Acyclic Query Answering under Guarded Disjunctive Existential Rules and Consequences to DLs

Acyclic Query Answering Under Guarded Disjunctive Existential Rules and Consequences to DLs

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

[1] CNRS LIFL Université Lille1/INRIA Lille, France.
pierre.bourhis@univ-lille1.fr
[2] University of Oxford, Department of Computer Science, United Kingdom.
{michael.morak,andreas.pieris}@cs.ox.ac.uk

Abstract. The complete picture of the complexity of conjunctive query answering under guarded disjunctive existential rules has been recently settled. However, in the case of (unions of) acyclic conjunctive queries ((U)ACQs) there are some fundamental questions which are still open. It is the precise aim of the present paper to close those questions, and to understand whether the acyclicity of the query has a positive impact on the complexity of query answering. Our main result states that acyclic conjunctive query answering under a *fixed* set of guarded disjunctive existential rules is EXPTIME-hard. This result together with an EXPTIME upper bound obtained by exploiting classical results on guarded first-order logic, gives us a complete picture of the complexity of our problem. We also show that our results can be used as a generic tool for establishing results on (U)ACQ answering under several central DLs. In fact, restricting the query language to UACQs improves the complexity to EXPTIME-complete for any DL between *DL-Lite*$_{bool}$ and $\mathcal{ALCHI}$; this holds even for fixed TBoxes.

1 Introduction

Rule-based languages have a prominent presence in the areas of artificial intelligence and databases. A noticeable formalism, originally intended for expressing complex queries over relational databases, is Datalog, i.e., function-free first-order Horn logic. As Datalog itself is not able to infer the existence of new objects that do not occur in the extensional database, strong interest in enhancing Datalog with existential quantification in rule-heads emerged in recent years; see, e.g., [5, 6, 12, 23, 26]. The obtained rules are known under a variety of names such as *existential rules*, *tuple-generating dependencies (TGDs)*, and *Datalog*$^{\pm}$ *rules*. However, these languages are still not expressive enough for non-deterministic reasoning, e.g., to express simple statements like "a grandchild is a male *or* a female", which can be logically expressed using the rules

$$\forall X \forall Y (\mathit{parentOf}(X, Y) \wedge \mathit{isfather}(Y) \rightarrow \exists Z (\mathit{grparentOf}(X, Z) \wedge \mathit{human}(Z)))$$
$$\forall X (\mathit{human}(X) \rightarrow \mathit{male}(X) \vee \mathit{female}(X)).$$

Extending TGDs to allow such a disjunction in the head yields the formalism of *disjunctive TGDs (DTGDs)* [17]. Such an extension of plain Datalog (a.k.a. *full TGDs*), called *disjunctive Datalog*, has been studied in [18]. However, the addition of existential quantifiers easily leads to undecidability of the main reasoning tasks, including conjunctive query (CQ) answering [8].

Several concrete languages which ensure decidability have thus been proposed; see, e.g., [5, 10, 12, 13, 19, 23, 24]. In this paper, we will focus on a concept called "guardedness" [10], which requires that all universally quantified variables in a rule appear together in an atom in the left-hand side of the implication. Guardedness is a well-accepted paradigm, giving rise to robust languages that capture important description logics such as *DL-Lite* [14], $\mathcal{EL}$ [4] and even $\mathcal{ALCI}$. Recently, the complexity picture for query answering under guarded DTGDs has been investigated and completed for (unions of) arbitrary CQs [9]. Also for simple atomic queries the complexity has been investigated (see, e.g., [1, 21]). As acyclic CQs (ACQs) in the literature usually lead to a reduction in the complexity of the query answering problem, the objective of the present paper is to investigate whether this positive impact can also be observed in our setting. To this end, we aim to pinpoint the exact complexity of answering (unions of) acyclic CQs ((U)ACQs), in order to compare it to existing results for answering (unions of) arbitrary CQs. More precisely, we concentrate on the following fundamental question: What is the exact complexity of answering (U)ACQs under guarded DTGDs, and how is it affected if we consider a signature of bounded arity, or a fixed set of dependencies?

