Tractable Query Answering for Expressive Ontologies and Existential Rules

David Carral, Irina Dragoste, Markus Krötzsch*

Center for Advancing Electronics Dresden (cfaed), TU Dresden, Germany

Abstract. The disjunctive skolem chase is a sound and complete (albeit non-terminating) algorithm that can be used to solve conjunctive query answering over DL ontologies and programs with disjunctive existential rules. Even though acyclicity notions can be used to ensure chase termination for a large subset of real-world knowledge bases, the complexity of reasoning over acyclic theories still remains high. Hence, we study several restrictions which not only guarantee chase termination but also ensure polynomiality. We include an evaluation that shows that almost all acyclic DL ontologies do indeed satisfy these general restrictions.

1 Introduction

Answering conjunctive queries (CQs) over knowledge bases is an important reasoning task with many applications in data management and knowledge representation. A flurry of research efforts have significantly improved our understanding of this problem, and led to different solutions for description logics (DL) ontologies [2, 6, 25] and programs with disjunctive existential rules [1, 5]. One such proposed approach is the use of *acyclicity notions* [9, 10, 19, 21]; i.e., sufficient conditions that guarantee termination of the *disjunctive chase algorithm* [3] – a sound and complete materialization-based procedure where all relevant consequences of a knowledge base are precomputed, allowing queries to be directly evaluated over materialized sets of facts. As shown in [9, 10], acyclicity notions can be used to determine that the chase will indeed terminate over a large subset of real-world DL ontologies.

Nevertheless, even if a knowledge base is characterized as acyclic, CQ answering still remains a problem of high theoretical complexity: CQ answering over acyclic programs with disjunctive existential rules is coN2ExpTime-complete [7]. For acyclic Horn-$\mathcal{SROIQ}$ ontologies, it is ExpTime-complete [10].

Example 1. Let $\mathcal{R}_n = \{D_{i-1}(x) \rightarrow \exists y_i.L_i(x, y_i) \wedge D_i(y_i), D_{i-1}(x) \rightarrow \exists z_i.R_i(x, z_i) \wedge D_i(z_i) \mid i = 1, \ldots, n\}$. The chase of the program $\mathcal{P} = \langle \mathcal{R}_n, \{D_0(c)\}\rangle$, depicted in Figure 1, is exponentially large in n. Note that, $\mathcal{P}$ is acyclic with respect to all notions described in [10] and can be expressed in most DL fragments.

* The author thanks the competent and friendly staff of trauma surgery ward OUC-S2 at the University Hospital *Carl Gustav Carus*, Dresden, where some of this research has been executed.

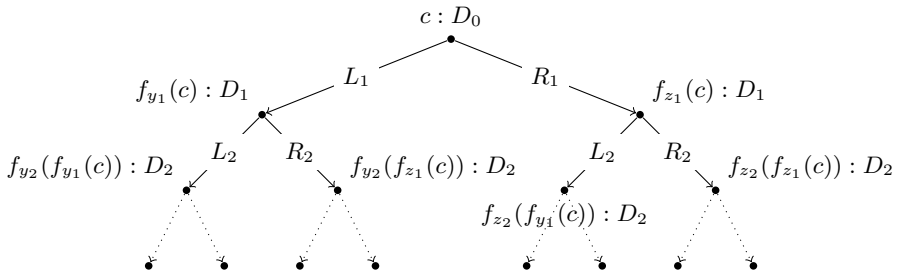

Fig. 1. Graphical Representation of the Chase of $\mathcal{P}$.

In this paper, we study the limits of tractable reasoning using the chase and propose a series of restrictions that, if combined, prevent the exponential blow-up highlighted in the previous example. Moreover, we define a novel acyclicity notion, namely *tractable acyclicity*, tailored for DL ontologies, which ensures that the size of the chase stays polynomial. In turn, this implies that CQ answering over deterministic "tractably acyclic" ontologies is (theoretically) as hard as solving the same problem over a given set of facts. On the practical side, we assess the generality of tractable acyclicity using two different corpuses of real-world ontologies. As it turns out, our notion does characterize almost all acyclic ontologies, thus showing that CQ answering may be quite efficient in practice.

