subseteq \mathsf{null}(\gamma)$. By guardedness, α is the δ-pivotal atom for Q, and, for each $\alpha' \in Q$, it holds that $\alpha \prec_\delta^{gp,+} \alpha'$ or $\alpha' \in \mathsf{type}_\delta(\alpha)$. Hence, $Q \subseteq \mathsf{chase}_{\delta[\alpha]}(\mathsf{type}_\delta(\alpha),\Sigma)$.[2]

The Alternating Algorithm

Given a database D, a set $\Sigma \in \mathbb{G}$, a CQ $R(\bar{x})$, and a tuple $\bar{c} \in \mathsf{dom}(D)^{\mathsf{ar}(R)}$, checking whether $\bar{c} \in \mathsf{cert}(R(\bar{x}), D, \Sigma)$ boils down to checking whether $R(\bar{c})$ belongs to $\mathsf{chase}_\delta(D,\Sigma)$ for some chase derivation δ of D w.r.t. Σ. Lemma 7 suggests the following strategy for the latter task: find $\sigma \in \Sigma$ and a mapping h from the variables in $\mathsf{body}(\sigma)$ to $\mathsf{dom}(D) \cup \mathbf{N}$ such that $R(\bar{c}) \in h(\mathsf{head}(\sigma))$, and check whether there exists a δ-pivotal atom for the ungrounded subset of $h(\mathsf{body}(\sigma))$, for some chase derivation δ of D w.r.t. Σ, while for each ground atom in $h(\mathsf{body}(\sigma))$ recursively apply the above strategy. Our alternating algorithm, which is described in detail next, performs the above steps in parallel universal computations, which ensures that only logarithmic space is needed. Recall that polynomial time coincides with alternating logspace.

Some Preparation. Let us first introduce some useful notions. Since we can use only logarithmic space, we cannot afford to explicitly store all the atoms generated via a single chase step since we deal with multi-heads; this could be possible in the case of single-head TGDs. Therefore, we need a way to compactly represent such a set of atoms, while such a representation should take only logspace. This is done via a so-called (D,Σ)-step, which is a compact representation of the guard atom of a chase step, together with its type, and the atoms that are generated via this chase step.

Let $\Delta_\Sigma = \{\bot_1, \ldots, \bot_{2\cdot(\mathsf{ar}(\Sigma)+1)}\}$ that collects all the different sorts of nulls that are needed to perform our check using a bounded number of nulls, which in turn will ensure that we use only logspace. Let $\Delta_\Sigma^\star = \{s^i \mid s \in \Delta_\Sigma\}_{j \in [\ell_\Sigma]} \subseteq \mathbf{N}$, where $\ell_\Sigma = \max_{\sigma\in\Sigma}\{\ell_\sigma\}$ with ℓ_σ be the number of existential variables in σ. A *(D,Σ)-step* for a set $S \subsetneq \Delta_\Sigma$ is a tuple (σ, h, s, T) – we assume that Σ is over $\mathbf{S}$ – where :

- σ is a TGD of Σ,
- $h : \mathsf{dom}(\mathsf{body}(\sigma)) \rightarrow \mathsf{dom}(D) \cup \Delta_\Sigma^\star$,
- $s \in \Delta_\Sigma \setminus S$, and
- $h(\mathsf{body}(\sigma)) \subseteq T \subseteq \mathsf{B}(\mathsf{dom}(h(\mathsf{guard}(\sigma))), \mathbf{S})$.

Such a (D,Σ)-step should be interpreted as follows. The triple (σ, h, s) encodes the set of atoms $h'(\mathsf{head}(\sigma))$, where h' extends h as follows: if $z_1, \ldots, z_k$ are the existential variables of σ, then $h'(z_i) = s^i$, i.e., the sort s and the variable z_i uniquely determine the null that is assigned to z_i. We denote this set of atoms $[\![\sigma, h, s]\!]$. Note that the fresh nulls of sort s in $[\![\sigma, h, s]\!]$ do not occur in $h(\mathsf{guard}(\sigma))$ since $h(\mathsf{guard}(\sigma))$ does not contain a null of sort s. Moreover, T corresponds to the type of $h(\mathsf{guard}(\sigma))$. For a set $N \subseteq \Delta_\Sigma^\star$, it would be

[2]Note that $\delta[\alpha]$ is well defined due to Lemma 6.

