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

First-Order Rewritability of Frontier-Guarded Ontology-Mediated Queries

Pablo Barceló[1], Gerald Berger[2], Carsten Lutz[3] and Andreas Pieris[4]
[1] Millennium Institute for Foundational Research on Data & DCC, University of Chile
[2] Institute of Logic and Computation, TU Wien
[3] Department of Mathematics and Computer Science, University of Bremen
[4] School of Informatics, University of Edinburgh

Abstract

We focus on ontology-mediated queries (OMQs) based on (frontier-)guarded existential rules and (unions of) conjunctive queries, and we investigate the problem of FO-rewritability, i.e., whether an OMQ can be rewritten as a first-order query. We adopt two different approaches. The first approach employs standard two-way alternating parity tree automata. Although it does not lead to a tight complexity bound, it provides a transparent solution based on widely known tools. The second approach relies on a sophisticated automata model, known as cost automata. This allows us to show that our problem is 2EXPTIME-complete. In both approaches, we provide semantic characterizations of FO-rewritability that are of independent interest.

1 Introduction

Ontology-based data access (OBDA) is a successful application of KRR technologies in information management systems [Poggi *et al.*, 2008]. One premier goal is to facilitate access to data that is heterogeneous and incomplete. This is achieved via an ontology that enriches the user query, typically a union of conjunctive queries, with domain knowledge. It turned out that the ontology and the user query can be seen as two components of one composite query, called *ontology-mediated query* (OMQ) [Bienvenu *et al.*, 2014]. The problem of answering OMQs is thus central to OBDA.

Building ontology-aware database systems from scratch, with sophisticated optimization techniques, is a non-trivial task that requires a great effort. An important route towards practical implementation of OMQ answering is thus to use conventional database management systems. The problem that such systems are unaware of ontologies can be addressed by query rewriting: the ontology $\mathcal{O}$ and the database query q are combined into a new query $q_{\mathcal{O}}$, the so-called rewriting, which gives the same answer as the OMQ consisting of $\mathcal{O}$ and q over all input databases. It is of course essential that the rewriting $q_{\mathcal{O}}$ is expressed in a language that can be handled by standard database systems. The typical language that is considered in this setting is first-order (FO) queries.

Although in the OMQ setting description logics (DLs) are often used for modeling ontologies, it is widely accepted that for handling arbitrary arity relations in relational databases it is convenient to use *tuple-generating dependencies* (TGDs), a.k.a. *existential rules* or *Datalog$^{\pm}$ rules*. It is known, however, that evaluation of rule-based OMQs is undecidable [Calì *et al.*, 2013]. This has led to a flurry of activity for identifying restrictions on TGDs that lead to decidability. The main decidable classes are (i) *(frontier-)guarded* TGDs [Baget *et al.*, 2011; Calì *et al.*, 2013], which includes *linear* TGDs [Calì *et al.*, 2012a], (ii) *acyclic* sets of TGDs [Fagin *et al.*, 2005], and (iii) *sticky* sets of TGDs [Calì *et al.*, 2012b]. There are also extensions that capture Datalog; see the same references.

For OMQs based on linearity, acyclicity, and stickiness, FO-rewritings are always guaranteed to exist [Gottlob *et al.*, 2014]. In contrast, there are (frontier-)guarded OMQs that are inherently recursive, and thus not expressible as a first-order query. This brings us to our main question: *Can we check whether a (frontier-)guarded OMQ is FO-rewritable?* Notice that for OMQs based on more expressive classes of TGDs that capture Datalog, the answer to the above question is negative, since checking whether a Datalog query is FO-rewritable is an undecidable problem. Actually, we know that a Datalog query is FO-rewritable iff it is bounded [Ajtai and Gurevich, 1994], while the boundedness problem for Datalog is undecidable [Gaifman *et al.*, 1993].

The above question has been studied for OMQ languages based on Horn DLs, including $\mathcal{EL}$ and $\mathcal{ELI}$, which (up to a certain normal form) are a special case of guarded TGDs [Bienvenu *et al.*, 2013; 2016; Lutz and Sabellek, 2017]. More precisely, FO-rewritability is semantically characterized in terms of the existence of certain tree-shaped ABoxes, which in turn allows the authors to pinpoint the complexity of the problem by employing automata-based procedures. As usual in the DL context, schemas consist only of unary and binary relations. However, in our setting we have to deal with relations of higher arity. This indicates that the techniques devised for checking the FO-rewritbility of DL-based OMQs cannot be directly applied to rule-based OMQs; this is further explained in Section 3. Therefore, we develop new semantic characterizations and procedures that are significantly different from those for OMQs based on description logics.

Our analysis aims to develop specially tailored techniques that allow us to understand the problem of checking whether a (frontier-)guarded OMQ is FO-rewritable, and also to pinpoint its computational complexity. Our plan of attack and results can be summarized as follows:

▶ We first focus on the simpler OMQ language based on guarded TGDs and atomic queries, and, in Section 3, we provide a characterization of FO-rewritability that forms the basis for applying tree automata techniques.

▶ We then exploit, in Section 4, standard two-way alternating parity tree automata. In particular, we reduce our problem to the problem of checking the finiteness of the language of an automaton. The reduction relies on a refined version of the characterization of FO-rewritability established in Section 3. This provides a transparent solution to our problem based on standard tools, but it does not lead to an optimal result.

▶ Towards an optimal result, we use, in Section 5, a more sophisticated automata model, known as cost automata. This allows us to show that FO-rewritability for OMQs based on guarded TGDs and atomic queries is in 2ExpTime, and in ExpTime for predicates of bounded arity. Our application of cost automata is quite transparent, which, as above, relies on a refined version of the characterization of FO-rewritability established in Section 3. However, the complexity analysis relies on an intricate result on the boundedness problem for a certain class of cost automata from [Benedikt *et al.*, 2015].

▶ Finally, in Section 6, by using the results of Section 5, we obtain our main results. We show that FO-rewritability is 2ExpTime-complete for OMQs based on guarded TGDs and on frontier-guarded TGDs, no matter whether the actual queries are conjunctive queries, unions thereof, or the simple atomic queries. This remains true when the arity of the predicates is bounded by a constant, with the exception of guarded TGDs and atomic queries, for which the complexity then drops to ExpTime-complete.

In principle, the procedure based on tree automata also provides concrete FO-rewritings when they exist, but it is not tailored towards doing this in an efficient way. Efficiently constructing rewritings is beyond the scope of this work.