In summary, our main contributions are as follows:

- We consider five general restrictions on the expressivity of rules and ontologies, and thoroughly study the complexity of CQ answering when combinations of these restrictions are satisfied (Section 3).
- Using some of these restrictions, we define tractable acyclicity, a notion specially tailored for DL ontologies which guarantees tractability of reasoning over expressive deterministic ontologies (Section 4). To the best of our knowledge, the use of notion is the only approach to guarantee tractable CQ answering over ontologies besides the combined approach [12, 17, 18, 20, 25].
- We empirically study the generality of tractable acyclicity on two large corpuses of real-world ontologies with encouraging results (Section 5).

2 Preliminaries

Let $\mathbf{P}$, $\mathbf{V}$ and $\mathbf{F}$ be some infinite countable and pairwise disjoint sets of *predicates*, *variables* and *function symbols*, respectively, such that every $S \in \mathbf{P} \cup \mathbf{F}$ is associated with some arity $ar(S) \geq 0$. *Constants* are function symbols of arity 0. *Terms* are built from variables and function symbols as usual. We abbreviate a sequence of terms $t_1, \ldots, t_n$ with $\boldsymbol{t}$, and identify such a sequence with the set $\{\boldsymbol{t}\}$. An *atom* is a formula of the form $P(\boldsymbol{t})$ with P a $|\boldsymbol{t}|$-ary predicate. With $\varphi[\boldsymbol{x}]$ we stress that $\boldsymbol{x}$ are the free variables in the formula φ. We identify a conjunction of formulas with the set of all the formulas in the conjunction and vice-versa.

A *(disjunctive existential) rule* is a first-order logic (FOL) formula of the form

$$\forall \boldsymbol{x}, \boldsymbol{y}.(B[\boldsymbol{x}, \boldsymbol{y}] \rightarrow \vee_{i=1}^{n} \exists \boldsymbol{v}_i.H_i[\boldsymbol{x}, \boldsymbol{v}_i]) \tag{1}$$

where B (the *body*) and H_i (the *heads*) are conjunctions of atoms with $H_i \neq \emptyset$ for all $i = 1, \ldots, n$; and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$, $\boldsymbol{y}$ and $\boldsymbol{x}$ are pairwise disjoint. For the sake of brevity, we omit universal quantifiers when writing rules. The variables in $\boldsymbol{x}$ are called *frontier variables*. A rule is *Horn* if $n = 1$ and *non-Horn* otherwise. A *fact* is a *ground atom*; i.e., an atom without occurrences of variables. An *instance* $\mathcal{I}$ is a finite set of facts only containing constants as terms. A *program* is a pair $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set and $\mathcal{I}$ an instance. Without loss of generality, we assume that every existentially quantified variable occurs in at most one rule (†).

The main reasoning task we are studying in this paper is CQ answering. Nevertheless, without loss of generality, we restrict our attention to the simpler task of entailment of Boolean conjunctive queries (BCQs). A *BCQ*, or simply a *query*, is a formula of the form $\exists \boldsymbol{y}.Q[\boldsymbol{y}]$ with Q a conjunction of atoms.

A *substitution* is a partial function defined over the set of terms. The *application of a substitution σ to an atom α*, denoted with $\alpha\sigma$, is the atom that results from replacing all occurrences of every term t in the domain of σ with $\sigma(t)$. We denote the substitution $\{(t_1, u_1), \ldots, (t_n, u_n)\}$ with $[t_1/u_1, \ldots, t_n/u_n]$.

The *skolemization* $sk(\rho)$ of a rule ρ as in (1) is the formula $B \rightarrow \bigvee_{i=1}^{n} sk(H_i)$ where, for every $i = 1, \ldots, n$, $sk(H_i)$ is the conjunction that results from replacing every (existentially quantified variable) $v \in \boldsymbol{v}_i$ by the term $f_v(\boldsymbol{x})$ with f_v a fresh function symbol specific to v (which, by assumption (†) and the definition of a rule, is also specific to the i-th disjunct in the head of the rule ρ).