Algorithm 1: Instance Checking

```
Entail($D, \Sigma, R(\bar{x}), \bar{c}$)

if $R(\bar{c}) \in D$ then
  return Accept
else
  guess $\sigma \in \Sigma$ and
    $h : \mathsf{dom}(\mathsf{body}(\sigma)) \rightarrow \mathsf{dom}(D) \cup \Delta^{\star}_{\Sigma}$
  if $R(\bar{c}) \notin h(\mathsf{head}(\sigma))$ then
    return Reject
  else
    return Proof($h(\mathsf{body}(\sigma))$)

Proof($Q$)

$Q' := Q \setminus \{R(\bar{c}) \in Q \mid \bar{c} \in \mathbf{C}^{\mathsf{ar}(R)}\}$
$N := \mathsf{null}(Q')$
universally do:
▷ universally select every atom $R(\bar{c}) \in Q \setminus Q'$
  return Entail($D, \Sigma, R(\bar{x}), \bar{c}$)
▷ if $N = \varnothing$ then
  return Accept
else
  guess $(D, \Sigma)$-step $(\sigma, h, s, T)$ for $s_{\Sigma}(\sigma, h)$
  if $N \subseteq \mathsf{null}(h(\mathsf{guard}(\sigma)))$ or $N \not\subseteq \{s^1, \ldots, s^{\ell_\sigma}\}$
   then
    return Reject
  else
    universally do:
    ▷ universally select every atom $\beta \in Q' \setminus T$
      return Reach($\sigma, h, s, T, \beta$)
    ▷ return Proof($T$)

Reach($\sigma, h, s, T, \beta$)

guess an atom $\alpha \in [\![\sigma, h, s]\!]$
if $\alpha = \beta$ then
  return Accept
else
  $S := s_{\Sigma}(\sigma, h) \cup \{s\} \cup s_{\Sigma}(\mathsf{null}(\beta))$
  guess $(D, \Sigma)$-step $(\sigma', h', s', T')$ for $S$
  if $h'(\mathsf{guard}(\sigma')) \neq \alpha$ then
    return Reject
  else
    universally do:
    ▷ return Reach($\sigma', h', s', T', \beta$)
    ▷ universally select every atom $\beta' \in T' \setminus T$
      return Reach($\sigma, h, s, T, \beta'$)
```

useful to be able to extract the set of sorts of nulls occurring in N, i.e., the set $s_{\Sigma}(N) = \{s \mid s^j \in N\} \subseteq \Delta_{\Sigma}$. We write $s_{\Sigma}(\sigma, h)$ instead of the formal $s_{\Sigma}(\mathsf{null}(h(\mathsf{body}(\sigma))))$.

Description of the Algorithm. Our alternating algorithm is depicted in the box above. Here is a detailed description:

▶ The procedure Entail implements the strategy discussed above: find $\sigma \in \Sigma$ and a mapping h that maps $\mathsf{var}(\mathsf{body}(\sigma))$ to $\mathsf{dom}(D) \cup \Delta^{\star}_{\Sigma}$ with $R(\bar{c}) \in h(\mathsf{head}(\sigma))$, and check, via the procedure Proof, that $h(\mathsf{body}(\sigma))$ is derivable via some chase derivation. Note that we cannot afford to use an unlimited number of null values since we have limited space. The nulls of $\Delta^{\star}_{\Sigma}$ are enough for faithfully checking the above.

▶ The procedure Proof checks whether a set of atoms Q is derivable via some chase derivation. Each ground atom of Q is proved in a parallel universal computation by recursively calling the procedure Entail. The remaining atoms form a guarded and ungrounded set Q'; if $\mathsf{null}(Q)$ is empty, which means that Q' is empty, then the algorithm accepts. In a parallel universal computation, it checks whether there is a δ-pivotal atom α for Q', for some chase derivation δ, such that Q' is derivable by applying chase steps starting from the δ-type of α (this is enough due to Lemma 7; notice that here we simply exploit the existence of a δ-pivotal atom, i.e., item (2) of Lemma 7). The guessed (D, Σ)-step (σ, h, s, T), actually $h(\mathsf{guard}(\sigma))$, corresponds to the δ-pivotal atom for Q', and the algorithm performs in parallel universal computations the following checks: (i) each atom β of Q', which is not already in the δ-type of $h(\mathsf{guard}(\sigma))$ given by T, is derivable by applying chase steps starting from T, which is done in parallel universal computations by calling the procedure Reach, and (ii) T is indeed the δ-type of $h(\mathsf{guard}(\sigma))$, which is done by recursively calling the procedure Proof.

▶ The procedure Reach actually checks whether, for some chase derivation δ of T w.r.t. Σ, $h(\mathsf{guard}(\sigma)) \prec^{gp,+}_{\delta} \beta$. It starts by guessing an atom α from $[\![\sigma, h, s]\!]$. If α is the atom that we are targeting, namely β, then it accepts; otherwise, it proceeds to check whether $\alpha \prec^{gp,+}_{\delta} \beta$. Due to Lemma 6, to check whether $\alpha \prec^{gp,+}_{\delta} \beta$, it suffices to consider only the δ-type of α. In fact, the guessed (D, Σ)-step (σ', h', s', T') provides the trigger to apply at the next step, that is, (σ', h') with $h'(\mathsf{guard}(\sigma')) = \alpha$, the sort of the fresh nulls that will appear in the generated atoms, that is, s', and the δ-type of α, that is, T'. It remains to verify that (i) $\alpha \prec^{gp,+}_{\delta} \beta$, and (ii) T' is indeed the δ-type of α. The former check is done by recursively calling the procedure Reach with input (σ', h', s', T'). For the latter check, one may suggest that the algorithm can simply call Proof(T'). It should not be overlooked, though, that β and T' may share null values, and this fact should be preserved. However, by calling Proof(T') in a parallel universal computation, we loose this connection between β and T', which may lead to unsound results. The key observation is that $T' \setminus T$ is a guarded and ungrounded set of atoms that has the same δ-pivotal atom as $Q' \setminus T$ (the set of atoms from which β is coming from), which, by item (3) of Lemma 7, is unique; this atom is $h(\mathsf{guard}(\sigma))$. Hence, to prove T', the algorithm recursively calls for each atom $\beta' \in T' \setminus T$, in a parallel universal computation, Reach(σ, h, s, T, β').