2 Preliminaries

Basics. Let $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ be disjoint, countably infinite sets of *constants*, *(labeled) nulls*, and (regular) *variables*, respectively. A *schema* $\mathbf{S}$ is a finite set of relation symbols. The *width* of $\mathbf{S}$, denoted $\mathsf{wd}(\mathbf{S})$, is the maximum arity among all relation symbols of $\mathbf{S}$. We write R/n to denote that the relation symbol R has arity $n \geq 0$. A *term* is either a constant, null, or variable. An *atom* over $\mathbf{S}$ is an expression of the form $R(\bar{v})$, where $R \in \mathbf{S}$ is of arity $n \geq 0$ and $\bar{v}$ is an n-tuple of terms. A *fact* is an atom whose arguments are constants.

Databases. An $\mathbf{S}$-*instance* is a (possibly infinite) set of atoms over the schema $\mathbf{S}$ that contain only constants and nulls, while an $\mathbf{S}$-*database* is a finite set of facts over $\mathbf{S}$. The *active domain* of an instance $\mathfrak{J}$, denoted $\mathsf{adom}(\mathfrak{J})$, consists of all terms occurring in $\mathfrak{J}$. For $X \subseteq \mathsf{adom}(\mathfrak{J})$, we denote by $\mathfrak{J}[X]$ the *subinstance of* $\mathfrak{J}$ *induced by* X, i.e., the set of all facts $R(\bar{a})$ with $\bar{a} \subseteq X$. A *tree decomposition* of an instance $\mathfrak{J}$ is a tuple $\delta = (\mathcal{T}, (X_t)_{t \in T})$, where $\mathcal{T} = (T, E)$ is a (directed) tree with nodes T and edges E, and $(X_t)_{t \in T}$ is a collection of subsets of $\mathsf{adom}(\mathfrak{J})$, called *bags*, such that (i) if $R(\bar{a}) \in \mathfrak{J}$, then there is $v \in T$ such that $\bar{a} \subseteq X_v$, and (ii) for all $a \in \mathsf{adom}(\mathfrak{J})$, the set $\{v \in T \mid a \in X_v\}$ induces a connected subtree of $\mathcal{T}$. The *width* of δ is the maximum size among all bags X_v ($v \in T$) minus one. The *tree-width* of $\mathfrak{J}$, denoted $\mathsf{tw}(\mathfrak{J})$, is $\min\{n \mid \text{there is a tree decomposition of width } n \text{ of } \mathfrak{J}\}$.

Conjunctive queries. A *conjunctive query* (CQ) over $\mathbf{S}$ is a first-order formula of the form $q(\bar{x}) \coloneqq \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where $\bar{x}$ and $\bar{y}$ are tuples of variables, and φ is a conjunction of atoms $R_1(\bar{v}_1) \wedge \cdots \wedge R_m(\bar{v}_m)$ over $\mathbf{S}$ that mention variables from $\bar{x} \cup \bar{y}$ only. The variables $\bar{x}$ are the *answer variables* of $q(\bar{x})$. If $\bar{x}$ is empty then q is a *Boolean CQ*. Let $\mathsf{var}(q)$ be the set of variables occurring in q. As usual, the evaluation of CQs over instances is defined in terms of homomorphisms. A *homomorphism* from q to $\mathfrak{J}$ is a mapping $h\colon \mathsf{var}(q) \to \mathsf{adom}(\mathfrak{J})$ such that $R_i(h(\bar{v}_i)) \in \mathfrak{J}$ for each $1 \leq i \leq m$. We write $\mathfrak{J} \models q(\bar{a})$ to indicate that there is such a homomorphism h such that $h(\bar{x}) = \bar{a}$. The *evaluation of* $q(\bar{x})$ *over* $\mathfrak{J}$, denoted $q(\mathfrak{J})$, is the set of all tuples $\bar{a}$ such that $\mathfrak{J} \models q(\bar{a})$. A *union of conjunctive queries* (UCQ) $q(\bar{x})$ over $\mathbf{S}$ is a disjunction $\bigvee_{i=1}^{n} q_i(\bar{x})$ of CQs over $\mathbf{S}$. The *evaluation of* $q(\bar{x})$ *over* $\mathfrak{J}$, denoted $q(\mathfrak{J})$, is the set of tuples $\bigcup_{1 \leq i \leq n} q_i(\mathfrak{J})$. We write $\mathfrak{J} \models q(\bar{a})$ to indicate that $\mathfrak{J} \models q_i(\bar{a})$ for some $1 \leq i \leq n$. Let CQ be the class of conjunctive queries, and UCQ the class of UCQs. We also write AQ_0 for the class of *atomic queries* of the form $P()$, where P is a 0-ary predicate.

Tuple-generating dependencies. A *tuple-generating dependency* (TGD) (a.k.a. *existential rule*) is a first-order sentence of the form $\tau\colon \forall \bar{x}, \bar{y}\, (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where φ and ψ are conjunctions of atoms that mention only variables. For brevity, we write $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for conjoining atoms. We assume that each variable of $\bar{x}$ is mentioned in ψ. We call φ and ψ the *body* and *head* of the TGD, respectively. The TGD τ is logically equivalent to the sentence $\forall \bar{x}\, (q_\varphi(\bar{x}) \rightarrow q_\psi(\bar{x}))$, where $q_\varphi(\bar{x})$ and $q_\psi(\bar{x})$ are the CQs $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ and $\exists \bar{z}\, \psi(\bar{x}, \bar{z})$, respectively. Thus, an instance $\mathfrak{J}$ *satisfies* τ if $q_\varphi(\mathfrak{J}) \subseteq q_\psi(\mathfrak{J})$. Also, $\mathfrak{J}$ satisfies a set of TGDs $\mathcal{O}$, denoted $\mathfrak{J} \models \mathcal{O}$, if $\mathfrak{J}$ satisfies every $\tau \in \mathcal{O}$. Let TGD be the class of finite sets of TGDs.

Ontology-mediated queries. An *ontology-mediated query* (OMQ) is a triple $Q = (\mathbf{S}, \mathcal{O}, q(\bar{x}))$, where $\mathbf{S}$ is a (non-empty) schema (the *data schema*), $\mathcal{O}$ is a set of TGDs (the *ontology*), and $q(\bar{x})$ is a UCQ over $\mathbf{S} \cup \mathsf{sig}(\mathcal{O})$, where $\mathsf{sig}(\mathcal{O})$ is the set of relation symbols in $\mathcal{O}$. Notice that the ontology $\mathcal{O}$ can introduce relations that are not in $\mathbf{S}$; this allows us to enrich the schema of $q(\bar{x})$. We include $\mathbf{S}$ in the specification of Q to emphasize that Q will be evaluated over $\mathbf{S}$-databases, even though $\mathcal{O}$ and $q(\bar{x})$ may use additional relation symbols.

