Towards Efficient Reasoning under Guarded-based Disjunctive Existential Rules

Pierre Bourhis, Michael Morak, Andréas Pieris

Towards Efficient Reasoning Under Guarded-based Disjunctive Existential Rules

Pierre Bourhis[1], Michael Morak[2], and Andreas Pieris[2]

[1]CNRS LIFL Université Lille 1/INRIA Lille, France
[2]Department of Computer Science, University of Oxford, UK

pierre.bourhis@univ-lille1.fr
{michael.morak, andreas.pieris}@cs.ox.ac.uk

Abstract. The complete picture of the complexity of answering (unions of) conjunctive queries under the main guarded-based classes of disjunctive existential rules has been recently settled. It has been shown that the problem is very hard, namely 2ExpTime-complete, even for fixed sets of rules expressed in lightweight formalisms. This gives rise to the question whether its complexity can be reduced by restricting the query language. Several subclasses of conjunctive queries have been proposed with the aim of reducing the complexity of classical database problems such as query evaluation and query containment. Three of the most prominent subclasses of this kind are queries of bounded hypertree-width, queries of bounded treewidth and acyclic queries. The central objective of the present paper is to understand whether the above query languages have a positive impact on the complexity of query answering under the main guarded-based classes of disjunctive existential rules.
We show that (unions of) conjunctive queries of bounded hypertree-width and of bounded treewidth do not reduce the complexity of our problem, even if we focus on predicates of bounded arity, or on fixed sets of disjunctive existential rules. Regarding acyclic queries, although our problem remains 2ExpTime-complete in general, in some relevant settings the complexity reduces to ExpTime-complete; in fact, this requires to bound the arity of the predicates, and for some expressive guarded-based formalisms, to fix the set of rules.

1 Introduction

Rule-based languages lie at the core of several areas of central importance to artificial intelligence and databases, such as knowledge representation and reasoning, data exchange and integration, and web data extraction. A prominent rule-based formalism, originally intended for expressing complex recursive queries over relational databases, is Datalog, i.e., function-free first-order Horn logic. As already criticized in [28], the main weakness of this language for representing knowledge is its inability to infer the existence of new objects which are not explicitly stated in the extensional data set.

Existential rules, a.k.a. *tuple-generating dependencies (TGDs)* and *Datalog$^{\pm}$ rules*, overcome this limitation by extending Datalog with existential quantification in rule-heads; see, e.g., [6, 12–14, 26, 27]. More precisely, existential rules are implications among conjunctions of atoms, and they essentially say that some tuples in a relational instance I imply the presence of some other tuples in I (hence the name tuple-generating dependencies). Unfortunately, the addition of existential quantifiers immediately leads to undecidability of conjunctive query answering [10, 12], which is the main reasoning service under existential rules. *Conjunctive queries (CQs)*, which form one of the most commonly used language for querying relational databases, are assertions of the form $\exists \mathbf{Y}\, \varphi(\mathbf{X}, \mathbf{Y})$, where φ is a conjunction of atoms, and correspond to the select-project-join fragment of relational algebra [1]. The answer to a CQ w.r.t. a database D and a set Σ of existential rules consists of all the tuples $\mathbf{t}$ of constants such that $\exists \mathbf{Y} \varphi(\mathbf{t}, \mathbf{Y})$ evaluates to *true* in every model of $(D \wedge \Sigma)$.

Several concrete languages which ensure the decidability of CQ answering have been proposed over the last five years; see, e.g., [6, 12, 14, 18, 23, 25–27]. Nevertheless, existential rules are not expressive enough for nondeterministic reasoning; for example, the statement "each parent of a father is the grandparent of a boy *or* a girl" is not expressible via existential rules. Such a statement can be expressed using the rules

$$\forall X \forall Y\ \mathit{parentOf}(X, Y) \wedge \mathit{isfather}(Y) \ \rightarrow\ \exists Z\ \mathit{grandparentOf}(X, Z)$$
$$\forall X \forall Y\ \mathit{grandparentOf}(X, Y) \ \rightarrow\ \mathit{boy}(Y) \vee \mathit{girl}(Y).$$

Obviously, to represent such kind of disjunctive knowledge, we need to extend the existing classes of existential rules with disjunction in the head of rules. Enriching existential rules with disjunction yields the formalism of *disjunctive existential rules*, a.k.a. *disjunctive TGDs (DTGDs)* [17]; henceforth, for brevity, we adopt the terms (D)TGDs.