After establishing the complexity of (U)ACQ answering, we then investigate the consequences to DLs. We show that our results can be used as a generic tool for establishing results on (U)ACQ answering under several central DLs that can be found in the literature such as the key DL $\mathcal{ALCHI}$, that is, the DL $\mathcal{ALC}$ extended with role hierarchies and inverse roles. Our contributions can be summarized as follows:

1. We show that UACQ answering under guarded DTGDs with predicates of bounded arity is in EXPTIME. This upper bound is established by exploiting results from guarded first-order logic [22];
2. We show that ACQ answering under fixed sets of guarded DTGDs is EXPTIME-hard. This is a strong and general lower bound obtained by simulating the behavior of an alternating linear space Turing machine.
3. Finally, we show that our results for fixed arity and fixed sets extend to a number of expressive DLs, namely $\mathcal{ALCHI}$, $\mathcal{ELUHI}$, *DL-Lite*$^{\mathcal{H}}_{bool}$ and also *DL-Lite*$_{bool}$. For each of these DLs, and any others between them, answering UACQs, even over fixed TBoxes, is EXPTIME-complete. The upper bounds follow from the fact that guarded DTGDs are powerful enough to capture each of these DLs, and they thus inherit the positive effects of restricting the query language to acyclic queries. The lower bounds follow from the construction of our proof for guarded DTGDs.

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 $\exists \mathbf{X} \, (\bigwedge_{\underline{a} \in I} I)$, 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 $I \models q$.

Several subclasses of conjunctive queries have been considered in the literature, with the aim of reducing the complexity of the CQ evaluation problem. This paper focuses on the class of *acyclic* CQs (ACQs); see, e.g., [16]. The acyclicity of a CQ is defined via the acyclicity of its hypergraph. Informally, a hypergraph is acyclic if it can be reduced to the empty hypergraph by iteratively eliminating some non-maximal hyperedge, or some vertex contained in at most one hyperedge; this procedure is known as the GYO algorithm. The formal definition is as follows. Consider a hypergraph $\mathcal{H}$. The *GYO-reduct* of $\mathcal{H}$, denoted $GYO(\mathcal{H})$, is obtained by applying exhaustively the following two steps: (1) eliminate vertices that are contained in at most one hyperedge; and (2) eliminate hyperedges that are empty or contained in other hyperedges. We say that $\mathcal{H}$ is acyclic if $GYO(\mathcal{H}) = \langle \varnothing, \varnothing \rangle$. The hypergraph of a CQ q, denote $\mathcal{H}(q)$, is a hypergraph $\langle V, H \rangle$, where $V = dom(q)$, and, for each atom $\underline{a}$ in q, there exists a hyperedge $h \in H$ such that $h = dom(\underline{a})$. We say that q is *acyclic* iff $\mathcal{H}(q)$ is acyclic.

Disjunctive Tuple-Generating Dependencies. A *disjunctive tuple-generating dependency (DTGD)* σ is a first-order formula $\forall \mathbf{X} \, (\varphi(\mathbf{X}) \rightarrow \bigvee_{i=1}^{n} \exists \mathbf{Y}_i \, \psi_i(\mathbf{X}, \mathbf{Y}_i))$, where $n \geqslant 1$, $\mathbf{X} \cup \mathbf{Y}_1 \cup \ldots \cup \mathbf{Y}_n \subset \mathbf{V}$, 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 $head(\sigma)$. 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 in front of DTGDs, and use the comma (instead of $\wedge$) for conjoining atoms. An instance I satisfies σ, written $I \models \sigma$, if the following holds: 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 $\sigma \in \Sigma$. Whenever a set Σ of DTGDs is treated as a logical formula, in fact is the formula $(\bigwedge_{\sigma \in \Sigma} \sigma)$.

As said, (U)CQ answering under DTGDs (which is defined below) is undecidable; in fact, is undecidable already for TGDs [8], even when the set of TGDs is fixed and the query is acyclic [11], or even when the set of TGDs is singleton [5]. A well-known property which guarantees decidability of query answering is *guardedness* [11]. A DTGD σ is *guarded* if there exists an atom $\underline{a} \in body(\sigma)$, called *guard*, which contains all the variables occurring in $body(\sigma)$, i.e., $var(\underline{a}) = var(body(\sigma))$.

Query Answering. The *models* of a database D and a set Σ of DTGDs, denoted $mods(D, \Sigma)$, 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 $I \in mods(D, \Sigma)$.

The answer to a UCQ w.r.t. D and Σ is defined analogously. The problem tackled in this work 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 the input query is an ACQ, then the above problem is called *ACQ answering*. Analogously, the problem UACQ answering for unions of ACQs is defined. The *data complexity* of the above problems 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. We employ the *disjunctive chase* [17]. 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 $chase(D, \Sigma)$ 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 $I \in chase(D, \Sigma)$.