The semantics of Q is given in terms of certain answers. The *certain answers* to a UCQ $q(\bar{x})$ w.r.t. an $\mathbf{S}$-database $\mathfrak{D}$, and a set $\mathcal{O}$ of TGDs, is the set of all tuples $\bar{a}$ of constants, where $|\bar{a}| = |\bar{x}|$, such that $(\mathfrak{D}, \mathcal{O}) \models q(\bar{a})$, i.e., $\mathfrak{J} \models q(\bar{a})$ for every instance $\mathfrak{J} \supseteq \mathfrak{D}$ that satisfies $\mathcal{O}$. We write $\mathfrak{D} \models Q(\bar{a})$ if $\bar{a}$ is a certain answer to q w.r.t. $\mathfrak{D}$ and $\mathcal{O}$. Moreover, we set $Q(\mathfrak{D}) \coloneqq \{\bar{a} \in \mathsf{adom}(\mathfrak{D})^{|\bar{x}|} \mid \mathfrak{D} \models Q(\bar{a})\}$.

Ontology-mediated query languages. We write (C, Q) for

the class of OMQs $(\mathbf{S}, \mathcal{O}, q)$, where $\mathcal{O}$ falls in the class of TGDs C, and q in the query language Q. The evaluation problem for $(\mathsf{TGD}, \mathsf{UCQ})$, i.e., given a query $Q \in (\mathsf{TGD}, \mathsf{UCQ})$ with data schema $\mathbf{S}$, an $\mathbf{S}$-database $\mathfrak{D}$, and $\bar{a} \in \mathsf{adom}(\mathfrak{D})^{|\bar{x}|}$, to decide whether $\mathfrak{D} \models Q(\bar{a})$, is undecidable; this holds even for $(\mathsf{TGD}, \mathsf{AQ}_0)$ [Calì *et al.*, 2013]. Here we deal with one of the most paradigmatic decidable restrictions, i.e., *guardedness*. A TGD is *guarded* if it has a body atom, called *guard*, that contains all the body variables. Let G be the class of all finite sets of guarded TGDs. A TGD τ is called *frontier-guarded* if its body contains an atom, called *frontier-guard*, that contains the frontier of τ, i.e., the body variables that appear also in the head. We write FG for the class of all finite sets of frontier-guarded TGDs. Roughly, the evaluation problem for $(\mathsf{G}, \mathsf{UCQ})$ and $(\mathsf{FG}, \mathsf{UCQ})$ is decidable since G and FG admit tree-like universal models [Calì *et al.*, 2013].

First-order rewritability. A *first-order (FO) query* over a schema $\mathbf{S}$ is a (function-free) FO formula $\varphi(\bar{x})$, with $\bar{x}$ being its free variables, that uses only relations from $\mathbf{S}$. The *evaluation* of φ over an $\mathbf{S}$-database $\mathfrak{D}$, denoted $\varphi(\mathfrak{D})$, is the set of tuples $\{\bar{a} \in \mathsf{adom}(\mathfrak{D})^{|\bar{x}|} \mid \mathfrak{D} \models \varphi(\bar{a})\}$; $\models$ denotes the standard notion of satisfaction for FO. An OMQ $Q = (\mathbf{S}, \mathcal{O}, q(\bar{x}))$ is *FO-rewritable* if there exists a (finite) FO query $\varphi_Q(\bar{x})$ over $\mathbf{S}$ that is *equivalent* to Q, i.e., for every $\mathbf{S}$-database $\mathfrak{D}$ it is the case that $Q(\mathfrak{D}) = \varphi_Q(\mathfrak{D})$. We call $\varphi_Q(\bar{x})$ an *FO-rewriting of* Q. A fundamental task for an OMQ language (C, Q), where C is a class of TGDs and Q is a class of queries, is deciding first-order rewritability:

PROBLEM :	$\mathsf{FORew}(\mathsf{C}, \mathsf{Q})$
INPUT :	An OMQ $Q \in (\mathsf{C}, \mathsf{Q})$.
QUESTION :	Is it the case that Q is FO-rewritable?

First-order rewritability of $(\mathsf{FG}, \mathsf{UCQ})$-queries. As shown by the following example, there exist $(\mathsf{G}, \mathsf{CQ})$ queries (and thus, $(\mathsf{FG}, \mathsf{UCQ})$ queries) that are not FO-rewritable.

Example 1. Consider the OMQ $Q = (\mathbf{S}, \mathcal{O}, q) \in (\mathsf{G}, \mathsf{CQ})$, where $\mathbf{S} = \{S/3, A/1, B/1\}$, $\mathcal{O}$ consists of

$$\begin{array}{rcl} S(x,y,z), A(z) & \rightarrow & R(x,z), \\ S(x,y,z), R(x,z) & \rightarrow & R(x,y), \end{array}$$

and $q = \exists x, y, z\,(S(x,y,z) \wedge R(x,z) \wedge B(y))$. Intuitively, an FO-rewriting of Q should check for the existence of a set of atoms $\{S(c, a_i, a_{i-1})\}_{1 \leq i \leq k}$, among others, for $k \geq 0$. However, since there is no upper bound for k, this cannot be done via a finite FO-query, and thus, Q is not FO-rewritable. A proof that Q is not FO-rewritable is given below. ∎

On the other hand, there are (frontier-)guarded OMQs that are FO-rewritable; e.g., the OMQ obtained from the query Q in Example 1 by adding $A(z)$ to q is FO-rewritable with $\exists x, y, z\,(S(x,y,z) \wedge B(y) \wedge A(z))$ being an FO-rewriting.

3 Semantic Characterization

We proceed to give a characterization of FO-rewritability of OMQs from $(\mathsf{G}, \mathsf{AQ}_0)$ in terms of the existence of certain tree-like databases. Our characterization is related to, but different from characterizations used for OMQs based on DLs such as $\mathcal{EL}$ and $\mathcal{ELI}$ [Bienvenu *et al.*, 2013; 2016].

The characterizations in [Bienvenu *et al.*, 2013; 2016] essentially state that a unary OMQ Q is FO-rewritable iff there is a bound k such that, whenever the root of a tree-shaped database $\mathfrak{D}$ is returned as an answer to Q, then this is already true for the restriction of $\mathfrak{D}$ up to depth k. The proof of the (contrapositive of the) "only if" direction uses a locality argument: if there is no such bound k, then this is witnessed by an infinite sequence of deeper and deeper tree databases that establish non-locality of Q. For guarded TGDs, we would have to replace tree-shaped databases with databases of bounded tree-width. However, increasing depth of tree decompositions does not correspond to increasing distance in the Gaifman graph, and thus, does not establish non-locality. We therefore depart from imposing a bound on the depth, and instead we impose a bound on the number of facts, as detailed below.