Guarded-based DTGDs. Guardedness is a well-known restriction which guarantees good model-theoretic and computational properties for first-order sentences [3]. Recently, inspired by guardedness, the class of guarded TGDs, that is, rules with a guard atom in the left-hand side which contains (or guards) all the universally quantified variables, has been defined [12]. Several extensions and restrictions of guarded TGDs have been proposed [6, 13]; we refer to all those formalisms by the term guarded-based TGDs, and more details will be given in Section 2. Guarded-based TGDs can be naturally extended to DTGDs. For example, the above set of rules is guarded since the atoms $\mathit{parentOf}(X, Y)$ and $\mathit{grandparentOf}(X, Y)$ are guards.

The complexity picture for query answering under the main guarded-based classes of DTGDs has been recently completed for arbitrary CQs [11]. Moreover, the complexity of answering atomic CQs, i.e., CQs consisting of a single atom, has been also investigated [2, 22]. However, the complexity picture of the problem restricted on some important subclasses of CQs, namely queries of bounded hypertree-width [20], queries of bounded treewidth [16], and acyclic queries [19], is still foggy, and there are several challenging open questions.

Research Challenges. The above subclasses of CQs have been proposed with the aim of reducing the complexity of several key decision problems on CQs such as evaluation of (Boolean) queries and query containment; in fact, those problems are NP-complete in general, but become tractable if restricted to one of the above subclasses [16, 19, 20]. The main objective of this work is to understand whether the subclasses of CQs in question have an analogous positive impact on query answering under the main guarded-based classes of DTGDs.

Although we know that our problem is unlikely to become tractable (implicit in [15]), we would like to understand whether its complexity is reduced. To achieve this, we focus on the following fundamental questions: (1) What is the exact complexity of answering queries which fall in one of the above subclasses of CQs under the main guarded-based classes of DTGDs?; (2) How is it affected if we consider predicates of bounded arity, or a fixed set of DTGDs, or a fixed set of DTGDs and a fixed query (a.k.a. the data complexity, where only the database is part of the input)?; and (3) How is it affected if we consider unions of CQs, i.e., disjunctions of a finite number of CQs? We provide answers to all these questions. This allows us to close the picture of the complexity of our problem, and come up with some general and insightful conclusions.

Our Findings. Our findings can be summarized as follows:

1. We show that (unions of) CQs of bounded hypertree-width and of bounded treewidth do not reduce the complexity of the problem under investigation. In particular, we show that for all the guarded-based classed of DTGDs in question, the problem remains 2ExpTime-complete, even if we focus on predicates of bounded arity, or on fixed sets of DTGDs, while the data complexity remains coNP-complete. The data complexity results are inherited from existing works. However, all the other results are obtained by establishing a remarkably strong lower bound, namely query answering under a *fixed* set of DTGDs expressed in a lightweight fragment of guarded DTGDs, that is, constant-free rules with just one atom in the left-hand side without repeated variables, is 2ExpTime-hard.
2. Regarding acyclic (unions of) CQs, we show that for all the classes of DTGDs under consideration, the problem remains 2ExpTime-complete in general, and coNP-complete in data complexity. Again, the data complexity is inherited from existing results, while the 2ExpTime-completeness is obtained by establishing a non-trivial lower bound. However, in some relevant cases the acyclicity of the query reduces the complexity of our problem to ExpTime-complete. In fact, this requires to focus on predicates of bounded arity, and for some expressive classes of DTGDs, on fixed sets of DTGDs. The upper bounds are obtained by exploiting results on the guarded fragment of first-order logic, while the lower bounds required a non-trivial proof.

To sum up, queries of bounded hypertree-width and of bounded treewidth, as well as acyclic queries, do not have the expected positive impact on query answering under the main guarded-based classes of DTGDs. However, a positive impact can be observed on some relevant settings of our problem if we consider acyclic queries.

2 Preliminaries

General. Let $\mathbf{C}$, $\mathbf{N}$ and $\mathbf{V}$ be pairwise disjoint infinite countable sets of *constants*, *(labeled) nulls* and *variables*, respectively. We denote by $\mathbf{X}$ sequences (or sets) of variables $X_1, \ldots, X_k$. Let $[n] = \{1, \ldots, n\}$, for $n \geqslant 1$. A *term* is a constant, null or variable. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. For an atom $\underline{a}$, $dom(\underline{a})$ and $var(\underline{a})$ are the set of its terms and the set of its variables, respectively; those notations extend to sets of atoms. Usually conjunctions and disjunctions of atoms are treated as sets of atoms. An *instance* I is a (possibly infinite) set of atoms of the form $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. A *database* D is a finite instance with only constants. Whenever an instance I is treated as a logical formula, is the formula $\exists \mathbf{X} \left(\bigwedge_{\underline{a} \in I} I\right)$, where $\mathbf{X}$ contains a variable for each null in I.