By Lemmas 6 and 7, the algorithm Entail is correct, i.e., Entail($D, \Sigma, R(\bar{x}), \bar{c}$) accepts iff $R(\bar{c}) \in \mathsf{chase}_{\delta}(D, \Sigma)$, for some chase derivation δ of D w.r.t. Σ. Furthermore, at each step of its computation, the algorithm uses logarithmic space for storing elements of $\mathsf{dom}(D)$, and auxiliary pointers; recall that the schema is fixed. Since polynomial time coincides with alternating logarithmic space, Theorem 5 follows.

4.2 Step 2: Linearization

We now proceed with the second step of the proof for showing the NP upper bound stated in Theorem 4. We show that:

Lemma 8 *For a fixed schema* $\mathbf{S}$*, there exists a schema* $\mathbf{S}' \supseteq \mathbf{S}$ *with* $\mathsf{ar}(\mathbf{S}) = \mathsf{ar}(\mathbf{S}')$ *such that* $\mathsf{OQA}(\mathbb{G}[\mathbf{S}])$ *can be reduced in polynomial time to* $\mathsf{OQA}(\mathbb{L}[\mathbf{S}'])$.

The above has been shown for single-head TGDs in (Gottlob, Manna, and Pieris 2014). By using the property of the type for multi-head guarded TGDs discussed in the previous section (Lemma 6), we can adapt the reduction in (Gottlob, Manna, and Pieris 2014) in order to operate on multi-head TGDs. We only give the high-level idea of the reduction.

Let D be a database, and Σ a set from $\mathbb{G}[\mathbf{S}]$. Our goal is to devise a database D^*, and a set $\Sigma^* \in \mathbb{L}[\mathbf{S}']$, with $\mathbf{S}' \supseteq \mathbf{S}$ and $\mathsf{ar}(\mathbf{S}) = \mathsf{ar}(\mathbf{S}')$, such that, for every CQ q, $\mathsf{cert}(q, D, \Sigma) = \mathsf{cert}(q, D^*, \Sigma^*)$. For D^* the idea is to encode an atom $\alpha \in D$ and its δ-type, for some chase derivation δ of D w.r.t. Σ, as an atom of the form $[\tau](\cdot)$, where τ is a symbolic encoding of α and its δ-type. For example, given $R(a,b) \in D$ with

$$\mathsf{type}_\delta(R(a,b)) \ = \ \{R(a,b), S(b,a), T(a), T(b)\},$$

we can encode $R(a,b)$ and its δ-type as the atom

$$[R(1,2), \{S(2,1), T(1), T(2)\}](a,b).$$

When it comes to Σ^*, the intention is, for a TGD $\sigma \in \Sigma$, to encode the shape of the type τ of the guard of σ in a predicate $[\tau]$, and then replace σ with a linear TGD that uses in its body an atom of the form $[\tau](\cdot)$. However, we need an effective way to compute the type of an atom α by completing its known part, which is inherited from the type of the guard atom that generates α, with atoms that mention the new null values invented in α. This exploits the main property of the type discussed in the previous section. In particular, for each α obtained due to the application of the trigger (σ, h), where $\sigma = \varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}) \in \Sigma$, we can construct the type of α from $h'(\psi(\bar{x}, \bar{z}))$, where h' extends h employed during this application, together with the restriction of the type of α to the terms $h(\bar{x})$. To construct the predicates $[\tau]$ we heavily rely on instance checking for guarded TGDs. Thus, the fact D^* and Σ^* can be constructed in polynomial time follows from Theorem 5, which shows that $\mathsf{IC}(\mathbb{G}[\mathbf{S}])$ is in PTIME.

4.3 Step 3: The Linear Case

We now proceed with the last step of the proof for establishing the NP upper bound in Theorem 4. We show that:

Theorem 9 $\mathsf{OQA}(\mathbb{L})$ *is* NP*-complete for fixed arity.*

Note that $\mathsf{OQA}(\mathbb{L}_1)$ is NP-complete for fixed arity (Calì, Gottlob, and Lukasiewicz 2012), but it cannot be directly transferred to multi-head TGDs. We proceed to discuss the proof of Theorem 9. The NP-hardness holds even for CQ evaluation for schemas consisting of a single binary relation. The interesting task is to show the upper bound. To this end, we rely on the notion of polynomial witness property.