The Guarded Fragment of First-Order Logic. The *guarded fragment (GFO)* has been introduced in [2]. 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. It is known that the problem of deciding whether an GFO sentence is satisfiable is 2ExpTime-complete, and ExpTime-complete for predicates of bounded arity [22].

3 Answering (Unions of) Acyclic Queries

The problem of answering (unions of) arbitrary conjunctive queries has recently been investigated in [9]. It turns out that even for fixed sets of very simple forms of DTGDs, the problem is already 2ExpTime-hard. Restricting the query language to (unions of) acyclic queries has often proven to be a way to reduce the complexity. Our goal in this section is thus to investigate if this also holds true for guarded DTGDs, and to study how exactly the complexity of (U)CQ answering under our guarded DTGDs is affected if we focus on acyclic queries. We will begin this section by summarizing our results, and then proceed with our new complexity bounds.

Table 1 summarizes the complexity results for answering (U)ACQs under guarded DTGDs. Compared with the results for (U)CQs, as presented in [9], two observations can be made: Firstly, the combined and the data complexity do not change if we restrict ourselves to acyclic queries. Secondly, however, the complexity decreases from 2ExpTime-completeness to ExpTime-completeness, in the case of predicates of bounded arity, and also if we consider a fixed set of DTGDs. The impact of restricting

Combined Complexity	Bounded Arity	Fixed Rules	Data Complexity
2EXPTIME	EXPTIME	EXPTIME	coNP
UB: [9, Thm. 1]	UB: Thm. 1	UB: Thm. 1	UB: [7, Thm. 19]
LB: [11, Thm. 6.1]	LB: Thm. 2	LB: Thm. 2	LB: [15, Thm. 4.5]

Table 1: Complexity of answering (U)ACQs under guarded DTGDs.

the query language to acyclic queries is this only noticeable in these two cases. The novel results of this section follow:

1. UACQ answering under guarded DTGDs is in EXPTIME if we focus on predicates of bounded arity (Theorem 1) – this is shown by a reduction to satisfiability of a slight extension of GFO, and then exploit the fact that this problem is in EXPTIME in case of predicates of bounded arity [22]; and
2. ACQ answering under fixed sets of guarded DTGDs is EXPTIME-hard (Theorem 2) – by simulating the behavior of an alternating linear space Turing machine.

Existing results from the literature, provide us with the missing optimal upper and lower bounds, which complete the entire picture of the complexity of (U)ACQ answering. Firstly, the missing upper bounds are immediately inherited from results presented in [9], in particular the combined and data complexity of guarded DTGDs, which are in 2EXPTIME and coNP, respectively. Secondly, the coNP-hardness of ACQ answering under guarded DTGDs follows from [15, Theorem 4.5] – the CQ employed in the construction of the proof of this result is $\exists X \exists Y_1 \exists Y_2 \exists Z_1 \exists Z_2 (\bigwedge_{i \in \{1,2\}} (p_i(X, Y_i) \wedge s(Y_i) \wedge r_i(X, Z_i) \wedge t(Z_i)))$, which is clearly acyclic.

From the results of this section, we observe that the acyclicity of the query does not make our problem easier w.r.t. the combined and data complexity. However, in the cases of bounded arity and fixed sets of DTGDs, the complexity decreases from 2EXPTIME to EXPTIME. Let us now proceed with the formal proofs of our results.

3.1 Complexity Results

We start this section by showing that UACQ answering under guarded DTGDs is in EXPTIME in case of predicates of bounded arity. We can assume that each guarded DTGD σ is of the form $\alpha(\mathbf{X}) \wedge \phi(\mathbf{X}) \rightarrow \bigvee_{1 \leqslant i \leqslant m} \exists \mathbf{Y} \psi_i(\mathbf{X}, \mathbf{Y})$, where ϕ and ψ_i are conjunctions of atoms, and α is the atom that forms the guard of σ. Each such formula can now in fact be rewritten as the formula $\forall \mathbf{X}(\alpha(\mathbf{X}) \rightarrow (\phi(\mathbf{X}) \rightarrow \bigvee_{1 \leqslant i \leqslant m} \exists \mathbf{Y} \psi_i(\mathbf{X}, \mathbf{Y})))$. It can be verified that, if each ψ_i is a single atom, then the above formula falls into GFO. However, if ψ_i is a conjunction of atoms, then the formula is "almost" guarded since the existentially quantified variables are not necessarily guarded. Interestingly, the satisfiability algorithm proposed in [22] is able to treat formulas as the one above, even if the existentially quantified variables are not guarded, without increasing the complexity – this is explicitly stated in a remark on page 8 of [22]. By exploiting the above observation, it is now easy to reduce our problem to the satisfiability problem of sentences which are "almost" guarded, which is EXPTIME-complete in case of predicates of bounded arity [22]. Consider a database D, a set Σ of guarded DTGDs, and a UACQ