Conjunctive Queries. A *conjunctive query (CQ)* q is a sentence $\exists \mathbf{X} \varphi(\mathbf{X})$, where φ is a conjunction of atoms. If q does not have free variables, then it is called *Boolean*. For brevity, we consider only Boolean CQs; however, all the results of the paper can be easily extended to non-Boolean CQs. A *union of conjunctive queries (UCQ)* is a disjunction of a finite number of CQs. By abuse of notation, sometimes we consider a UCQ as set of CQs. A CQ $q = \exists \mathbf{X} \varphi(\mathbf{X})$ has a positive answer over an instance I, written $I \models q$, if there exists a homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$. The answer to a UCQ Q over I is positive, written $I \models Q$, if there exists $q \in Q$ such that $I \models q$. A key subclass of CQs is the class of CQs of *bounded treewidth* (BTWCQs) [16], i.e., the treewidth of their hypergraph is bounded. The hypergraph of a CQ q, denoted $\mathcal{H}(q)$, is a hypergraph $\langle V, H \rangle$, where $V = dom(q)$, and, for each $\underline{a} \in q$, there exists a hyperedge $h \in H$ such that $h = dom(\underline{a})$. The treewidth of q is defined as the treewidth of its hypergraph $\mathcal{H}(q)$, that is, the treewidth of the Gaifman graph $G_{\mathcal{H}(q)}$ of $\mathcal{H}(q)$. The Gaifman graph of $\mathcal{H}(q)$ is the graph $\langle V, E \rangle$, where V is the node set of $\mathcal{H}(q)$, and $(v, u) \in E$ iff $\mathcal{H}(q)$ has a hyperedge h such that $\{v, u\} \subseteq h$. Another important subclass of CQs is the class of *acyclic* CQs (ACQs) [16]. A CQ is acyclic if $\mathcal{H}(q)$ is acyclic, i.e., it can be reduced to the empty hypergraph by iteratively eliminating some non-maximal hyperedge, or some vertex contained in at most one hyperedge.

Disjunctive Tuple-Generating Dependencies. A *disjunctive TGD* (or simply *DTGD*) σ is a first-order formula $\forall \mathbf{X} \left(\varphi(\mathbf{X}) \rightarrow \bigvee_{i=1}^{n} \exists \mathbf{Y}_i \ \psi_i(\mathbf{X}, \mathbf{Y}_i)\right)$, where $n \geqslant 1$, $\mathbf{X} \cup \mathbf{Y} \subset \mathbf{V}$, and $\varphi, \psi_1, \ldots, \psi_n$ are conjunctions of atoms. The formula φ is called the *body* of σ, denoted $body(\sigma)$, while $\bigvee_{i=1}^{n} \psi_i$ is the *head* of σ, denoted $head(\sigma)$. The set of variables $var(body(\sigma)) \cap var(head(\sigma)) \subseteq \mathbf{X}$ is known as the *frontier* of σ, denoted $frontier(\sigma)$. If $n = 1$, then σ is called *tuple-generating dependency (TGD)*. The *schema* of a set Σ of DTGDs, denoted $sch(\Sigma)$, is the set of all predicates occurring in Σ. For brevity, we will omit the universal quantifiers, and use the comma (instead of $\wedge$). An instance I satisfies σ, written $I \models \sigma$, if whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$, then there exists $i \in [n]$ and $h' \supseteq h$ such that $h'(\psi_i(\mathbf{X}, \mathbf{Y}_i)) \subseteq I$; I satisfies a set Σ of DTGDs, denoted $I \models \Sigma$, if $I \models \sigma$, for each $\sigma \in \Sigma$. A *disjunctive inclusion dependency (DID)* is a constant-free DTGD with only one body-atom, the head is a disjunction of atoms, and there are no repeated variables in the body

or in the head. A DTGD σ is *linear* if it has only one body-atom. A DTGD σ is *guarded* if there exists $\underline{a} \in \mathit{body}(\sigma)$, called *guard*, which contains all the variables in $\mathit{body}(\sigma)$. *Weakly-guarded* DTGDs extend guarded DTGDs by requiring only the body-variables that appear at *affected positions*, i.e., positions at which a null value may appear during the disjunctive chase (defined below), to appear in the guard; see [12]. A DTGD σ is *frontier-guarded* if there exists $\underline{a} \in \mathit{body}(\sigma)$ which contains all the variables of $\mathit{frontier}(\sigma)$. *Weakly-frontier-guarded* DTGDs are defined analogously.