Definition 10 *A class* $\mathbb{C} \subseteq \mathbb{TGD}$ *of TGDs enjoys the* polynomial witness property *(PWP) if there exists a polynomial* $\mathsf{pol}(\cdot)$ *such that, for every database D, set $\Sigma \in \mathbb{C}$, CQ $q(\bar{x})$, and tuple $\bar{c} \in \mathbf{C}^{|\bar{x}|}$, $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$ implies the existence of a sequence $(I_i)_{0 \leq i \leq n}$, with $n \leq \mathsf{pol}(||\Sigma|| + ||q||)$, that is a prefix of a chase derivation of D w.r.t. Σ, and $\bar{c} \in q(I_n)$.*[3]

The PWP guarantees that if $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$, then this can be realized after polynomially many chase applications. This leads to a simple guess-and-check algorithm for establishing that $\mathsf{OQA}(\mathbb{C})$ is in NP for any class $\mathbb{C}$ that enjoys the PWP: guess a sequence of instances $\delta = (I_i)_{0 \leq i \leq n}$, with $I_0 = D$ and $I_{i+1} \supsetneq I_i$, and a sequence of TGD-mapping pairs $((\sigma_i, h_i))_{0 \leq i < n}$, where $0 \leq n \leq \mathsf{pol}(||\Sigma|| + ||q||)$, where $\mathsf{pol}(\cdot)$ is the polynomial provided by the PWP, together with an additional mapping from the variables in q to $\mathsf{dom}(I_n)$, and then verify that δ is a chase derivation of D w.r.t. Σ, and that $\bar{c} \in q(I_n)$. Thus, to establish the NP upper bound in Theorem 9, it suffices to show the following result:

Theorem 11 *For a schema* $\mathbf{S}$ *of fixed arity,* $\mathbb{L}[\mathbf{S}]$ *enjoys the polynomial witness property.*

The above holds for $\mathbb{L}_1[\mathbf{S}]$; implicit in (Calì, Gottlob, and Lukasiewicz 2012). However, it cannot be straightforwardly transferred to $\mathbb{L}[\mathbf{S}]$. Our proof proceeds in two steps:

1. We provide a characterization of the PWP for subclasses of $\mathbb{L}$ via parsimonious quasi chase derivations, which are sequences that are "almost" a prefix of a chase derivation, and involve polynomially many distinct triggers.
2. We then show that in the case of $\mathbb{L}[\mathbf{S}]$, for a schema $\mathbf{S}$ of fixed arity, the fact that a tuple is a certain answer can be witnessed via a parsimonious quasi chase derivation.

Before we give further details for the above two steps, we first need to introduce quasi chase derivations.

Quasi Chase Derivations. Let (σ, h) be a trigger for a set Σ of TGDs on an instance I. A *partial application* of (σ, h) to I returns an instance $J = I \cup K$, where $K \subseteq h'(\mathsf{head}(\sigma))$, while h' extends h in such a way that for each existentially quantified variable z of σ, $h'(z) \in \mathbf{N} \setminus \mathsf{dom}(I)$, and for distinct existentially quantified variables z and w of σ, $h'(z) \neq h'(w)$. Such a partial application is denoted $I\langle\sigma, h\rangle^p J$. A sequence of instances $(I_i)_{0 \leq i \leq n}$, for $n \geq 0$, is a *quasi chase derivation* (resp., *quasi chase derivation with total applications*) of D w.r.t. Σ if, for each $0 \leq i < n$, there exists a trigger (σ, h) for Σ on I_i such that $I_i\langle\sigma, h\rangle^p I_{i+1}$ (resp., $I_i\langle\sigma, h\rangle I_{i+1}$). Observe that a quasi chase derivation with total applications is trivially a quasi chase derivation, but the opposite is not necessarily true. A quasi chase derivation $\delta = (I_i)_{0 \leq i \leq n}$, with $I_i\langle\sigma_i, h_i\rangle^p I_{i+1}$, is *$k$-parsimonious*, for $k \geq 0$, if $|\bigcup_{0 \leq i < n}\{(\sigma_i, h_i)\}| \leq k$, i.e., at most k distinct triggers are involved in δ. The parent relation of δ, denoted $\prec_\delta^p$, and its transitive closure $\prec_\delta^{p,+}$, are defined as expected. In a nutshell, a quasi chase derivation is almost a chase derivation, but it is not exactly a chase derivation for the following three reasons: (1) applications may be partial, that is, during a trigger application some atoms are not generated, (2) tiggers may repeat, i.e., the same trigger is used in different applications, and (3) fairness is not satisfied, i.e., some triggers have not been applied. Of course, in the case

[3]As usual, we write $||O||$ for the size of a syntactic object O.

of quasi chase derivations with total applications only the last two reasons apply. Moreover, although triggers may repeat in a k-parsimonious quasi chase derivation, at most k distinct triggers have been used for its generation.