Definition 2. *Consider a rule ρ of the form* (1)*, a substitution σ defined only on $\boldsymbol{x} \cup \boldsymbol{y}$, and a set of facts $\mathcal{F}$. Then, $\langle \rho, \sigma \rangle$ is* applicable *to $\mathcal{F}$ if $B\sigma \subseteq \mathcal{F}$. In this case, the* result of applying $\langle \rho, \sigma \rangle$ to $\mathcal{F}$ *is* $\{\mathcal{F} \cup sk(H_i)\sigma \mid i = 1, \ldots, n\}$.

A chase tree *of $\langle \mathcal{R}, \mathcal{I} \rangle$ is a (possibly infinite) tree where each node is labeled by a set of facts, such that all of the following conditions hold.*

1. *The root is labeled with $\mathcal{I}$.*
2. *If a node labeled with $\mathcal{F}$ has n children labeled with $\mathcal{F}_1, \ldots, \mathcal{F}_n$, then there is some rule $\rho \in \mathcal{R}$ and some substitution σ such that $\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is the result of applying $\langle \rho, \sigma \rangle$ to $\mathcal{F}$.*
3. *(Fairness) If there is a node α labeled with a set $\mathcal{F}$, a rule $\rho \in \mathcal{R}$, and a substitution σ such that $\langle \rho, \sigma \rangle \in \mathcal{R}$ is applicable to $\mathcal{F}$; then, in all paths starting from α, there is some node β with n children, each of them labeled with a different set in the result of applying $\langle \rho, \sigma \rangle$ to the label of β.*

The result of the (Skolem) chase is the (possibly infinite) set of all (possibly infinite) sets of facts obtained as the union of all sets of facts along some path.

Due to the order of rule applications, a program $\mathcal{P}$ may admit many different chase trees but, nevertheless, the result of the Skolem chase of $\mathcal{P}$ is always unique.

Fact 3 *A program $\mathcal{P}$ entails a query $\exists \boldsymbol{v}.Q$ if and only if $\mathcal{F} \models \exists \boldsymbol{v}.Q$ holds for every set of facts $\mathcal{F}$ in the result of the chase of $\mathcal{P}$.*

If the chase terminates for some program, then the result of the chase is the set of all (finite) leaf labels. In this case, Fact 3 leads to an effective decision procedure for BCQ entailment. Therefore, in the subsequent section, we study several restrictions on a set of rules which ensure efficient chase termination.

3 Tractable Reasoning for Disjunctive Existential Rules

In this section we present and study several restrictions, which can ensure tractability of BCQ entailment over rule sets. These insights will be the basis for our investigation of tractable query answering for ontologies in Section 4. An important concept for predicting the behaviour of the chase procedure is the *dependency graph* of a rule set:

Definition 4. *The* dependency graph $G(\mathcal{R})$ *of a rule set $\mathcal{R}$ has the existential variables in $\mathcal{R}$ as nodes, and an edge $y \to z$ if the skolem chase of some program $\langle \mathcal{R}, \mathcal{I} \rangle$ contains terms of the form $f_z(\boldsymbol{t})$ and $f_y(\boldsymbol{s})$ such that $f_y(\boldsymbol{s}) \in \boldsymbol{t}$.*

The key to our tractability results is the notion of a *braid*, which, intuitively speaking, consists of a possibly large number of intertwined paths.