Query Answering. The *models* of a database D and a set Σ of DTGDs, denoted $\mathit{mods}(D, \Sigma)$, is the set of instances $\{I \mid I \supseteq D \text{ and } I \models \Sigma\}$. The *answer* to a CQ q w.r.t. D and Σ is *positive*, denoted $D \cup \Sigma \models q$, if $I \models q$, for each $I \in \mathit{mods}(D, \Sigma)$. The answer to a UCQ w.r.t. D and Σ is defined analogously. Our problem is defined as follows: Given a CQ q, a database D, and a set Σ of DTGDs, decide whether $D \cup \Sigma \models q$. If q is a BTWCQ (resp., ACQ), then the above problem is called *BTWCQ* (resp., *ACQ*) *answering*. The problem BTWUCQ (resp., AUCQ) answering is defined analogously. The *data complexity* is calculated taking only the database as input. For the *combined complexity*, the query and set of DTGDs count as part of the input as well.

Disjunctive Chase. Consider an instance I, and a DTGD $\sigma : \varphi(\mathbf{X}) \rightarrow \bigvee_{i=1}^{n} \exists \mathbf{Y}\, \psi_i(\mathbf{X}, \mathbf{Y})$. We say that σ is *applicable* to I if there exists a homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$, and the result of applying σ to I with h is the set $\{I_1, \ldots, I_n\}$, where $I_i = I \cup h'(\psi_i(\mathbf{X}, \mathbf{Y}))$, for each $i \in [n]$, and $h' \supseteq h$ is such that $h'(Y)$ is a "fresh" null not occurring in I, for each $Y \in \mathbf{Y}$. For such an application of a DTGD, which defines a single DTGD *chase step*, we write $I\langle\sigma, h\rangle\{I_1, \ldots, I_n\}$. A *disjunctive chase tree* of a database D and a set Σ of DTGDs is a (possibly infinite) tree such that the root is D, and for every node I, assuming that $\{I_1, \ldots, I_n\}$ are the children of I, there exists $\sigma \in \Sigma$ and a homomorphism h such that $I\langle\sigma, h\rangle\{I_1, \ldots, I_n\}$. The disjunctive chase algorithm for D and Σ consists of an exhaustive application of DTGD chase steps in a fair fashion, which leads to a disjunctive chase tree T of D and Σ; we denote by $\mathit{chase}(D, \Sigma)$ the set $\{I \mid I \text{ is a leaf of } T\}$. It is well-known that, given a UCQ Q, $D \cup \Sigma \models Q$ iff $I \models Q$, for each $I \in \mathit{chase}(D, \Sigma)$.

The Guarded Fragment of First-Order Logic. The *guarded fragment (GFO)* has been introduced in [3]. The set of GFO formulas over a schema $\mathcal{R}$ is the smallest set (1) containing all atomic $\mathcal{R}$-formulas and equalities; (2) closed under the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$; and (3) if $\underline{a}$ is an $\mathcal{R}$-atom containing all the variables of $\mathbf{X} \cup \mathbf{Y}$, and φ is a GFO formula with free variables contained in $(\mathbf{X} \cup \mathbf{Y})$, then $\forall \mathbf{X}(\underline{a} \rightarrow \varphi)$ and $\exists \mathbf{X}(\underline{a} \wedge \varphi)$ are GFO formulas. The *loosely guarded fragment (LGFO)* is a generalization of GFO where the quantifiers are guarded by conjunctions of atomic formulas; for details see, e.g., [24].

Alternation. An *alternating Turing machine* is a tuple $M = (S, \Lambda, \delta, s_0)$, where $S = S_\forall \uplus S_\exists \uplus \{s_a\} \uplus \{s_r\}$ is a finite set of states partitioned into universal states, existential states, an accepting state and a rejecting state, Λ is the tape alphabet, $\delta \subseteq (S \times \Lambda) \times (S \times \Lambda \times \{-1, 0, +1\})$ is the transition relation, and $s_0 \in S$ is the initial state. We assume that Λ contains a special blank symbol $\sqcup$.

	Combined Complexity	Bounded Arity	Fixed Rules	Data Complexity
DID	2ExpTime	2ExpTime	2ExpTime LB: Thm. 1	coNP LB: [15, Thm. 4.5]
L/G	2ExpTime	2ExpTime	2ExpTime	coNP
F-G	2ExpTime	2ExpTime	2ExpTime	coNP UB: [11, Thm. 7]
W-G	2ExpTime	2ExpTime	2ExpTime	ExpTime LB: [12, Thm. 4.1]
W-F-G	2ExpTime UB: [11, Thm. 1]	2ExpTime	2ExpTime	ExpTime UB: [11, Thm. 7]

Table 1. Complexity of BTW(U)CQ answering. Each row corresponds to a class of DTGDs; substitute L for linear, G for guarded, F for frontier, and W for weakly. UB and LB stand for upper and lower bound. The missing references for the upper (lower) bounds are inherited from the first lower-left (upper-right) cell with a reference.