Contractions. Consider a database D, a set $\Sigma \in \mathbb{L}$ of TGDs, and a quasi chase derivation $\delta = (I_i)_{0 \leq i \leq n}$, for $n \geq 0$, of D w.r.t. Σ. Let $A \subseteq I_n$, and h a mapping from $\mathsf{dom}(A)$ to $\mathsf{dom}(I_n)$. The (A, h)-*contraction of* δ is the sequence of instances $\delta' = (J_i)_{0 \leq i \leq n}$ such that, for each $0 \leq i \leq n$

$$J_i \;=\; I_i \setminus \left\{\alpha \mid \alpha \in I_i \text{ and } A \prec_{\delta}^{p,+} \alpha\right\} \;\cup \\ \left\{h(\alpha) \mid \alpha \in I_i \text{ and } A \prec_{\delta}^{p,+} \alpha\right\}.$$

For brevity, we write $A \prec_{\delta}^{p,+} \alpha$ for the fact that $\beta \prec_{\delta}^{p,+} \alpha$ for some atom $\beta \in A$. Simply stated, the (A, h)-contraction of δ is essentially what we obtain after updating, according to the mapping h, the atoms in the subtrees of $\prec_{\delta}^{p}$ rooted at A; recall that, due to linearity, $\prec_{\delta}^{p}$ forms a rooted forest. We can now discuss the details of the proof for Theorem 11.

PWP and Parsimonious Quasi Chase Derivations

The characterization of PWP via parsimonious quasi chase derivations (step 1 above) is given by the following result:

Proposition 12 *For $\mathbb{C} \subseteq \mathbb{L}$, the following are equivalent:*

1. *$\mathbb{C}$ enjoys the PWP.*
2. *There is a polynomial* $\mathsf{pol}(\cdot)$ *such that, for every database D, set $\Sigma \in \mathbb{C}$, CQ $q(\bar{x})$, and $\bar{c} \in \mathbf{C}^{|\bar{x}|}$, $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$ implies there exists a* $\mathsf{pol}(||\Sigma|| + ||q||)$*-parsimonious quasi chase derivation with total applications $(I_i)_{0 \leq i \leq n}$, where $n \geq 0$, of D w.r.t. Σ such that $\bar{c} \in q(I_n)$.*

Let $k = \mathsf{pol}(||\Sigma|| + ||q||)$. It is clear that (1) implies (2) since a prefix of a chase derivation of length k of D w.r.t. Σ is by definition a k-parsimonious quasi chase derivation of D w.r.t. Σ. The other direction, however, is not immediate since a k-parsimonious quasi chase derivation δ of D w.r.t. Σ with total applications is not necessarily a prefix of length at most k of a chase derivation of D w.r.t. Σ since triggers may repeat. Thus, if we could eliminate from δ the repeated triggers, without introducing more repetitions of triggers, then we will end up with a finite prefix of a chase derivation of D w.r.t. Σ of length at most k, which in turn implies the PWP. This can be achieved via iterative contractions.

Assume that a trigger (σ, h) has been applied at steps i, j, for $i < j$, with H_i and H_j being the set of atoms generated at the i-th and j-th step, respectively. The key observation is that $H_i \simeq_S H_j$, where $S = \mathsf{dom}(H_i) \cap \mathsf{dom}(H_j)$. Due to linearity, we can safely move the subtrees rooted at H_j under the corresponding atoms of H_i, and consistently update the atoms, without introducing more repetitions of triggers. This is essentially what the (H_j, μ^{-1})-contraction of δ does, with μ being the S-isomorphism from H_i to H_j. This is formalized by the following technical lemma. For a quasi chase derivation $\delta = (I_i)_{0 \leq i \leq n}$ with $I_i\langle\sigma_i, h_i\rangle I_{i+1}$, we write H_{δ}^{i} for the set of atoms generated at the i-th step, i.e., the set of atoms $h_i'(\mathsf{head}(\sigma_i))$, where h_i' is the extension of h_i employed during the trigger application $I_i\langle\sigma_i, h_i\rangle I_{i+1}$.

Lemma 13 *Consider a database D, $\Sigma \in \mathbb{L}$, CQ $q(\bar{x})$, and $\bar{c} \in \mathbf{C}^{|\bar{x}|}$. Let $\delta = (I_i)_{0 \leq i \leq n}$ be a quasi chase derivation with total applications of D w.r.t. Σ with $I_i\langle\sigma_i, h_i\rangle I_{i+1}$, and assume $\bar{c} \in q(I_n)$. For a pair of indices $0 \leq k < \ell < n$ with $(\sigma_k, h_k) = (\sigma_\ell, h_\ell)$, let $\delta_{k,\ell} = (J_i)_{0 \leq i \leq n}$ be the $(H_{\delta}^{\ell}, \mu^{-1})$-contraction of δ, with μ being the* $(\mathsf{dom}(H_{\delta}^{k}) \cap \mathsf{dom}(H_{\delta}^{\ell}))$*-isomorphism from H_{δ}^{k} to H_{δ}^{ℓ}. The following hold:*