Definition 5. *Consider a directed graph G. A* path *is a sequence of nodes $\alpha_1, \ldots, \alpha_n$ with $\alpha_i \to \alpha_{i+1} \in G$ for all $i = 1, \ldots, n-1$. The graph G is* acyclic *if, for every path $\alpha_1, \ldots, \alpha_n$ with $n \geq 2$, $\alpha_1 \neq \alpha_n$. A* simple path *is a path which does not contain two occurrences of the same node. A* braid *is a sequence of nodes $\alpha_1, \ldots, \alpha_n$ such that, for all $i = 1, \ldots, n-1$, there are at least two different simple paths from α_i to α_{i+1}.*

A number of natural conditions on a set of rules $\mathcal{R}$ might be considered in order to reduce the complexity of the chase. We will consider the following five:

(a) The graph $G(\mathcal{R})$ is acyclic.
(f) The arity of all function symbols in $sk(\mathcal{R})$ is at most 1.
(b) The length of the braids in $G(\mathcal{R})$ is bounded.
(w) The treewidth of the rules in $\mathcal{R}$ is bounded.
(p) The arity of the predicates in $\mathcal{R}$ is bounded.

Most of these conditions are self-explanatory and straightforward to check. The treewidth of rules is the treewidth of the graph that has the terms of a rule as nodes, and an undirected edge whenever two terms appear in the same atom [13]. It is a well-known measure for "tree-likeness", which is bounded by the number of terms per rule.[1] Checking if a graph G has treewidth at most k for a given constant k is polynomial in G. Both acyclicity and the maximal braid

[1] Readers not familiar with treewidth may safely use this number as a surrogate.

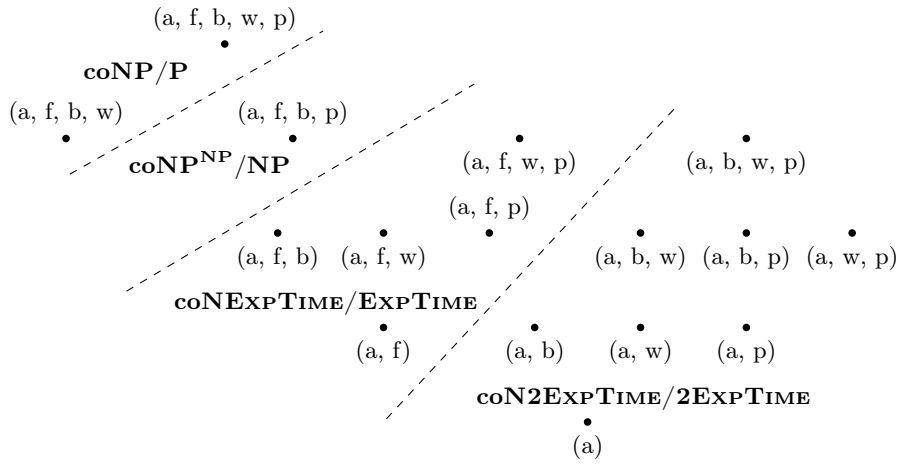

Fig. 2. Complexity of BCQ entailment with respect to the size of the rule set satisfying some combination of (a), (f), (b), (w) and (p). All of the above results are tight and refer to the combined complexity of BCQ entailment over nondeterministic and deterministic rule sets, respectively.

length can be computed efficiently if the dependecy graph is known. We present ways of approximating these conditions efficiently in Section 4.

In the remainder of the section, we characterize the (combined) complexity of BCQ entailment over sets of rules satisfying every possible combination of the above restrictions. We summarize our findings in Figure 2, which only includes cases that satisfy (a), since its omission leads to undecidability (Theorem 11). Moreover, as indicated in Theorem 7, the "coNP/ P" result refers to the complexity regarding the size of the rule set, with the query considered fixed.

Whilst restrictions (a), (f), (w), and (p) have been considered in previous work [10], (b) is a novel notion instrumental to ensure tractability of reasoning. See how the rule set from Example 1 may not satisfy such a restriction.

Example 6. Let $\mathcal{R}_n$ be the set of rules presented in Example 1 and let $G(\mathcal{R}_n)$ be the graph depicted in Figure 3. Note how, for every every odd $n \geq 1$, there is a braid of length $(n+1)/2$ in $G(\mathcal{R}_n)$; e.g., $z_1, z_3, \ldots, z_n$ or $y_1, y_3, \ldots, y_n$.

Combining all restrictions allows us to obtain the main result of this section.