3 Bounded Treewidth Queries

In this section, we focus on answering (U)CQs of bounded treewidth under our respective classes of DTGDs. Table 1 gives the complete picture of the complexity of our problem. As you can observe, the data complexity for all the classes of DTGDs under consideration is obtained from existing results. More precisely, the coNP-hardness for DIDs is obtained from [15, Theorem 4.5], where it is shown that CQ answering under a single DID of the form $p_1(X) \rightarrow p_2(X) \vee p_3(X)$, is already coNP-hard, even if the input query is fixed (and thus of bounded treewidth). The coNP upper bound for frontier-guarded DTGDs has been established in [11, Theorem 7] by a reduction to UCQ answering under GFO sentences. The ExpTime-hardness for weakly-guarded DTGDs is inherited from [12, Theorem 4.1], where it is shown that CQ answering under a fixed set of weakly-guarded TGDs is ExpTime-hard, even if the input query is a single atom. The ExpTime upper bound for weakly-frontier-guarded DTGDs has be shown in [11, Theorem 7] again by a reduction to UCQ answering under GFO.

Although the data complexity of our problem can be settled by exploiting known results, the picture for all the other cases is still foggy. The best known upper bound is the 2ExpTime upper bound for answering arbitrary UCQs under weakly-frontier-guarded DTGDs [11, Theorem 1], established by a reduction to the satisfiability problem of the guarded negation fragment [9], an extension of GFO. This result, combined with the fact that CQ answering under guarded TGDs is 2ExpTime-hard in the combined complexity [12, Theorem 6.1], even for atomic queries of the form $\exists X p(X)$ (and thus of bounded treewidth), closes the combined complexity for (weakly-)(frontier-)guarded DTGDs. However, the above lower bound for guarded DTGDs is not strong enough to complete the complexity picture of our problem. We establish a strong lower bound which, together with the above 2ExpTime upper bound, gives us the complete picture of the complexity of the problem studied in this section.

Theorem 1. *BTWCQ answering under fixed sets of DIDs is 2*ExpTime*-hard.*

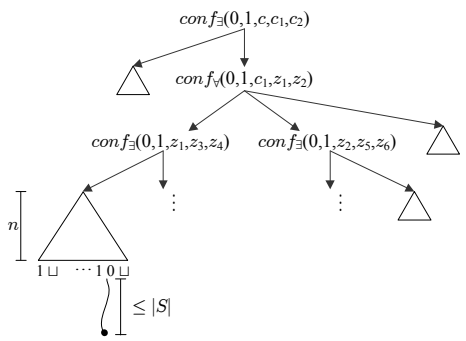

Fig. 1. Representation of the computation tree of M in the proof of Theorem 1.

Proof (sketch). The proof is by a reduction from the non-acceptance problem of an alternating exponential space Turing machine $M = (S, \Lambda, \delta, s_0)$ on the empty input. We assume that M uses exactly 2^n tape cells, $\Lambda = \{0, 1, \sqcup\}$, the initial configuration is existential, and every universal configuration is followed by two existential configurations and vice versa. Our goal is to construct a database D, a fixed set Σ of DIDs, and a BTWCQ q such that $D \cup \Sigma \models q$ iff M rejects. The general idea is to construct, by chasing D and Σ, all the trees which may encode a possible computation tree of M; in other words, each instance $I \in \mathit{chase}(D, \Sigma)$ will encode such a tree T_I. More precisely, the initial configuration is stored in the database D as the atom $\mathit{conf}_\exists(0, 1, c, c_1, c_2)$, where $\{c, c_1, c_2\} \subset \mathbf{C}$ are constants which represent the initial configuration (c), and its two subsequent configurations (c_1 and c_2) and 0 and 1 are auxiliary constants that will allow us to have access to 0 and 1 without explicitly mention them in Σ. Then, starting from the initial configuration, we construct a tree whose nodes are configurations, i.e., atoms of the form $\mathit{conf}_x(0, 1, t, n_1, n_2)$, where $x \in \{\exists, \forall\}$. Moreover, on each configuration node v, which represents the configuration C_v of M, we attach a configuration tree, that is, a full binary tree of depth n, and thus at its n-th level there are exactly 2^n nodes which represent the cells of the tape of M in C_v. Furthermore, for each cell we guess its content (0, 1 or $\sqcup$), and also whether the cursor of M is at this cell, and if so, we attach a chain of length at most $|S|$, which encodes the state of C_v. The above informal description is illustrated in Figure 1. Finally, we construct a BTWCQ q such that, if $I \in \mathit{chase}(D, \Sigma)$ entails q, then T_I is *not* a valid computation tree of M. □

The above strong lower bound closes all the missing cases regarding the complexity of our problem, and the next result follows:

Corollary 1. *BTW(U)CQ answering under (weakly-)(frontier-)guarded DTGDs, linear DTGDs and DIDs is 2*ExpTime*-complete in the combined complexity. The same holds for predicates of bounded arity, and for fixed sets of DTGDs.*

Another key class of queries is the class of CQs of *bounded hypertree-width* [20]. The hypertree-width is a measure of how close to acyclic a hypergraph is, analo-

	Combined Complexity	Bounded Arity	Fixed Rules	Data Complexity
DID	2ExpTime LB: Thm. 4	ExpTime	ExpTime LB: Thm. 6	coNP LB: [15, Thm. 4.5]
L/G	2ExpTime	ExpTime	ExpTime	coNP
F-G	2ExpTime	2ExpTime UB: [11, Thm. 1] LB: Thm. 5	ExpTime	coNP UB: [11, Thm. 7]
W-G	2ExpTime	ExpTime UB: Thm. 2	ExpTime	ExpTime LB: [12, Thm. 4.1]
W-F-G	2ExpTime UB: [11, Thm. 1]	2ExpTime LB: Thm. 5	ExpTime UB: Thm. 3	ExpTime UB: [11, Thm. 7]

Table 2. Complexity of answering acyclic (U)CQs.

gous to treewidth for graphs. The hypertree-width of a CQ is less than or equal to its treewidth. Since all the upper bounds in Table 1 hold for arbitrary (U)CQs, we get that arbitrary (U)CQs, (U)CQs of bounded treewidth and (U)CQs of bounded hypertree-width are indistinguishable w.r.t. to the complexity of query answering under our DTGDs.

4 Acyclic Queries

In this section, we focus on answering (unions of) acyclic queries under our respective classes of DTGDs. Table 2 gives the complete picture of the complexity of our problem. Compared with Table 1, it is immediately apparent that we can inherit from existing works the same results as for BTW(U)CQ answering, namely the data complexity in all the cases, and the 2ExpTime upper bound for answering arbitrary UCQs under weakly-frontier-guarded DTGDs. Therefore, apart from the data complexity, several non-trivial cases are still missing. As you can observe in Table 2, the combined and the data complexity do not change if we restrict our selves to acyclic queries. However, the complexity decreases from 2ExpTime to ExpTime for the non-frontier classes of DTGDs, i.e., DIDs, linear and (weakly-)guarded DTGDs, in the case of predicates of bounded arity, and also for all the classes if we consider a fixed set of DTGDs. This is an interesting finding as, in general, queries of bounded treewidth and acyclic queries behave in the same way. Let us now proceed with our results.

4.1 Upper Bounds

We start this section by showing that answering acyclic queries under weakly-guarded sets of DTGDs is in ExpTime in case of bounded arity. This is shown by a reduction to satisfiability of LGFO. Let us briefly explain the reduction via a simple example. Consider the database $D = \{s(a, a), p(a, b, c)\}$ and the set Σ consisting of

$$\sigma_1 : t(X, Y), p(Z, W, X), s(Z, V) \rightarrow \exists U\, p(Y, Z, U) \vee s(Y, Z),$$
$$\sigma_2 : p(X, Y, Z) \rightarrow \exists W\, t(Z, W).$$

Observe that the affected positions of $sch(\Sigma)$, i.e., the positions where nulls may appear during the construction of $chase(D, \Sigma)$, are $p[3]$, $t[1]$ and $t[2]$. Clearly, $t(X, Y)$ is the weak-guard for σ_1 and $p(X, Y, Z)$ the weak-guard for σ_2. Notice that σ_2 is already a GFO (and thus an LGFO) sentence. Thus, we need to convert σ_1 into an LGFO sentence. This can be done by expanding the weak-guard $t(X, Y)$ into a conjunction of atoms to guard the variables Z, V and W, and obtain the sentence Ψ_{σ_1}

$$\forall X \forall Y \forall Z \forall V \forall W \left(\left(\hat{t}(Z, V, X, Y) \wedge \hat{t}(Z, W, X, Y) \wedge \hat{t}(V, W, X, Y) \right) \rightarrow \right. \\ \left. (p(Z, W, X) \wedge s(Z, V)) \rightarrow \exists U \left(p(Y, Z, U) \vee s(Y, Z) \right) \right).$$

It should not be forgotten to properly generate the atoms with the auxiliary predicate $\hat{t}$. Since the variables Z, V and W can be satisfied only with constants of $dom(D) = \{a, b, c\}$, those atoms can be generated via the GFO sentence $\Psi_{\hat{t}}$

$$\forall X \forall Y \left(t(X, Y) \rightarrow \hat{t}(a, a, X, Y) \wedge \hat{t}(a, b, X, Y) \wedge \hat{t}(a, c, X, Y) \wedge \right. \\ \hat{t}(b, a, X, Y) \wedge \hat{t}(b, b, X, Y) \wedge \hat{t}(b, c, X, Y) \wedge \\ \left. \hat{t}(c, a, X, Y) \wedge \hat{t}(c, b, X, Y) \wedge \hat{t}(c, c, X, Y) \right).$$