1. *$\delta_{k,\ell}$ is a quasi chase derivation with total applications of D w.r.t. Σ with $J_i\langle\sigma_i, \mu_i\rangle J_{i+1}$ for some mapping μ_i.*
2. *There is no $\alpha \in J_n$ such that $H_{\delta}^{\ell} \prec_{\delta_{k,\ell}}^{p,+} \alpha$.*
3. *For each $0 \leq i, j < n$, $(\sigma_i, h_i) = (\sigma_j, h_j)$ if and only if $(\sigma_i, \mu_i) = (\sigma_j, \mu_j)$.*
4. *$\bar{c} \in q(J_n \setminus H_{\delta}^{\ell})$.*

We can now explain how the direction (2) implies (1) is shown. Assume that $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$. By hypothesis, there exists a $\mathsf{pol}(||\Sigma|| + ||q||)$-parsimonious quasi chase derivation with total applications $\delta = (I_i)_{0 \leq i \leq n}$ of D w.r.t. Σ such that $\bar{c} \in q(I_n)$. By iteratively applying Lemma 13, we can eventually construct a $\mathsf{pol}(||\Sigma|| + ||q||)$-parsimonious quasi chase derivation with total applications $\delta' = (J_i)_{0 \leq i \leq n}$ of D w.r.t. Σ with $J_i\langle\sigma_i, h_i\rangle J_{i+1}$ such that, for each $0 < j < n$ for which there exists $0 \leq i < j$ with $(\sigma_i, h_i) = (\sigma_j, h_j)$:

- There is no $\alpha \in J_n$ such that $H_{\delta}^{j} \prec_{\delta'}^{p,+} \alpha$.
- $\bar{c} \in q(J_n \setminus H_{\delta}^{j})$.

Therefore, we can drop the applications in δ' that use a trigger that has been applied before, and get $\delta'' = (K_i)_{0 \leq i \leq m}$, with $m \leq \mathsf{pol}(||\Sigma|| + ||q||)$, that is a prefix of a chase derivation of D w.r.t. Σ, and $\bar{c} \in q(K_m)$, as needed.

OQA via Parsimonious Quasi Chase Derivations

We now show that in the case of the class $\mathbb{L}[\mathbf{S}]$, for a schema $\mathbf{S}$, the fact that a tuple is a certain answer can be witnessed via a k-parsimonious quasi chase derivation, where k depends polynomially on the cardinality of the set of TGDs Σ, the cardinality of the schema $\mathbf{S}$, and the number of atoms in the CQ q, and exponentially on $\mathsf{ar}(\mathbf{S})$. This in turn implies that in the case of a schema of fixed arity, k is a polynomial (step 2 above). More precisely, assuming that

$$T_{q,\Sigma} \;=\; |\Sigma| \cdot |\mathbf{S}| \cdot \left(2 \cdot |q| \cdot \mathsf{ar}(\mathbf{S}) + \mathsf{ar}(\mathbf{S})\right)^{\mathsf{ar}(\mathbf{S})},$$

we can show the following result:

Proposition 14 *Consider a database D, a set $\Sigma \in \mathbb{L}[\mathbf{S}]$, a CQ $q(\bar{x})$, and $\bar{c} \in \mathbf{C}^{|\bar{x}|}$. If $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$, then there is a $T_{q,\Sigma}$-parsimonious quasi chase derivation with total applications $(J_i)_{0 \leq i \leq n}$ of D w.r.t. Σ such that $\bar{c} \in q(J_n)$.*

To establish the above result it suffices to show that the prefix $(I_i)_{0 \leq i \leq n}$ of some chase derivation of D w.r.t. Σ that witnesses the fact that $\bar{c} \in \mathsf{cert}(q, D, \Sigma)$, i.e., $\bar{c} \in q(I_n)$, can be converted into a $T_{q,\Sigma}$-parsimonious quasi chase derivation with total applications $(J_i)_{0 \leq i \leq n}$ of D w.r.t. Σ such that $\bar{c} \in q(J_n)$. We can rely again on iterative contractions.

Consider two S-isomorphic atoms α and β of I_n, where S is the set of terms occurring in $h(q(\bar{c}))$ with h being a homomorphism that maps $q(\bar{c})$ to I_n. Due to linearity, we can safely move the subtree rooted at β under α, and consistently update its atoms, without affecting the fact that $\bar{c}$ is a certain answer. We can achieve this via the (β, μ^{-1})-contraction of $(I_i)_{0 \leq i \leq n}$, with μ being the S-isomorphism from α to β.[4] Notice, however, that such a contraction may create partial applications and repeated triggers. Hence, the result is not necessarily a prefix of a chase derivation, but a quasi chase derivation. This is formalized by the following lemma.