Theorem 7. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a), (f), (b), and (w) is in co*NP *provided that the size of the query is fixed. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is in* P.

The key for proving this result is a property that relates braid length to the size of the chase. As we will show, if a rule set $\mathcal{R}$ satisfies (f), then every term in the chase of $\langle \mathcal{R}, \mathcal{I} \rangle$ corresponds to some path in $G(\mathcal{R})$ and some constant. In

turn, this implies that, if there is a polynomial bound on the number of paths in $G(\mathcal{R})$, then the number of terms introduced during the computation of the chase of $\langle \mathcal{R}, \mathcal{I} \rangle$ is also polynomially bounded. Therefore, we first show that there is indeed such a polynomial upper bound on the number of paths in a graph if the length of the braids in such a graph is fixed. Once this is shown, we can easily verify that, if $\mathcal{R}$ satisfies (b) and (f), then there is a polynomial upper bound on the number of terms that may occur in the chase of a program $\langle \mathcal{R}, \mathcal{I} \rangle$.

Lemma 8. *Consider some directed acyclic graph G with n nodes. If there is a bound k on the length of the braids, then there are at most $3k \cdot n^{3k}$ paths in G.*

Proof. First, we verify the following intermediate result: $(*)$ Consider two nodes α and β in G. If, for every node γ, the sequence α, γ, β is not a braid in G; then $P_G(\alpha, \beta) \leq 3n^2$ with $P_G(\alpha, \beta)$ the number of paths from α to β.

Let G' be the graph that results from removing every node γ not occurring in a path from α to β. Then, for every node γ in G' with $\gamma \neq \alpha$ and $\gamma \neq \beta$, $P_{G'}(\alpha, \gamma) = 1$ or $P_{G'}(\gamma, \beta) = 1$. Let G'' be the graph obtained from G' via simultaneous application of the following rules to every node γ in G': If $P_{G'}(\alpha, \gamma) = 1$ and $\alpha \to \gamma \notin G'$, then remove the (only) edge of the form $\delta \to \gamma \in G'$ and add $\alpha \to \gamma$. If $P_{G'}(\gamma, \beta) = 1$ and $\gamma \to \beta \notin G'$, then remove the edge of the form $\gamma \to \delta \in G'$ and add $\gamma \to \beta$.

The previously presented transformation preserves the number of paths from α to β; i.e., $P_G(\alpha, \beta) = P_{G''}(\alpha, \beta)$. Moreover, the nodes in G'' can be fully distributed into four pairwise disjoint s ets L_1, L_2, L_3 and L_4 such that all of the following hold: $L_1 = \{\alpha\}$; $L_4 = \{\beta\}$; and, for every pair of nodes γ and δ, $\gamma \to \delta \in G''$ only if (i) $\gamma = \alpha$ and $\delta \in L_2 \cup L_4$, (ii) $\gamma \in L_2$ and $\delta \in L_3 \cup L_4$, or (iii) $\gamma \in L_3$ and $\delta = \beta$. As the sets L_2 and L_3 may contain at most n nodes, then $P_{G''}(\alpha, \beta) \leq n^2 + n + 1 \leq 3n^2$ (as n is at least 2).

We now proceed to show the lemma. Let B_m be the set of paths containing a path p in G iff (i) p contains a braid of length m and (ii) p does not contain a braid of length $m + 1$. Then, every path $p \in B_m$ is of the form $\alpha_1, s_1, \alpha_2, s_2, \ldots, s_{m-1}, \alpha_m$ where α_1, ..., α_m is a braid; and, for every $i = 2, \ldots, m - 1$, s_i is a sequence of nodes not containing a node γ such that $\alpha_i, \gamma, \alpha_{i+1}$ is a braid in G (as this would imply that p contains a braid of length $m + 1$). By $(*)$, there are at most $3n^2$ possible paths in G for every s_i. Moreover, there are at most n^m braids of length m. Therefore, B_m contains at most $n^m \cdot 3n^{2(m-1)} \leq 3n^{3m}$ paths. The number of paths in G is at most $\sum_{i=0}^{k} |B_i| \leq k|B_k|$ (as every B_j with $j > k$ is empty). Hence, the number of paths in G is necessarily less than $3k \cdot n^{3k}$. □