It is not difficult to see that, for every acyclic UCQ Q, $D \cup \Sigma \models Q$ iff the sentence $\Psi = (D \wedge \Psi_{\sigma_1} \wedge \Psi_{\hat{t}} \wedge \sigma_2 \wedge \neg Q)$ is unsatisfiable. Notice that Ψ does not immediately fall into LGFO because of the query Q. However, since Q is acyclic, there exists an equivalent UCQ Q' which falls in GFO [21]. Thus, $\Psi' = (D \wedge \Psi_{\sigma_1} \wedge \Psi_{\hat{t}} \wedge \sigma_2 \wedge \neg Q')$ falls in LGFO and is equivalent to Ψ. Let us clarify that, if the head of σ_1 is a disjunction of conjunctions (instead of atoms as in the above example), then Ψ' is "almost" loosely-guarded since the existentially quantified variables are not necessarily guarded. However, as explicitly remarked in [24], the satisfiability algorithm for LGFO sentences is general enough to also treat sentences which are "almost" LGFO without increasing the complexity. From the above informal discussion we get that:

Theorem 2. *AUCQ answering under weakly-guarded sets of DTGDs is in* ExpTime *in case of predicates of bounded arity.*

The above machinery cannot be applied in the case of weakly-frontier-guarded DTGDs since the variables that we need to guard may appear at affected positions. However, if we focus on fixed sets of DTGDs, then the complexity is reduced to ExpTime. This is established by first reducing our problem to UCQ answering under GFO sentences, and then exploit a result in [8], where the problem of querying GFO is studied.

Theorem 3. *AUCQ answering under fixed weakly-frontier-guarded sets of DTGDs is in* ExpTime.

We believe that the results of this section can have a practical impact on other important tasks, such as querying graph databases [5], or querying description logic ontologies [4], where the attention is usually focussed on unary and binary predicates.

4.2 Lower Bounds

We start this section by showing the following non-trivial lower bound:

Theorem 4. *ACQ answering under DIDs is 2*ExpTime*-hard in combined complexity.*

Proof (sketch). We follow the same approach as in the proof of Theorem 1. However, the way that a computation tree of the alternating Turing machine $M = (S, \Lambda, \delta, s_0)$ is represented in that proof is not useful since it will necessarily lead to a cyclic query Q. This is exactly the non-trivial part of the proof, i.e., to construct, by chasing D and Σ, all the possible trees which may encode a computation tree of M in such a way that an acyclic query Q can be employed. To this aim, the configurations of M are represented using atoms of the form $\mathit{conf}[s](b_1, \ldots, b_n, a, h, t, p, n_1, n_2)$, where $s \in S$ is the state of the encoded configuration (and is part of the predicate), $(b_1, \ldots, b_n) \in \{0, 1\}^n$ is an integer of $\{0, \ldots, 2^n - 1\}$ in binary encoding which represents the index of the encoded cell, $h \in \{0, 1\}$ and $h = 1$ means that the cursor of M is at the encoded cell, and t, p, n_1 and n_2 represent the current, the previous and the next two configurations, respectively. More precisely, using a fixed number of DIDs, one can construct a tree with nodes of the form $\mathit{conf}_0[s](0, 1, \sqcup, 0^{2n}, 1^n, 1, z_1, z_2, z_3, z_4)$; such an atom is associated with the configuration z_1, and contains all the auxiliary constants that will allow us to generate, via polynomially many DIDs, all the 2^n atoms of the form $\mathit{conf}[s](b_1, \ldots, b_n, a, h, z_1, z_2, z_3, z_4)$. □

The above lower bound and the 2ExpTime upper bound for weakly-frontier-guarded sets of DTGDs in combined complexity [11, Theorem 1] imply that:

Corollary 2. *A(U)CQ answering under (weakly-)(frontier-)guarded DTGDs, linear DTGDs and DIDs is 2*ExpTime*-complete in combined complexity.*

Let us now focus on frontier-guarded DTGDs, and show that query answering is 2ExpTime-hard, even for ACQs and predicates of bounded arity. This is shown by exploiting the fact that a CQ $\exists \mathbf{X}\, \varphi(\mathbf{X})$ is actually the frontier-guarded TGD $\varphi(\mathbf{X}) \rightarrow p$. We thus have a reduction from CQ answering under frontier-guarded DTGDs, which is 2ExpTime-hard even for predicates of bounded arity [7].