It is also interesting to note that, while it is implicit in [Bienvenu *et al.*, 2016] that an OMQ based on $\mathcal{ELI}$ and CQs is FO-rewritable iff it is Gaifman local, there exists an OMQ from $(\mathsf{G}, \mathsf{CQ})$ that is Gaifman local, but not FO-rewritable. Such an OMQ is the one obtained from the query Q given in Example 1, by removing the existential quantification on the variable x in the CQ q, i.e., converting q into a unary CQ.

Theorem 1. *Consider an OMQ $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ with data schema $\mathbf{S}$. The following are equivalent:*

1. *Q is FO-rewritable.*
2. *There is a $k \geq 0$ such that, for every $\mathbf{S}$-database $\mathfrak{D}$ of tree-width at most $\mathsf{wd}(\mathbf{S}) - 1$, if $\mathfrak{D} \models Q$, then there is a $\mathfrak{D}' \subseteq \mathfrak{D}$ with at most k facts such that $\mathfrak{D}' \models Q$.*

For (1) $\Rightarrow$ (2) we exploit the fact that, if $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ is FO-rewritable, then it can be expressed as a UCQ q_Q. This follows from the fact that OMQs from $(\mathsf{G}, \mathsf{AQ}_0)$ are preserved under homomorphisms [Bienvenu *et al.*, 2014], and Rossman's Theorem stating that an FO query is preserved under homomorphisms over finite instances iff it is equivalent to a UCQ [Rossman, 2008]. It is then easy to show that (2) holds with k being the size of the largest disjunct of the UCQ q_Q. For (2) $\Rightarrow$ (1), we use the fact that, if there is an $\mathbf{S}$-database $\mathfrak{D}$ that entails Q, then there exists one of tree-width at most $\mathsf{wd}(\mathbf{S}) - 1$ that entails Q, and can be mapped to $\mathfrak{D}$. The next example illustrates Theorem 1.

Example 2. Consider the OMQ $Q = (\mathbf{S}, \mathcal{O}, P) \in (\mathsf{G}, \mathsf{AQ}_0)$, where $\mathbf{S} = \{S/3, A/1, B/1\}$, and $\mathcal{O}$ consists of the TGDs given in Example 1 plus the guarded TGD

$$S(x,y,z), R(x,z), B(y) \rightarrow P,$$

which is essentially the CQ q from Example 1. It is easy to verify that, for an arbitrary $k \geq 0$, the $\mathbf{S}$-database

$$\mathfrak{D}_k = \{A(a_0), S(c, a_1, a_0), \ldots, S(c, a_{k-1}, a_{k-2}), B(a_{k-1})\}$$

of tree-width $\mathsf{wd}(\mathbf{S}) - 1 = 2$ is such that $\mathfrak{D}_k \models Q$, but for every $\mathfrak{D}' \subset \mathfrak{D}_k$ with at most k facts, $\mathfrak{D}' \not\models Q$. Thus, by Theorem 1, Q is not FO-rewritable. ∎

4 Alternating Tree Automata Approach

In this section, we exploit the well-known algorithmic tool of *two-way alternating parity tree automata* (2ATA) over finite trees of bounded degree (see, e.g., [Cosmadakis *et al.*, 1988]), and prove that $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ can be solved in elementary time. Although this result is not optimal, our construction provides a transparent solution to $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ based on standard tools. This is in contrast with previous studies on closely related problems for guarded logics, in which all elementary bounds heavily rely on the use of intricate results on cost automata [Blumensath *et al.*, 2014; Benedikt *et al.*, 2015]. We also apply such results later, but only in order to pinpoint the exact complexity of $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$.

The idea behind our solution to $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ is, given a query $Q \in (\mathsf{G}, \mathsf{AQ}_0)$, to devise a 2ATA $\mathcal{B}_Q$ such that Q is FO-rewritable iff the language accepted by $\mathcal{B}_Q$ is finite. This is a standard idea with roots in the study of the boundedness problem for *monadic Datalog* (see e.g., [Vardi, 1992]). In particular, our main result establishes the following:

Theorem 2. *Let $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ with data schema $\mathbf{S}$. There is a 2ATA $\mathcal{B}_Q$ on trees of degree at most $2^{\mathsf{wd}(\mathbf{S})}$ such that Q is FO-rewritable iff the language of $\mathcal{B}_Q$ is finite. The state set of $\mathcal{B}_Q$ is of double exponential size in $\mathsf{wd}(\mathbf{S})$, and of exponential size in $|\mathbf{S} \cup \mathsf{sig}(\mathcal{O})|$. Furthermore, $\mathcal{B}_Q$ can be constructed in double exponential time in the size of Q.*

As a corollary to Theorem 2 we obtain the following result:

Corollary 3. $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ *is in* 3ExpTime*, and in* 2ExpTime *for predicates of bounded arity.*

From Theorem 2, to check whether a query $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ is FO-rewritable, it suffices to check that the language of $\mathcal{B}_Q$ is finite. The latter is done by first converting $\mathcal{B}_Q$ into a nondeterministic bottom-up tree automaton $\mathcal{B}'_Q$; see, e.g., [Vardi, 1998]. This incurs an exponential blowup, and thus, $\mathcal{B}'_Q$ has triple exponentially many states. We then check the finiteness of the language of $\mathcal{B}'_Q$ in polynomial time in the size of $\mathcal{B}'_Q$ by applying a standard reachability analysis; see [Vardi, 1992]. For predicates of bounded arity, a similar argument as above provides a double exponential time upper bound.

In the rest of Section 4 we explain the proof of Theorem 2. The intuitive idea is to construct a 2ATA $\mathcal{B}_Q$ whose language corresponds to suitable encodings of databases $\mathfrak{D}$ of bounded tree-width that "minimally" satisfy Q, i.e., $\mathfrak{D} \models Q$, but if we remove any atom from $\mathfrak{D}$, then Q is no longer satisfied.

A refined semantic characterization. In order to apply an approach based on 2ATA, it is essential to revisit the semantic characterization provided by Theorem 1. To this end, we need to introduce some auxiliary terminology.