Proof (of Theorem 7). Since $\mathcal{R}$ satisfies (w), we can apply a normalization procedure to compute a conservative extension $\langle \mathcal{R}', \mathcal{I} \rangle$ of $\langle \mathcal{R}, \mathcal{I} \rangle$ with an upper bound on the number of variables per rule [13]. Moreover, this transformation does not modify the dependency graph of $\mathcal{R}$ (i.e., $G(\mathcal{R}) = G(\mathcal{R}')$).

We first determine an upper bound on the maximal number of terms T and atoms A that may occur in the chase of $\langle \mathcal{R}', \mathcal{I} \rangle$. By (f), every term in the chase

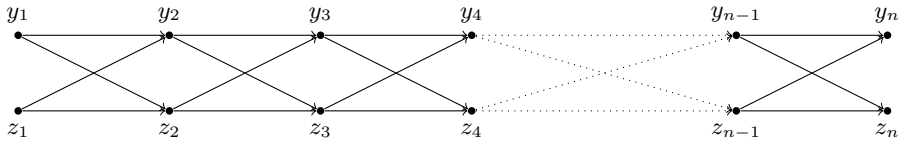

Fig. 3. Dependency Graph of the Set of rules $\mathcal{R}_n$ from Example 6.

of $\langle \mathcal{R}', \mathcal{I} \rangle$ is of the form $f_{y_n}(\ldots(f_{y_1}(c))\ldots)$ with c a constant. Furthermore, such a term occurs in the chase only if $y_1, \ldots, y_n$ is a path in $G(\mathcal{R}')$. Hence, every term in the chase of $\langle \mathcal{R}', \mathcal{I} \rangle$ corresponds to some path in $G(\mathcal{R}')$ and some constant. By Lemma 8 and the fact that $\mathcal{R}'$ satisfies (b), we conclude that the number of paths in $G(\mathcal{R}')$ is polynomial in the number of nodes in $G(\mathcal{R}')$ (which coincides with the number of existentially quantified variables in $\mathcal{R}'$). Therefore, T is polynomially large with respect to $\langle \mathcal{R}', \mathcal{I} \rangle$ and, since the number of variables per rule in $\mathcal{R}'$ is fixed, so is A. If $\mathcal{R}'$ is a set of deterministic rules, then we can compute the only branch on some (arbitrarily chosen) chase tree of $\langle \mathcal{R}', \mathcal{I} \rangle$ to solve BCQ entailment. This branch is a sequence of at most A sets of facts; and, as there is an upper bound on the number of variables per rule in $\mathcal{R}'$, each of these sets can be computed in polynomial time. Moreover, checking if the facts in the branch entail a query is in P if the size of the query is fixed.

In the nondeterministic case, we can guess some sequence of facts and then check whether (i) such a sequence is a complete branch in some chase tree of $\langle \mathcal{R}', \mathcal{I} \rangle$. Then, a query is not entailed by $\langle \mathcal{R}', \mathcal{I} \rangle$ iff (ii) it is not entailed by the facts in this branch. Note that, (i-ii) can be checked in P. □

We proceed by showing the complexity of BCQ answering for any other combination of the restrictions (a), (f), (b), (w), and (p). This shows, in particular, that our chosen set of restrictions is minimal (among these selected conditions) and any other combination leads to intractability.

Theorem 9. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a), (f), (b), (w), and (p) is co*NP*-hard. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is* P*-hard.*

Proof. The results stated in the theorem follow from hardness of SAT and propositional Horn logic entailment, respectively. □

Theorem 10. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a), (f), (b), and (p) is in co*$\mathrm{NP}^{\mathrm{NP}}$*-complete. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is in* NP*-complete.*

Proof. To show membership, we can make an analogous argument to the one in the proof of Theorem 7 to show that there is a polynomial upper bound on the number of terms T that may occur during the computation of the chase $\langle \mathcal{R}, \mathcal{I} \rangle$.