Theorem 5. *ACQ answering under frontier-guarded DTGDs is 2*ExpTime*-hard, even for predicates of bounded arity.*

The above result and the 2ExpTime upper bound for weakly-frontier-guarded sets of DTGDs in combined complexity [11, Theorem 1] imply that:

Corollary 3. *A(U)CQ answering under (weakly-)frontier-guarded DTGDs is 2*ExpTime*-complete in case of predicates of bounded arity.*

We conclude this section by establishing the ExpTime-hardness of ACQ answering when we focus on fixed sets of DIDs. This is done by simulating an alternating linear space Turing machine M. The idea of the proof is along the lines of the proofs of Theorems 1 and 4. In particular, on each configuration node v, which represents the configuration C_v of M, we attach a cell-chain which mimics the tape in C_v, and a state-chain which encodes the state of C_v.

Theorem 6. *ACQ answering under fixed sets of DIDs is* ExpTime*-hard.*

From Theorems 2, 3 and 6 we get the that:

Corollary 4. *A(U)CQ answering under (weakly-)guarded DTGDs, linear DTGDs and DIDs is* ExpTime*-complete for predicates of bounded arity. The same problem under (weakly-)(frontier-)guarded DTGDs, linear DTGDs and DIDs is* ExpTime*-complete for fixed sets of DTGDs.*

5 Conclusions

We studied the problem of answering (U)CQs under the main guarded-based classes of DTGDs. We focussed on three key subclasses of (U)CQs, namely (U)CQs of bounded hypertree-width, (U)CQs of bounded treewidth, and acyclic (U)CQs. Our investigation shows that the above query languages do not have the expected positive impact on our problem, and in most of the cases the complexity of the problem remains 2ExpTime-complete. However, in some relevant settings, the complexity reduces to ExpTime-complete if we focus on acyclic queries. We believe that this finding can have a practical impact on crucial tasks such as querying graph databases and querying description logic ontologies, where the attention is usually focussed on unary and binary predicates.

Acknowledgements. Pierre thankfully acknowledges projet équipe associée Inria North-European Labs 2013-2016, Michael his DOC Fellowship of the Austrian Academy of Sciences, and Andreas the EPSRC grant EP/J008346/1 (PrOQAW). We thank the anonymous referees for many helpful comments.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alviano, M., Faber, W., Leone, N., Manna, M.: Disjunctive Datalog with existential quantifiers: Semantics, decidability, and complexity issues. TPLP 12(4-5), 701–718 (2012)
3. Andréka, H., van Benthem, J., Németi, I.: Modal languages and bounded fragments of predicate logic. J. Philosophical Logic 27, 217–274 (1998)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
5. Baeza, P.B.: Querying graph databases. In: PODS. pp. 175–188 (2013)
6. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. 175(9-10), 1620–1654 (2011)

7. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: IJCAI. pp. 712–717 (2011)
8. Bárány, V., Gottlob, G., Otto, M.: Querying the guarded fragment. In: LICS. pp. 1–10 (2010)
9. Bárány, V., ten Cate, B., Segoufin, L.: Guarded negation. In: ICALP. pp. 356–367 (2011)
10. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: ICALP. pp. 73–85 (1981)
11. Bourhis, P., Morak, M., Pieris, A.: The impact of disjunction on query answering under guarded-based existential rules. In: IJCAI (2013)
12. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. 48, 115–174 (2013)
13. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
14. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artif. Intell. 193, 87–128 (2012)
15. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artif. Intell. 195, 335–360 (2013)
16. Chekuri, C., Rajaraman, A.: Conjunctive query containment revisited. Theor. Comput. Sci. 239(2), 211–229 (2000)
17. Deutsch, A., Tannen, V.: Reformulation of XML queries and constraints. In: ICDT. pp. 225–241 (2003)
18. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
19. Gottlob, G., Leone, N., Scarcello, F.: The complexity of acyclic conjunctive queries. J. ACM 48(3), 431–498 (2001)
20. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. J. Comput. Syst. Sci. 64(3), 579–627 (2002)
21. Gottlob, G., Leone, N., Scarcello, F.: Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. J. Comput. Syst. Sci. 66(4), 775–808 (2003)
22. Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: MFCS. pp. 1–18 (2012)
23. Gottlob, G., Manna, M., Pieris, A.: Combining decidability paradigms for existential rules. TPLP 13(4-5), 877–892 (2013)
24. Grädel, E.: On the restraining power of guards. J. Symb. Log. 64(4), 1719–1742 (1999)
25. Grau, B.C., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. J. Artif. Intell. Res. 47, 741–808 (2013)
26. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI. pp. 963–968 (2011)
27. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: KR (2012)
28. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the semantic web. J. Web Sem. 5(4), 240–250 (2007)