Let $\mathfrak{D}$ be a database, and $\delta = (\mathcal{T}, (X_v)_{v \in T})$, where $\mathcal{T} = (T, E)$, a tree decomposition of $\mathfrak{D}$. An *adornment* of the pair $(\mathfrak{D}, \delta)$ is a function $\eta \colon T \to 2^{\mathfrak{D}}$ such that $\eta(v) \subseteq \mathfrak{D}[X_v]$ for all $v \in T$, and $\bigcup_{v \in T} \eta(v) = \mathfrak{D}$. Therefore, the pair (δ, η) can be viewed as a representation of the database $\mathfrak{D}$ along with a tree decomposition of it. For the intended characterization, it is important that this representation is free of redundancies, formalized as follows. We say that δ is *η-simple* if $|\eta(v)| \leq 1$ for all $v \in T$, and non-empty η-labels are unique, that is, $\eta(v) \neq \eta(w)$ for all distinct $v, w \in T$ with $\eta(v)$ and $\eta(w)$ non-empty. Nodes $v \in T$ with $\eta(v)$ empty, called *white* from now on, are required since we might not have a (unique!) fact available to label them. Note, though, that white nodes v are still associated with a non-empty set of constants from $\mathfrak{D}$ via X_v. All other nodes are called *black*. While δ being η-simple avoids redundancies that are due to a fact occurring in the label of multiple black nodes, additional redundancies may arise from the inflationary use of white nodes. We say that a node $v \in T$ is *η-well-colored* if it is black, or it has at least two successors and all its successors are η-well-colored. We say that δ is *η-well-colored* if every node in T is η-well-colored. For example, δ is not η-well-colored if it has a white leaf, or if it has a white node and its single successor is also white. Informally, requiring δ to be η-well-colored makes it impossible to blow up the tree by introducing white nodes without introducing black nodes. For $i \in \{1, 2\}$, let $\mathfrak{D}_i$ be a database, δ_i a tree decomposition of $\mathfrak{D}_i$, and η_i an adornment of $(\mathfrak{D}_i, \delta_i)$. We say that $(\mathfrak{D}_1, \delta_1, \eta_1)$ and $(\mathfrak{D}_2, \delta_2, \eta_2)$ are *isomorphic* if the latter can be obtained from the former by consistenly renaming constants in $\mathfrak{D}_1$ and tree nodes in δ_1.

We are now ready to revisit the characterization of FO-rewritability for OMQs from $(\mathsf{G}, \mathsf{AQ}_0)$ given in Theorem 1.

Theorem 4. *Consider an OMQ $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ with data schema $\mathbf{S}$. The following are equivalent:*

1. *Condition 2 from Theorem 1 is satisfied.*
2. *There are finitely many non-isomorphic triples $(\mathfrak{D}, \delta, \eta)$, where $\mathfrak{D}$ is an $\mathbf{S}$-database, δ a tree decomposition of $\mathfrak{D}$ of width at most $\mathsf{wd}(\mathbf{S}) - 1$, and η an adornment of $(\mathfrak{D}, \delta)$, such that*
   (a) δ is η-simple and η-well-colored,
   (b) $\mathfrak{D} \models Q$, and
   (c) for every $\alpha \in \mathfrak{D}$, it is the case that $\mathfrak{D} \setminus \{\alpha\} \not\models Q$.

Devising automata. We proceed to discuss how the 2ATA announced in Theorem 2 is constructed. Consider an OMQ $Q = (\mathbf{S}, \mathcal{O}, P)$ from $(\mathsf{G}, \mathsf{AQ}_0)$. Our goal is to devise an automaton $\mathcal{B}_Q$ whose language is finite iff Condition 2 from Theorem 4 is satisfied. By Theorems 1 and 4, Q is then FO-rewritable iff the language of $\mathcal{B}_Q$ is finite.

The 2ATA $\mathcal{B}_Q$ will be the intersection of several automata that check the properties stated in item 2 of Theorem 4. But first we need to say a few words about tree encodings. Let Γ be a finite alphabet, and let $(\mathbb{N} \setminus \{0\})^*$ denote the set of all finite words of positive integers, including the empty word. A *finite Γ-labeled tree* is a partial function $t \colon (\mathbb{N} \setminus \{0\})^* \to \Gamma$ such that the domain of t is finite and prefix-closed. Moreover, if $v \cdot i$ belongs to the domain of t, then $v \cdot (i-1)$ also belongs to the domain of t. In fact, the elements in the domain of t identify the nodes of the tree. It can be shown that an $\mathbf{S}$-database $\mathfrak{D}$, a tree decomposition δ of $\mathfrak{D}$ of width $w - 1$, and an adornment η of $(\mathfrak{D}, \delta)$, can be encoded as a $\Gamma_{\mathbf{S},w}$-labeled tree t of degree at most 2^w, where $\Gamma_{\mathbf{S},w}$ is an alphabet of size double exponential in w and exponential in $\mathbf{S}$, such that each node of δ corresponds to exactly one node of t and vice versa. Although every $\mathfrak{D}$ can be encoded into a $\Gamma_{\mathbf{S},w}$-labeled tree t, the converse is not true in general. However, it is possible to

define certain syntactic consistency conditions such that every *consistent* $\Gamma_{\mathbf{S},w}$-labeled t can be decoded into an $\mathbf{S}$-database, denoted $[\![t]\!]$, whose tree-width is at most w. We are going to abbreviate the alphabet $\Gamma_{\mathbf{S},\mathsf{wd}(\mathbf{S})}$ by $\Gamma_{\mathbf{S}}$.

Lemma 5. *There is a 2ATA $\mathcal{C}_{\mathbf{S}}$ that accepts a $\Gamma_{\mathbf{S}}$-labeled tree t iff t is consistent. The number of states of $\mathcal{C}_{\mathbf{S}}$ is constant. $\mathcal{C}_{\mathbf{S}}$ can be constructed in polynomial time in the size of $\Gamma_{\mathbf{S}}$.*

Since a $\Gamma_{\mathbf{S}}$-labeled tree incorporates the information about an adornment, the notions of being well-colored and simple can be naturally defined for $\Gamma_{\mathbf{S}}$-labeled trees. Then:

Lemma 6. *There is a 2ATA $\mathcal{R}_{\mathbf{S}}$ that accepts a consistent $\Gamma_{\mathbf{S}}$-labeled tree iff it is well-colored and simple. The number of states of $\mathcal{R}_{\mathbf{S}}$ is exponential in $\mathsf{wd}(\mathbf{S})$ and linear in $|\mathbf{S}|$. $\mathcal{R}_{\mathbf{S}}$ can be constructed in polynomial time in the size of $\Gamma_{\mathbf{S}}$.*

Concerning property 2(b) of Theorem 4, we can devise a 2ATA that accepts those trees whose decoding satisfies Q:

Lemma 7. *There is a 2ATA $\mathcal{A}_Q$ that accepts a consistent $\Gamma_{\mathbf{S}}$-labeled tree iff $[\![t]\!] \models Q$. The number of states of $\mathcal{A}_Q$ is exponential in $\mathsf{wd}(\mathbf{S})$ and linear in $|\mathbf{S} \cup \mathsf{sig}(\mathcal{O})|$. $\mathcal{A}_Q$ can be constructed in double exponential time in the size of Q.*

The crucial task is to check condition 2(c) of Theorem 4, which states the key minimality criterion. Unfortunately, this involves an extra exponential blowup:

Lemma 8. *There is a 2ATA $\mathcal{M}_Q$ that accepts a consistent $\Gamma_{\mathbf{S}}$-labeled tree t iff $[\![t]\!] \setminus \{\alpha\} \not\models Q$ for all $\alpha \in [\![t]\!]$. The state set of $\mathcal{M}_Q$ is of double exponential size in $\mathsf{wd}(\mathbf{S})$, and of exponential size in $|\mathbf{S} \cup \mathsf{sig}(\mathcal{O})|$. Furthermore, $\mathcal{M}_Q$ can be constructed in double exponential time in the size of Q.*

Let us briefly explain how $\mathcal{M}_Q$ is constructed. This will expose the source of the extra exponential blowup, which prevents us from obtaining an optimal complexity upper bound for $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$. We first construct a 2ATA $\mathcal{D}_Q$ that runs on $\Lambda_{\mathbf{S}}$-labeled trees, where $\Lambda_{\mathbf{S}}$ is an alphabet that extends $\Gamma_{\mathbf{S}}$ with auxiliary symbols that allow us to tag some facts in the input tree. In particular, $\mathcal{D}_Q$ accepts a tree t iff t is consistent, there is at least one tagged fact, and $[\![t]\!]^- \models Q$ where $[\![t]\!]^-$ is obtained from $[\![t]\!]$ by removing the tagged facts. Having $\mathcal{D}_Q$ in place, we can then construct a 2ATA $\exists\mathcal{D}_Q$ that accepts a $\Gamma_{\mathbf{S}}$-labeled tree t if there is a way to tag some of its facts so as to obtain a $\Lambda_{\mathbf{S}}$-labeled tree t' with $[\![t']\!]^- \models Q$. This is achieved by applying the projection operator on $\mathcal{D}_Q$. Since for 2ATAs projection involves an exponential blowup and $\mathcal{D}_Q$ already has exponentially many states, $\exists\mathcal{D}_Q$ has double exponentially many. It should be clear now that $\mathcal{M}_Q$ is the complement of $\exists\mathcal{D}_Q$, and we recall that complementation of 2ATAs can be done in polynomial time.

The desired automaton $\mathcal{B}_Q$ is obtained by intersecting the 2ATAs in Lemmas 5 to 8. Since the intersection of 2ATA is feasible in polynomial time, $\mathcal{B}_Q$ can be constructed in double exponential time in the size of Q.

5 Cost Automata Approach

We proceed to study $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ using the more sophisticated model of cost automata. This allows us to improve the complexity of the problem obtained in Corollary 3 as follows:

Theorem 9. $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ *is in* 2ExpTime*, and in* ExpTime *for predicates of bounded arity.*

As in the previous approach, we develop a semantic characterization that relies on a minimality criterion for trees accepted by cost automata. The extra features provided by cost automata allow us to deal with such a minimality criterion in a more efficient way than standard 2ATA. While our application of cost automata is transparent, the complexity analysis relies on an intricate result on the boundedness problem for a certain class of cost automata from [Benedikt *et al.*, 2015]. Before we proceed further, let us provide a brief overview of the cost automata model that we are going to use.

Cost automata models. Cost automata extend traditional automata by providing counters that can be manipulated at each transition. Instead of assigning a Boolean value to each input structure (indicating whether it is accepted or not), these automata assign a value from $\mathbb{N}_\infty := \mathbb{N} \cup \{\infty\}$ to each input.

Here, we focus on cost automata that work on finite trees of unbounded degree, and allow for two-way movements; in fact, the automata that we need are those that extend 2ATA over finite trees with a *single* counter. The operation of such an automaton $\mathcal{A}$ on each input t will be viewed as a two-player *cost game* $\mathcal{G}(\mathcal{A}, t)$ between players Eve and Adam. Recall that the acceptance of an input tree for a conventional 2ATA can be formalized via a two-player game as well. However, instead of the parity acceptance condition for 2ATA, plays in the cost game between Eve and Adam will be assigned costs, and the cost automaton specifies via an *objective* whether Eve's goal is to minimize or maximize that cost. In case of a minimizing (resp., maximizing) objective, a strategy ξ of Eve in the cost game $\mathcal{G}(\mathcal{A}, t)$ is n-*winning* if any play of Adam consistent with ξ has cost at most n (resp., at least n). Given an input tree t, one then defines the *value of t in $\mathcal{A}$* as

$$[\![\mathcal{A}]\!](t) := \mathrm{op}\{n \mid \text{Eve has an } n\text{-winning strategy in } \mathcal{G}(\mathcal{A}, t)\},$$

where $\mathrm{op} = \inf$ (resp., $\mathrm{op} = \sup$) in case Eve's objective is to minimize (resp., maximize). Therefore, $[\![\mathcal{A}]\!]$ defines a function from the domain of input trees to $\mathbb{N}_\infty$. We call functions of that type *cost functions*. A key property of such functions is boundedness. We say that $[\![\mathcal{A}]\!]$ is *bounded* if there exists an $n \in \mathbb{N}$ such that $[\![\mathcal{A}]\!](t) \leq n$ for every input tree t.

We employ automata with a single counter, where Eve's objective is to minimize the cost, while satisfying the parity condition. Such an automaton is known in the literature as $\mathrm{dist} \wedge \mathrm{parity}$-*automaton* [Benedikt *et al.*, 2015]. To navigate in the tree, it may use the directions $\{0, \updownarrow\}$, where 0 indicates that the automaton should stay in the current node, and $\updownarrow$ means that the automaton may move to an arbitrary neighboring node, including the parent. For this type of automaton, we can decide whether its cost function is bounded [Benedikt *et al.*, 2015; Colcombet and Fijalkow, 2016]. As usual, $\|\mathcal{A}\|$ denotes the size of $\mathcal{A}$. Then:

Theorem 10. *There is a polynomial f such that, for every $\mathrm{dist} \wedge \mathrm{parity}$-automaton $\mathcal{A}$ using priorities $\{0, 1\}$ for the parity acceptance condition, the boundedness for $[\![\mathcal{A}]\!]$ is decidable in time $\|\mathcal{A}\|^{f(m)}$, where m is the number of states of $\mathcal{A}$.*

Our goal is to reduce $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ to the boundedness problem for $\mathrm{dist} \wedge \mathrm{parity}$-automata.