Lemma 15 *Consider a database D, $\Sigma \in \mathbb{L}$, a CQ $q(\bar{x})$, and $\bar{c} \in \mathbf{C}^{|\bar{x}|}$. Let $\delta = (I_i)_{0 \leq i \leq n}$ be a quasi chase derivation of D w.r.t. Σ such that $q(\bar{c}) \rightarrow I_n$ via h. For atoms $\alpha, \beta \in I_n$ such that $\beta \not\prec_{\delta}^{p,+} \alpha$, and $\alpha \simeq_S \beta$, where $S = \mathsf{dom}(h(q(\bar{c})))$, let $\delta_{\alpha,\beta} = (J_i)_{0 \leq i \leq n}$ be the (β, μ^{-1})-contraction of δ, with μ being the S-isomorphism from α to β. Then:*

1. *$\delta_{\alpha,\beta}$ is a quasi chase derivation of D w.r.t. Σ.*
2. *There is no $\gamma \in J_n$ such that $\beta \prec_{\delta_{\alpha,\beta}}^{p,+} \gamma$.*
3. *For each $\gamma, \gamma' \in J_n$, $\gamma \simeq_S \gamma'$ iff $\mu(\gamma) \simeq_S \mu(\gamma')$.*
4. *$q(\bar{c}) \rightarrow J_n$ via h.*

We can now explain how Proposition 14 is shown. By hypothesis, there exists a prefix $\delta = (I_i)_{0 \leq i \leq n}$, for $n \geq 0$, of a chase derivation of D w.r.t. Σ such that $q(\bar{c}) \rightarrow I_n$ via a homomorphism h. By definition, δ is a quasi chase derivation of D w.r.t. Σ. The binary relation $\simeq_S$ over I_n, where $S = \mathsf{dom}(h(q(\bar{c})))$, is an equivalence relation. Let $I_n/\simeq_S$ be the set of all equivalence classes in I_n w.r.t. $\simeq_S$. For each equivalence class C in $I_n/\simeq_S$, its canonical atom is arbitrarily chosen from the set of atoms $\{\alpha \in C \mid$ there is no $\beta \in C$ such that $\beta \prec_{\delta}^{p,+} \alpha\}$. For an atom $\alpha \in I_n$, we write $[\alpha]$ for the canonical atom of its equivalence class, and ι_α for the S-isomorphism that maps $[\alpha]$ to α. We proceed to construct a $T_{q,\Sigma}$-parsimonious quasi chase derivation with total applications $\delta' = (J_i)_{0 \leq i \leq n}$ of D w.r.t. Σ such that $\bar{c} \in q(J_n)$).

Let $\alpha \in I_n$ with $\alpha \neq [\alpha]$. By Lemma 15, the $(\alpha, \iota_\alpha^{-1})$-contraction $\delta_\alpha = (I_i^\alpha)_{0 \leq i \leq n}$ of δ enjoys the following:

- δ_α is a quasi chase derivation of D w.r.t. Σ.
- There is no $\beta \in J_n$ such that $\alpha \prec_{\delta_\alpha}^{p,+} \beta$.
- For each $\beta, \beta' \in J_n$, $\beta \simeq_S \beta'$ iff $\mu(\beta) \simeq_S \mu(\beta')$.
- $q(\bar{c}) \rightarrow J_n$ via h.

We can therefore iteratively apply Lemma 15 as discussed above, until we get a quasi chase derivation $\delta^\diamond = (I_i^\diamond)_{0 \leq i \leq n}$ of D w.r.t. Σ such that the following hold:

1. For each $\alpha \in I_n^\diamond$, $\alpha \neq [\alpha]$ implies there is no $\beta \in I_n^\diamond$ such that $\alpha \prec_{\delta^\diamond} \beta$.
2. $q(\bar{c}) \rightarrow I_n^\diamond$ via h.

Note that $\delta^\diamond$ is not necessarily with total applications. However, it can be converted into one with total applications by simply adding the atoms that are needed in order to ensure that every application in $\delta^\diamond$ is total. Let $\delta' = (J_i)_{0 \leq i \leq n}$ be the resulted quasi chase derivation with total applications of D w.r.t. Σ. It is clear that $q(\bar{c}) \rightarrow J_n$ via h, and thus $\bar{c} \in q(I_n)$. It remains to show that δ' is $T_{q,\Sigma}$-parsimonious.

[4]For a singleton instance $\{\gamma\}$ we simply write γ.

Since in δ' only canonical atoms trigger TGDs (the first property of $\delta^\diamond$ stated above, which is inherited by δ'), it suffices to count how many triggers for Σ on the instance

$$K \;=\; \{\alpha \mid \alpha \text{ is the canonical atom of a set } C \in J_n/\simeq_S\}$$

can be formed. We can assume, w.l.o.g. due to linearity, that in δ' at most $|q|$ atoms from D trigger a TGD of Σ, which implies that K contains at most $|q|$ atoms of D. Therefore, $|K| \leq |\mathbf{S}| \cdot \big(2 \cdot |q| \cdot \mathsf{ar}(\mathbf{S}) + \mathsf{ar}(\mathbf{S})\big)^{\mathsf{ar}(\mathbf{S})}$, which is the number of non-S^+-isomorphic atoms over $\mathbf{S}$ that can be formed, where S^+ consists of the set S, and the set of constants T occurring in the atoms of D that trigger a TGD. Actually, the above upper bound is a consequence of the fact that $|S| \leq |q| \cdot \mathsf{ar}(\mathbf{S})$ and $|T| \leq |q| \cdot \mathsf{ar}(\mathbf{S})$. Since each atom of K can trigger several TGDs of Σ, the total number of triggers for Σ on K that can be formed is $T_{q,\Sigma}$, and Proposition 14 follows.