Moreover, since the arity of the predicates is bounded by some ℓ, the number of atoms in the chase is at most $A = P^{\ell} \cdot T$.

If $\mathcal{R}$ is a set of deterministic rules, then we can guess some sequence of sets $\mathcal{F}_1, \ldots, \mathcal{F}_n$ of facts with $\mathcal{F}_1 = \mathcal{I}$; some sequence $\langle \rho_1, \sigma_1 \rangle, \ldots, \langle \rho_{n-1}, \sigma_{n-1} \rangle$ of pairs of rules and substitutions with $\rho_i \in \mathcal{R}$ for every $i = 1, \ldots, n-1$; and some additional substitution σ. To determine if $\langle \mathcal{R}, \mathcal{I} \rangle$ entails some query Q, we check that, for every $i = 1, \ldots, n-1$, (i) $\mathcal{F}_{i+1}$ is the result of the application of $\langle \rho_i, \sigma_i \rangle$ on $\mathcal{F}_i$; and (ii) $\mathcal{F}_n \models Q\sigma$. Note that, (i-ii) can be verified in polynomial time, and $\mathcal{F}_1, \ldots, \mathcal{F}_n$ may not necessarily be a complete branch in a chase tree of $\langle \mathcal{R}, \mathcal{I} \rangle$.

If $\mathcal{R}$ is a set of nondeterministic rules, then we simply guess some sequence of sets $\mathcal{F}_1, \ldots, \mathcal{F}_n$ of facts with $\mathcal{F}_1 = \mathcal{I}$. To determine that $\langle \mathcal{R}, \mathcal{I} \rangle$ does not entail some query Q, we check that, for every $i = 1, \ldots, n-1$, $(i)\mathcal{F}_{i+1}$ is the result of the application of some rule in $\mathcal{R}$ and some substitution on $\mathcal{F}_i$; (ii) no rule in $\mathcal{R}$ and substitution is applicable to $\mathcal{F}_n$; and (iii)$\mathcal{F}_n \not\models Q$. (i-iii) can be polynomially checked using an NP oracle.

For coNP$^{\text{NP}}$-hardness, we reduce from the valuation problem of quantified Boolean formulas (QBF) of the form $\forall \boldsymbol{X}.\exists \boldsymbol{Y}.\varphi$, where $\boldsymbol{X}, \boldsymbol{Y}$ are lists of propositional variables and φ is in 3CNF, i.e., $\varphi = (L_1^1 \vee L_2^1 \vee L_3^1) \wedge \ldots \wedge (L_1^n \vee L_2^n \vee L_3^n)$, such that the literals L_j^i are variables or negated variables from $\boldsymbol{X} \cup \boldsymbol{Y}$.

We construct a set of nondeterministic without existential variables rules using constants t (true) and f (false). We add two facts $\mathsf{tf}(t)$ and $\mathsf{tf}(f)$. For every $i \in \{1, \ldots, n\}$, we add all (polynomially many) facts of the form $c_i(v_1, v_2, v_3)$ with $v_1, v_2, v_3 \in \{t, f\}$ such that $(L_1^i \vee L_2^i \vee L_3^i)$ is true when assigning the values v_1, v_2, v_3 to the (at most) three variables in the clause. In addition, for each universally quantified $X \in \boldsymbol{X}$, we add a disjunctive fact $\mathsf{val}_X(t) \vee \mathsf{val}_X(f)$. Finally, QBF valuation is encoded in the rule:

$$\bigwedge_{1 \leq i \leq n} c_i(x_1^i, x_2^i, x_3^i) \wedge \bigwedge_{X \in \boldsymbol{X}} \mathsf{val}_X(v_X) \wedge \bigwedge_{Y \in \boldsymbol{Y}} \mathsf{tf}(v_Y) \rightarrow \mathsf{trueQBF} \tag{2}$$

where each variable has the form v_Z for $Z \in \boldsymbol{X} \cup \boldsymbol{Y}$, and x_j^i denotes v_Z for the propositional variable Z that occurs in L_j^i. Then $\mathsf{trueQBF}$ is entailed iff, for all models (i.e., all assignments of universal variables $X \in \boldsymbol{X}$), there is an assignment for the variables $Y \in \boldsymbol{Y}$, such that each clause in φ is true.