A refined semantic characterization. We first need to revisit the semantic characterization provided by Theorem 1.

Consider an $\mathbf{S}$-database $\mathfrak{D}$, and a query $Q = (\mathbf{S}, \mathcal{O}, P) \in (\mathsf{G}, \mathsf{AQ}_0)$. Let $k_Q := |\mathbf{S} \cup \mathsf{sig}(\mathcal{O})| \cdot w^w$, where $w := \mathsf{wd}(\mathbf{S} \cup \mathsf{sig}(\mathcal{O}))$. A *derivation tree* for $\mathfrak{D}$ and Q is a labeled k_Q-ary tree $\mathcal{T}$, with η being a node labeling function that assigns facts $R(\bar{a})$, where $R \in \mathbf{S} \cup \mathsf{sig}(\mathcal{O})$ and $\bar{a} \subseteq \mathsf{adom}(\mathfrak{D})$, to its nodes, that satisfies the following conditions:

1. For the root node v of $\mathcal{T}$, $\eta(v) = P$.
2. For each leaf node v of $\mathcal{T}$, $\eta(v) \in \mathfrak{D}$.
3. For each non-leaf node v of $\mathcal{T}$, with $u_1, \ldots, u_k$ being its children, $(\{\eta(u_1), \ldots, \eta(u_k)\}, \mathcal{O}) \models \eta(v)$.

Roughly, $\mathcal{T}$ describes how the 0-ary predicate P can be entailed from $\mathfrak{D}$ and $\mathcal{O}$. In fact, it is easy to show that $\mathfrak{D} \models Q$ iff there is a derivation tree for $\mathfrak{D}$ and Q. The *height* of $\mathcal{T}$, denoted $\mathrm{hgt}(\mathcal{T})$, is the maximum length of a branch in $\mathcal{T}$, i.e., of a path from the root to a leaf node. Assuming that $\mathfrak{D} \models Q$, the *cost of* $\mathfrak{D}$ *w.r.t.* Q, denoted $\mathrm{cost}(\mathfrak{D}, Q)$, is defined as

$$\min\{\mathrm{hgt}(\mathcal{T}) \mid \mathcal{T} \text{ is a derivation tree for } \mathfrak{D} \text{ and } Q\},$$

while the *cost* of Q, denoted $\mathrm{cost}(Q)$, is defined as

$$\sup\{\mathrm{cost}(\mathfrak{D}, Q) \mid \mathfrak{D} \models Q, \mathfrak{D} \text{ is an } \mathbf{S}\text{-database with } \mathsf{tw}(\mathfrak{D}) \leq \mathsf{wd}(\mathbf{S}) - 1\}.$$

In other words, the cost of Q is the *least upper bound* of the height over all derivation trees for all $\mathbf{S}$-databases $\mathfrak{D}$ of treewidth at most $\mathsf{wd}(\mathbf{S}) - 1$ such that $\mathfrak{D} \models Q$. If there is no such a database, then the cost of Q is zero since $\sup \emptyset := 0$. Actually, $\mathrm{cost}(Q) = 0$ indicates that Q is unsatisfiable, which in turn means that Q is trivially FO-rewritable.

Having the notion of the cost of an OMQ from $(\mathsf{G}, \mathsf{AQ}_0)$ in place, it should not be difficult to see how we can refine the semantic characterization provided by Theorem 1.

Theorem 11. *Consider an OMQ $Q \in (\mathsf{G}, \mathsf{AQ}_0)$ with data schema $\mathbf{S}$. The following are equivalent:*

1. *Condition 2 from Theorem 1 is satisfied.*
2. *$\mathrm{cost}(Q)$ is finite.*

Devising automata. We briefly describe how we can use cost automata in order to devise an algorithm for $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ that runs in double exponential time.

Consider an OMQ $Q = (\mathbf{S}, \mathcal{O}, P) \in (\mathsf{G}, \mathsf{AQ}_0)$. Our goal is to devise a $\mathrm{dist} \wedge \mathrm{parity}$-automaton $\mathcal{B}_Q$ such that the cost function $[\![\mathcal{B}_Q]\!]$ is bounded iff $\mathrm{cost}(Q)$ is finite. Therefore, by Theorems 1 and 11, to check whether Q is FO-rewritable we simply need to check if $[\![\mathcal{B}_Q]\!]$ is bounded, which, by Theorem 10, can be done in exponential time in the size of $\mathcal{B}_Q$. The input trees to our automata will be over the same alphabet $\Gamma_{\mathbf{S}}$ that is used to encode tree-like $\mathbf{S}$-databases in Section 4. Recall that for a $\mathrm{dist} \wedge \mathrm{parity}$-automaton $\mathcal{A}$, the cost function $[\![\mathcal{A}]\!]$ is bounded over a certain class $\mathcal{C}$ of trees if there is an $n \in \mathbb{N}$ such that $[\![\mathcal{A}]\!](t) \leq n$ for every input tree $t \in \mathcal{C}$. Then:

Lemma 12. *There is a $\mathrm{dist} \wedge \mathrm{parity}$-automaton $\mathcal{H}_Q$ such that $[\![\mathcal{H}_Q]\!]$ is bounded over consistent $\Gamma_{\mathbf{S}}$-labeled trees iff $\mathrm{cost}(Q)$ is finite. The number of states of $\mathcal{H}_Q$ is exponential in $\mathsf{wd}(\mathbf{S})$, and polynomial in $|\mathbf{S} \cup \mathsf{sig}(\mathcal{O})|$. Moreover, $\mathcal{H}_Q$ can be constructed in double exponential time in the size of Q.*

The automaton $\mathcal{H}_Q$ is built in such a way that, on an input tree t, Eve has an n-winning strategy in $\mathcal{G}(\mathcal{H}_Q, t)$ iff there is a derivation tree for $[\![t]\!]$ and Q of height at most n. Thus, Eve tries to construct derivation trees of minimal height. The counter is used to count the height of the derivation tree.