Q. The database D itself contains only ground atoms and is thus trivially guarded. The set Σ can be rewritten as a sentence Φ_Σ which is "almost" guarded, as explained above. Finally, the query Q, although is not guarded, it can be rewritten as a GFO sentence Φ_Q due to acyclicity [20]. Thus, $D \cup \Sigma \models Q$ iff the "almost" GFO sentence $(D \wedge \Phi_\Sigma \wedge \neg\Phi_Q)$ is not satisfiable, and we obtain the following:

Theorem 1. *UACQ answering under guarded DTGDs is in* EXPTIME *in case of predicates of bounded arity.*

We continue this section by showing that ACQ answering under fixed sets of guarded DTGDs is EXPTIME-hard. The construction simulates an alternating LINSPACE Turing machine. Using a fixed set of rules, chains of non-deterministic length can be generated, where the length represents a number. By linking these chains appropriately, configurations of the Turing machine can be guessed. An acyclic query is then used to check consistency w.r.t. the configuration and the transition function. The desired result follows since alternating LINSPACE already coincides with EXPTIME.

Theorem 2. *ACQ answering under fixed sets of guarded DTGDs is* EXPTIME*-hard.*

Proof. The proof is by a reduction from the non-acceptance of an alternating linear space Turing machine M on input I. Let $M = (S, \Lambda, \delta, s_0)$, 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, $\Lambda = \{0, 1, \sqcup\}$ is the tape alphabet with $\sqcup$ be the blank symbol, $\delta : S \times \Lambda \to (S \times \Lambda \times \{-1, 0, +1\})^2$ is the transition function, and $s_0 \in S$ is the initial state. We assume that M is well-behaved and never tries to read beyond its tape boundaries, always halts, and uses exactly n tape cells, where $n = O(|I|)$. Furthermore, we assume that a rejecting configuration does not have a subsequent configuration, while an accepting configuration has only itself as a subsequent configuration. Finally, we assume that $s_0 \in S_\exists$, and also that every universal configuration is followed by two existential configurations and vice versa. The above assumptions can be made, without sacrificing the generality of our proof, since the non-acceptance problem of M remains EXPTIME-hard. We are going to construct a database D, a set Σ of DTGDs, and a UACQ Q such that $D \cup \Sigma \models Q$ iff M rejects I, and Σ is a fixed set of guarded DTGDs. By [9, Lemma 4], the obtained lower bound holds even if we focus on ACQs. The idea of the proof is to construct trees, which encode possible computation trees of M on the input string I, by chasing D with Σ, and then exploit Q to check their consistency. On each configuration node v, which represents the configuration C_v of M, we attach a chain of length n, which mimics the tape in C_v, and a chain of length at most $|S|$, which encodes the state of C_v. The formal definition of D, Σ and Q follows.

The Database D**.** Let $D = \{conf_\exists(c)\}$, where $c \in \mathbf{C}$ is a special constant which represents the initial configuration (which is, by assumption, existential).

The Set Σ**.** The predicates that we are going to use are self-explanatory, and thus we proceed with the construction of Σ without describing the predicates of $sch(\Sigma)$.

– Each existential (universal) configuration has one (two) successor configuration(s) which is (are) universal (existential):

$$conf_\exists(X) \to \exists Y\, succ(X, Y), conf_\forall(Y),$$
$$conf_\forall(X) \to \exists Y \exists Z\, succ_1(X, Y), conf_\exists(Y), succ_2(X, Z), conf_\exists(Z).$$

– Each configuration has a cell- and state-chain which simulates its tape and encodes its state, respectively:

$$\begin{aligned} conf_x(X) &\rightarrow \exists Y\ cell(X,Y), \text{ for each } x \in \{\exists, \forall\}, \\ cell(X,Y) &\rightarrow \exists Z\ cell(Y,Z) \vee end(Y), \\ conf_x(X) &\rightarrow \exists Y\ state(X,Y), \text{ for each } x \in \{\exists, \forall\}, \\ state(X,Y) &\rightarrow \exists Z\ state(Y,Z) \vee end(Y). \end{aligned}$$

– Finally, we guess the content of each cell and the cursor position:

$$\begin{aligned} cell(X,Y) &\rightarrow 0(Y) \vee 1(Y) \vee \sqcup(Y), \\ cell(X,Y) &\rightarrow cursor(Y) \vee notCursor(Y). \end{aligned}$$

The construction of Σ is now completed. It is easy to see that Σ does not depend on M, and also that Σ is a set of guarded DTGDs.