Theorem 11 follows from Propositions 12 and 14.

5 Polynomial Combined Rewritability

We now turn our attention to the notion of polynomial combined first-order rewritability. A *database rewriter* is a function that receives as input a database and a set of TGDs, and outputs a database. A *query rewriter* is a function that takes a CQ and a set of TGDs, and outputs a first-order query.

Definition 16 *A class $\mathbb{C} \subseteq \mathbb{TGD}$ of TGDs is* polynomially combined first-order rewritable *if there are polynomial time computable database and query rewriters f_{DB} and f_{Q}, respectively, such that the following holds: for every database D, set $\Sigma \in \mathbb{C}$ of TGDs, and CQ q, $\mathsf{cert}(q, D, \Sigma) = q_\Sigma(D_\Sigma)$ with $D_\Sigma = f_{\mathsf{DB}}(D, \Sigma)$ and $q_\Sigma = f_{\mathsf{Q}}(q, \Sigma)$. We also say that $\mathbb{C}$ is polynomially combined first-order rewritable* targeting existential positive queries ($\exists$FO$^+$) *if q_Σ is an $\exists$FO$^+$ query.*

For a fixed schema $\mathbf{S}$, $\mathbb{G}_1[\mathbf{S}]$ is polynomially combined first-order rewritable targeting $\exists$FO$^+$ (Gottlob, Manna, and Pieris 2014). This carries over to multi-head TGDs:

Theorem 17 *For a fixed schema* $\mathbf{S}$, $\mathbb{G}[\mathbf{S}]$ *is polynomially combined first-order rewritable targeting* $\exists$FO$^+$.

Interestingly, the above result can be obtained by exploiting the machinery introduced in the previous section. In particular, Theorem 11, and the fact that every class of TGDs that enjoys the PWP is polynomially combined first-order rewritable targeting $\exists$FO$^+$ (Gottlob et al. 2014), imply that:

Theorem 18 *For a schema* $\mathbf{S}$ *of fixed arity,* $\mathbb{L}[\mathbf{S}]$ *is polynomially combined first-order rewritable targeting* $\exists$FO$^+$.

The above result, and the polynomial time reduction from $\mathsf{OQA}(\mathbb{G}[\mathbf{S}])$ to $\mathsf{OQA}(\mathbb{L}[\mathbf{S}'])$ with $\mathsf{ar}(\mathbf{S}) = \mathsf{ar}(\mathbf{S}')$ provided by Lemma 8, allow us to establish Theorem 17.

6 Conclusions

In this work, we closed a gap in the analysis of ontology-based query answering for guarded TGDs in the case of fixed schemas. In particular, we proved that the problem is NP-complete, and polynomially combined first-order rewritable, even if we consider multi-head guarded TGDs.

Acknowledgments

Gottlob is a Royal Society Research Professor and acknowledges support by the Royal Society for the present work in the context of the project ”RAISON DATA” (project reference: RP/R1/201074). Manna has been partially supported by MISE under the project “S2BDW” (F/050389/01-03/X32) - Horizon 2020 PON I&C2014-20 and by Regione Calabria under the project “DLV LargeScale” (CUP J28C17000220006) - POR Calabria 2014-20. Pieris was supported by the EPSRC grant EP/S003800/1 “EQUID”.

References

Andréka, H.; van Benthem, J.; and Németi, I. 1998. Modal languages and bounded fragments of predicate logic. *J. Philosophical Logic* 27:217–274.

Arenas, M.; Gottlob, G.; and Pieris, A. 2018. Expressive languages for querying the semantic web. *ACM Trans. Database Syst.* 43(3):13:1–13:45.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10):1620–1654.

Bárány, V.; Benedikt, M.; and ten Cate, B. 2013. Rewriting guarded negation queries. In *MFCS*, 98–110.

Benedikt, M. 2018. Personal Communication.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, 228–242.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V. V.; Schwentick, T.; and Zakharyaschev, M. 2014. The price of query rewriting in ontology-based data access. *Artif. Intell.* 213:42–59.

Gottlob, G.; Manna, M.; and Pieris, A. 2014. Polynomial combined rewritings for existential rules. In *KR*.

Gottlob, G.; Rudolph, S.; and Simkus, M. 2014. Expressiveness of guarded existential rule languages. In *PODS*, 27–38.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *IJCAI*, 2070–2075.