Having this automaton in place, we can now complete the proof of Theorem 9. The desired $\mathrm{dist} \wedge \mathrm{parity}$-automaton $\mathcal{B}_Q$ is defined as $\mathcal{C}'_{\mathbf{S}} \cap \mathcal{H}_Q$, where $\mathcal{C}'_{\mathbf{S}}$ is similar to the 2ATA $\mathcal{C}_{\mathbf{S}}$ (in Lemma 5) that checks for consistency of $\Gamma_{\mathbf{S}}$-labeled trees of bounded degree. Notice that $\mathcal{C}'_{\mathbf{S}}$ is essentially a $\mathrm{dist} \wedge \mathrm{parity}$-automaton that assigns zero (resp., ∞) to input trees that are consistent (resp., inconsistent), and thus, $\mathcal{C}'_{\mathbf{S}} \cap \mathcal{H}_Q$ is well-defined. Since the intersection of $\mathrm{dist} \wedge \mathrm{parity}$-automata is feasible in polynomial time, Lemma 5 and Lemma 12 imply that $\mathcal{B}_Q$ has exponentially many states, and it can be constructed in double exponential time. Lemma 12 implies also that $[\![\mathcal{B}_Q]\!]$ is bounded iff $\mathrm{cost}(Q)$ is finite. It remains to show that the boundedness of $[\![\mathcal{B}_Q]\!]$ can be checked in double exponential time. By Theorem 10, there is a polynomial f such that the latter task can be carried out in time $\|\mathcal{B}_Q\|^{f(m)}$, where m is the number of states of $\mathcal{B}_Q$, and the claim follows. For predicates of bounded arity, we provide a similar analysis.

6 Frontier-Guarded OMQs

The goal of this section is to show the following result:

Theorem 13. *It holds that:*

- $\mathsf{FORew}(\mathsf{FG}, \mathsf{Q})$ *is* 2ExpTime*-complete, for each* $\mathsf{Q} \in \{\mathsf{UCQ}, \mathsf{CQ}, \mathsf{AQ}_0\}$*, even for predicates of bounded arity.*
- $\mathsf{FORew}(\mathsf{G}, \mathsf{Q})$ *is* 2ExpTime*-complete, for each* $\mathsf{Q} \in \{\mathsf{UCQ}, \mathsf{CQ}\}$*, even for predicates of bounded arity.*
- $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ *is* 2ExpTime*-complete. Moreover, for predicates of bounded arity it is* ExpTime*-complete.*

Lower bounds. The 2ExpTime-hardness in the first and the second items is inherited from [Bienvenu *et al.*, 2016], where it is shown that deciding FO-rewritability for OMQs based on $\mathcal{ELI}$ and CQs is 2ExpTime-hard. For the 2ExpTime-hardness in the third item we exploit the fact that containment for OMQs from $(\mathsf{G}, \mathsf{AQ}_0)$ is 2ExpTime-hard, even if the right-hand side query is FO-rewritable; this is implicit in [Barceló *et al.*, 2014]. Finally, the ExpTime-hardness in the third item is inherited from [Bienvenu *et al.*, 2013], where it is shown that deciding FO-rewritability for OMQs based on $\mathcal{EL}$ and atomic queries is ExpTime-hard.

Upper bounds. The fact that for predicates of bounded arity $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ is in ExpTime is obtained from Theorem 9. It remains to show that $\mathsf{FORew}(\mathsf{FG}, \mathsf{UCQ})$ is in 2ExpTime. We reduce $\mathsf{FORew}(\mathsf{FG}, \mathsf{UCQ})$ in polynomial time to $\mathsf{FORew}(\mathsf{FG}, \mathsf{AQ}_0)$, and then show that the latter is in 2ExpTime. This reduction relies on a construction from [Bienvenu *et al.*, 2016], which allows us to reduce $\mathsf{FORew}(\mathsf{FG}, \mathsf{UCQ})$ to

$\mathsf{FORew}(\mathsf{FG}, \mathsf{UBCQ})$ with UBCQ being the class of union of Boolean CQs, and the fact that a Boolean CQ can be seen as a frontier-guarded TGD. To show that $\mathsf{FORew}(\mathsf{FG}, \mathsf{AQ}_0)$ is in 2ExpTime, we reduce it to $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$, and then apply Theorem 9. This relies on *treeification*, and is inspired by a translation of guarded negation fixed-point sentences into guarded fixed-point sentences [Bárány *et al.*, 2015]. Our reduction may give rise to exponentially many guarded TGDs, but the arity is increased only polynomially. Since the procedure for $\mathsf{FORew}(\mathsf{G}, \mathsf{AQ}_0)$ provided by Theorem 9 is double exponential only in the arity of the schema the claim follows.

7 Future Work

The procedure based on 2ATA provides an FO-rewriting in case the input OMQ admits one, but it is not tailored towards doing this in an efficient way. Our next step is to exploit the techniques developed in this work for devising practically efficient algorithms for constructing FO-rewritings.

Acknowledgements

Barceló is funded by the Millennium Institute for Foundational Research on Data and Fondecyt grant 1170109. Berger is funded by the FWF project W1255-N23 and a DOC fellowship of the Austrian Academy of Sciences. Lutz is funded by the ERC grant 647289 "CODA". Pieris is funded by the EPSRC programme grant EP/M025268/ "VADA".

References

[Ajtai and Gurevich, 1994] Miklós Ajtai and Yuri Gurevich. Datalog vs first-order logic. *J. Comput. Syst. Sci.*, 49(3):562–588, 1994.

[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.*, 2015] Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015.

[Barceló *et al.*, 2014] Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Does query evaluation tractability help query containment? In *PODS*, pages 188–199, 2014.

[Benedikt *et al.*, 2015] Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, pages 293–304, 2015.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First-order rewritability of atomic queries in horn description logics. In *IJCAI*, 2013.

[Bienvenu *et al.*, 2014] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

[Bienvenu *et al.*, 2016] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in horn description logics. In *IJCAI*, pages 965–971, 2016.

[Blumensath *et al.*, 2014] Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science*, 10(3), 2014.

[Calì *et al.*, 2012a] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[Calì *et al.*, 2012b] 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.

[Colcombet and Fijalkow, 2016] Thomas Colcombet and Nathanaël Fijalkow. The bridge between regular cost functions and omega-regular languages. In *ICALP*, pages 126:1–126:13, 2016.

[Cosmadakis *et al.*, 1988] Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *STOC*, pages 477–490, 1988.

[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.

[Gaifman *et al.*, 1993] Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, 1993.

[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.

[Lutz and Sabellek, 2017] Carsten Lutz and Leif Sabellek. Ontology-mediated querying with the description logic EL: trichotomy and linear datalog rewritability. In *IJCAI*, pages 1181–1187, 2017.

[Poggi *et al.*, 2008] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[Rossman, 2008] Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008.

[Vardi, 1992] Moshe Y. Vardi. Automata theory for database theoreticans. In *Theoretical Studies in Computer Science*, pages 153–180, 1992.

[Vardi, 1998] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.