The UACQ Q**.** We now proceed with the construction of Q which ensures that each model of $D \cup \Sigma$ encodes a valid computation tree. Roughly, Q consists of the following disjuncts:

1. $Q_{initial}$ for checking that in the initial configuration the tape contains the input string $I = a_1 \dots a_k$, the state is s_0, and the cursor is at the first cell;
2. Q_{length} for checking that cell-chains are of length n, and state-chains are of length at most $|S|$;
3. Q_{cursor} for checking that exactly one cell is pointed by the cursor;
4. Q_{trans} for checking the consistency w.r.t. the transition function of M; and
5. $Q_{inertia}$ for checking that the tape cells not under the cursor keep their old value during a transition.

We assume a fixed order on S. Given a state $s \in S$, let $\#_s$ be its position in this order. The length of the state-chain which encodes s is $\#_s$. For a predicate $p \in \{cell, state\}$, $p^i(X,Y)$ is used as a shorthand for a chain of length i along predicate p, namely $\exists Z_1 \dots \exists Z_{i-1}\,(p(X,Z_1) \wedge \dots \wedge p(Z_{i-1},Y))$. A useful subquery that we will use is

$$length[p]_k(X) \equiv \exists Y\,\left(p^k(X,Y) \wedge end(Y)\right)$$

stating that there exists a p-chain of length k. Having all the necessary ingredients in place, we are now ready to formalize the components of Q.

1. $Q_{initial}$ is defined as (recall that $c \in \mathbf{C}$ represents the initial configuration)

$$\begin{aligned} \bigvee_{i \in [n]} \bigvee_{a \in \Lambda \setminus \{a_i\}} & \exists X\,\left(conf_\exists(c) \wedge cell^i(c,X) \wedge a(X)\right) \vee \\ & \bigvee_{s \in S \setminus \{s_0\}} (conf_\exists(c) \wedge length[state]_{\#_s}(c)) \vee \\ & \bigvee_{1 < i \leqslant n} \exists X\,\left(conf_\exists(c) \wedge cell^i(c,X) \wedge head(X)\right). \end{aligned}$$

2. Q_{length} is defined as

$$\bigvee_{1 \leqslant i < n} \exists X \; length[cell]_i(X) \; \vee \; \exists X \exists Y \; cell^{n+1}(X, Y) \\ \vee \; \exists X \; length[state]_{|S|+1}(X).$$

3. Q_{cursor} is defined as

$$\bigvee_{1 \leqslant i < j \leqslant n} \exists X \exists Y \exists Z \left(cell^i(X, Y) \wedge cell^j(X, Z) \wedge cursor(Y) \wedge cursor(Z) \right).$$

4. In the definition of Q_{trans}, we use the function $f : S \to \{\exists, \forall\}$, where $f(s) = \exists$ if $s \in S_\exists$; otherwise, $f(s) = \forall$. Moreover, we use the binary operator $\odot^s$, where $s \in S$, which is defined as $\wedge$, if $s \in S_\exists$, and as $\vee$, if $s \in S_\forall$. Q_{trans} is defined as

$$\bigvee_{(s,a) \to ((s_1,a_1,d_1),(s_2,a_2,d_2)) \notin \delta} \; \overset{s}{\bigodot}_{i \in \{1,2\}} \; \bigvee_{j \in [n]} \exists X \exists Y \exists Z \exists V \exists W \\ \Big(conf_{f(s)}(X) \wedge length[state]_{\#_s}(X) \wedge cell^j(X, Y) \wedge cursor(Y) \wedge a(Y) \wedge \\ (succ(X, Z) \vee succ_1(X, Z) \vee succ_2(X, Z)) \wedge \\ length[state]_{\#_{s_i}}(Z) \wedge cell^j(Z, V) \wedge a_i(V) \wedge cell^{j+d_i}(Z, W) \wedge cursor(W) \Big).$$

5. Finally, $Q_{inertia}$ is defined as

$$\bigvee_{x \in \{\exists, \forall\}} \; \bigvee_{(a,a') \in \Lambda \times \Lambda, a \neq a'} \; \bigvee_{i \in [n]} \exists X \exists Y \exists Z_1 \exists Z_2 \, (conf_x(X) \wedge \\ (succ(X, Y) \vee succ_1(X, Y) \vee succ_2(X, Y)) \wedge \\ cell^i(X, Z_1) \wedge cell^i(Y, Z_2) \wedge notCursor(Z_1) \wedge a_1(Z_1) \wedge a_2(Z_2) \big).$$

This completes the construction of Q. It is easy to verify that each disjunct of Q is an acyclic query.

By construction, and the assumption that a rejecting configuration does not have a successor configuration, while an accepting configuration has only itself as a successor, it is not difficult to see that $D \cup \Sigma \models Q$ iff M rejects I. □

By Theorems 1 and 2, as well as inherited results described earlier, we can now state our main result that closes the complexity picture for guarded DTGDs w.r.t. answering (unions of) acyclic conjunctive queries.

Corollary 1. *(U)ACQ answering under guarded DTGDs is* 2EXPTIME*-complete in the combined complexity,* EXPTIME*-complete in case of fixed arity and fixed sets of DTGDs, and co*NP*-complete in the data complexity.*

It can be seen that, in comparison to arbitrary (unions of) conjunctive queries, the complexity in the general case remains the same. However, there is a notable complexity decrease from 2EXPTIME to EXPTIME in case where at least the arity is fixed. However, the data complexity remains coNP-complete.

4 Consequences to DLs

In this section, we will investigate the relationship between our previously discussed formalism of guarded DTGDs with DLs. Our aim is to show that our techniques and results can be used as a generic tool for establishing results on UACQ answering under several central DLs that can be found in the literature. To this end, we focus on the key DL $\mathcal{ALCHI}$, that is, the well known DL $\mathcal{ALC}$ extended with role hierarchies and inverse roles. The set of $\mathcal{ALCHI}$ concepts is the smallest set such that (1) any atomic concept, as well as $\bot$ and $\top$ are concepts; (2) for any role R, if C and D are concepts, then so are $C \sqcap D$, $\neg C$, $\forall R.C$, $\forall R^-.C$, $\exists R.C$ and $\exists R^-.C$. A TBox consists of a finite set of concept inclusions of the form $C \sqsubseteq D$, where C and D are concepts, as well as a finite set of role inclusions $R \sqsubseteq S$, where R and S are (possibly inverse) roles. An ABox consists of a finite set of assertions of the form $C(a)$ or $R(a, b)$, where a and b are individuals and C and R is a concept and role, respectively. A TBox $\mathcal{T}$ together with an ABox $\mathcal{A}$ consists a *knowledge base (KB)* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Its semantics are defined as usual via first-order interpretations.

4.1 A Generic Complexity Result

By exploiting our techniques and results, we can establish the following result about UACQ answering; $\mathcal{L}_1 \subseteq \mathcal{L}_2$ means that $\mathcal{L}_1$ is a subformalism of $\mathcal{L}_2$:

Theorem 3. *Consider a DL $\mathcal{L}$ such that $\mathcal{L} \subseteq \mathcal{ALCHI}$ and $\mathcal{L}$ is powerful enough for expressing the following inclusion assertions:*

$$A_1 \sqsubseteq A_2 \sqcup A_3 \qquad A \sqsubseteq \exists R.\top \qquad \exists R^-.\top \sqsubseteq A,$$

where A, A_1, A_2, A_3 are concept names. Then, UACQ answering under $\mathcal{L}$ is EXPTIME*-complete in combined complexity, even when the TBox is fixed.*

As we shall see, the above generic result closes some interesting open problems in a uniform way. The rest of this section is devoted to establish Theorem 3.

Upper Bound. The upper bound is obtained by reducing UACQ answering under $\mathcal{ALCHI}$ to UACQ answering under guarded DTGDs. First, we discuss how complex concepts can be eliminated from the given ABox by pushing them in the TBox. Such a reformulation of the given KB will allow us to consider the ABox as a relational database. Consider an $\mathcal{ALCHI}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We construct an ABox $\mathcal{A}'$ from $\mathcal{A}$ by replacing each assertion $C(a)$, where C is a complex concept, by $C^\star(a)$, where $C^\star$ is an auxiliary concept name not occurring in $\mathcal{A}$. Then, the TBox $\mathcal{T}'$ is constructed by adding to $\mathcal{T}$ an assertion of the form $C^\star \sqsubseteq C$, for each complex concept C occurring in $\mathcal{A}$. It is easy to see that $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{T}', \mathcal{A}')$ are equivalent w.r.t. UACQ answering. Let us now explain how an $\mathcal{ALCHI}$ TBox can be normalized in such a way that can be seen as a set of guarded DTGDs. In fact, we exploit the normalization procedure proposed in [25] for $\mathcal{ALCH}$, i.e., $\mathcal{ALCHI}$ without inverse roles. An $\mathcal{ALCHI}$ TBox is in normal form if it only contains axioms of the form given on the left column of Table 2. It is implicit in [25] that every $\mathcal{ALCH}$ TBox $\mathcal{T}$ can be transformed in polynomial time into an $\mathcal{ALCH}$ TBox in normal form which is equivalent to $\mathcal{T}$ w.r.t. UACQ answering. This

$\mathcal{ALCHI}$ Axiom	First-Order Representation
$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \bot$	$\forall X(A_1(X) \wedge \ldots \wedge A_n(X) \rightarrow \bot)$
$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B_1 \sqcup \ldots \sqcup B_m$	$\forall X(A_1(X) \wedge \ldots \wedge A_n(X) \rightarrow B_1(X) \vee \ldots \vee B_m(X))$
$A \sqsubseteq \exists R.B$	$\forall X(A(X) \rightarrow \exists Y R(X,Y) \wedge B(Y))$
$\exists R.A \sqsubseteq B$	$\forall X(R(X,Y) \wedge A(Y) \rightarrow B(X))$
$A \sqsubseteq \forall R.B$	$\forall X(A(X) \wedge R(X,Y) \rightarrow B(Y))$
$R \sqsubseteq S$	$\forall X \forall Y(R(X,Y) \rightarrow S(X,Y))$
$R \sqsubseteq S^-$	$\forall X \forall Y(R(X,Y) \rightarrow S(Y,X))$

Table 2: Normal form and first-order representation.

result can be straightforwardly extended to $\mathcal{ALCHI}$. From the above discussion, we immediately get the following auxiliary result:

Lemma 1. *Consider an $\mathcal{ALCHI}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We can construct in polynomial time an $\mathcal{ALCHI}$ KB $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$, where $\mathcal{T}'$ is in normal form and $\mathcal{A}'$ contains only concept and role names, such that $\mathcal{K} \models Q$ iff $\mathcal{K}' \models Q$, for every UACQ Q.*

As shown in Table 2, given an $\mathcal{ALCHI}$ TBox $\mathcal{T}$ in normal form, every axiom of $\mathcal{T}$ can be equivalently rewritten as a first-order sentence which is either a guarded DTGD or a constraint of the form $\forall X(A_1(X) \wedge \ldots \wedge A_n(X) \rightarrow \bot)$, where $\bot$ denotes the truth constant *false*, which may lead to an inconsistency. Such inconsistencies can be encoded in the query by considering the left-hand side of the constraints as disjuncts of the given UACQ. Moreover, observe that those disjuncts will be ACQs, and thus the obtained query remains acyclic. Now, an ABox which contains only concept and role names (and not complex concepts) can be naturally seen as a relational database. Therefore, from Lemma 1 and the above discussion, we conclude the following: given an $\mathcal{ALCHI}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a UACQ Q, we can construct in polynomial time a database $D_{\mathcal{A}}$, a set $\Sigma_{\mathcal{T}}$ of guarded DTGDs, and a UACQ $Q_{\mathcal{T}}$ such that $\mathcal{K} \models Q$ iff $D_{\mathcal{A}} \cup \Sigma_{\mathcal{T}} \models Q_{\mathcal{T}}$, and the next result follows:

Proposition 1. *UACQ answering under $\mathcal{ALCHI}$ KBs can be polynomially reduced to UACQ answering under guarded DTGDs.*

Clearly, the above proposition, combined with our result on UACQ answering under guarded DTGDs, implies the upper bound stated in Theorem 3.

Lower Bound. The desired lower bound is inherited from Theorem 2. More precisely, the guarded DTGDs employed in the proof of Theorem 2 have one of the following forms which corresponds to DL axioms (possibly more than one):

$$
\begin{aligned}
A(X) \rightarrow \exists Y\, R(X,Y) &\equiv A \sqsubseteq \exists R.\top \\
R(X,Y) \rightarrow A(Y) &\equiv \exists R^-.\top \sqsubseteq A \\
R(X,Y) \rightarrow \exists Z\, S(Y,Z) \vee A(Y) &\equiv \exists R^-.\top \sqsubseteq B_1 \quad B_1 \sqsubseteq B_2 \sqcup A \quad B_2 \sqsubseteq \exists S.\top \\
R(X,Y) \rightarrow A_1(Y) \vee A_2(Y) \vee A_3(Y) &\equiv \exists R^-.\top \sqsubseteq B \quad B \sqsubseteq A_1 \sqcup A_{23} \\
&\qquad\qquad A_{23} \sqsubseteq A_2 \sqcup A_3,
\end{aligned}
$$

and the next result follows.

Proposition 2. *Consider a DL $\mathcal{L}$ which is powerful enough for expressing the following inclusion assertions: $A_1 \sqsubseteq A_2 \sqcup A_3$, $A \sqsubseteq \exists R.\top$ and $\exists R^-.\top \sqsubseteq A$, where A, A_1, A_2, A_3 are concept names. Then, UACQ answering under $\mathcal{L}$ is* EXPTIME*-hard in combined complexity, even when the TBox is fixed.*

4.2 Other Description Logics.

In [3] the data complexity of answering arbitrary CQs over *DL-Lite*$^{\mathcal{H}}_{bool}$ and *DL-Lite*$_{bool}$ was investigated and shown to be coNP-complete; more recently, in [9], the combined complexity of this question was investigated and the problem shown to be 2EXPTIME-hard for the former DL. However, to the best of our knowledge, there do not exist any results on answering UACQs. Since *DL-Lite*$_{bool}$ is already equipped with limited existential quantification, role inverse and union, Theorem 3 implies the following:

Corollary 2. *UACQ answering under DL-Lite*$_{bool}$ *is* EXPTIME*-complete in combined complexity, even when the TBox is fixed.*

By extension, for every DL that is a subset of $\mathcal{ALCHI}$ and that possesses the features listed in Theorem 3, answering unions of acyclic queries is EXPTIME-complete, even if the TBox is fixed. In addition to the above corollary, this applies to, e.g., *DL-Lite*$^{\mathcal{H}}_{bool}$, $\mathcal{ELU}$, i.e., the extension of the well-known DL $\mathcal{EL}$ with union of concepts, and $\mathcal{ELUHI}$, that is, the extension of $\mathcal{ELU}$ with role hierarchies and inverse roles.

5 Conclusion

In this paper, we studied the acyclic query answering problem under guarded disjunctive TGDs. While in the combined complexity this task remains 2EXPTIME-complete, in case of fixed arity and fixed sets of rules, we observe the classical positive effect of restricting the query language to ACQs, as the complexity drops to EXPTIME. We also investigated the impact of our results on acyclic query answering under DL-based formalisms; in particular, we showed that for any DL at most as expressive as $\mathcal{ALCHI}$, the complexity of query answering is in EXPTIME if the query language is restricted to UACQs. We also showed that this problem for DLs equipped with limited existential quantification, role inverse and union is EXPTIME-hard, even when the TBox is fixed.

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

3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
4. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. of IJCAI. pp. 319–324 (2003)
5. 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)
6. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Proc. of IJCAI. pp. 712–717 (2011)
7. Bárány, V., Gottlob, G., Otto, M.: Querying the guarded fragment. In: Proc. of LICS. pp. 1–10 (2010)
8. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. of ICALP. pp. 73–85 (1981)
9. Bourhis, P., Morak, M., Pieris, A.: The impact of disjunction on query answering under guarded-based existential rules. In: Proc. IJCAI (2013)
10. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR. pp. 70–80 (2008)
11. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. (JAIR) 48, 115–174 (2013)
12. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web. Sem. 14, 57–83 (2012)
13. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artif. Intell. 193, 87–128 (2012)
14. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)
15. Calvanese, D., Giacomo, G.D., 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: Proc. of ICDT. pp. 225–241 (2003)
18. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Trans. Database Syst. 22(3), 364–418 (1997)
19. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
20. 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)
21. Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: Proc. of MFCS. pp. 1–18 (2012)
22. Grädel, E.: On the restraining power of guards. J. Symb. Log. 64(4), 1719–1742 (1999)
23. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proc. of IJCAI. pp. 963–968 (2011)
24. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: Proc. of KR (2012)
25. Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond horn ontologies. In: Proc. of IJCAI. pp. 1093–1098 (2011)
26. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: Proc. of KR (2012)