The hardness result for deterministic rules follows when considering QBF without universally quantified variables; i.e., propositional satisfiability. □

Theorem 11. *BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a set of deterministic rules satisfying (f), (b), (w), and (p) is undecidable.*

Proof. We use a reduction from a known undecidable problem described as follows (see Section 2.5.1 of [16] for a very similar and more detailed argument). A context-free grammar is a tuple $\langle S, P \rangle$ with S a non-terminal, and P a set of production rules of the form $A \rightarrow BC$ or $A \rightarrow a$ where A, B and C are non-terminals and a is a terminal. The language generated by a grammar $\langle S, P \rangle$ is the set of all strings of terminals which can be produced by rewriting S applying the production rules in P. The following problem is undecidable [14]: Given two

context-free grammars $G_1 = \langle P_1, S_1 \rangle$ and $G_2 = \langle P_2, S_2 \rangle$, with disjoint sets of non-terminals and common terminal symbols 0 and 1, determine whether there is some word in the intersection of the languages generated by G_1 and G_2.

Consider two binary predicates T_0 and T_1, a specific binary predicate NT_A for every non-terminal A occurring in G_1 or G_2, a unary predicate X, and a constant c. For all $i \in \{1, 2\}$, let $\mathcal{R}_i = \{T_a(x, y) \rightarrow NT_A(x, y) \mid A \rightarrow a \in P_i\} \cup \{NT_B(x, y) \wedge NT_C(y, z) \rightarrow NT_A(x, z) \mid A \rightarrow BC \in P_i\}$. Moreover, let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \{X(x) \rightarrow \exists y.T_0(x, y) \wedge X(y), X(x) \rightarrow \exists z.T_1(x, z) \wedge X(z)\}$. Then, the intersection of the languages generated by G_1 and G_2 is empty iff $\langle \mathcal{R}, \{X(c)\} \rangle$ does not entail the query $\exists x.NT_{S_1}(c, x) \wedge NT_{S_2}(c, x)$.

The rules in $\mathcal{R}$ satisfy (f), (b), (w), and (p): The arity of all of the symbols in $sk(\mathcal{R})$ (i.e., f_y and f_z) is one, $G(R)$ contains two nodes, and the arity of every predicate is at most two. Moreover, the number of variables per rule is bounded and hence, so is the treewidth. □

Theorem 12. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a) is in co*N2ExpTime*. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is in* 2ExpTime*.*

Proof. We first determine the maximal number of ground (skolem) terms and corresponding facts that may occur in the chase. Let n be the number of skolem functions in $sk(\mathcal{R})$, and let m be the maximal arity of such functions. The maximal nesting depth of ground terms in the chase is n, since every term of greater depth is cyclic and, by (a), such terms may not occur in the chase of $\langle \mathcal{R}, \mathcal{I} \rangle$. Ground terms then correspond to trees of depth at most n, fan-out at most m, and with leaves from the set C of constants in $\langle \mathcal{R}, \mathcal{I} \rangle$. Such trees have most $n \cdot m^n$ nodes in total. As each node is assigned a constant or function symbol, there are at most $T = (|C| + n)^{n \cdot m^n}$ trees, and hence ground terms, overall. Now, if $\langle \mathcal{R}, \mathcal{I} \rangle$ contains k different predicate symbols of arity at most ℓ, then the maximal number of ground facts based on T terms is $A = kT^\ell = k(C_\mathcal{I} + n)^{\ell \cdot n \cdot m^n}$. The number of facts A is therefore double exponential in the size of $\langle \mathcal{R}, \mathcal{I} \rangle$ and hence, so is the length of every branch in a chase tree of a program $\langle \mathcal{R}, \mathcal{I} \rangle$.

If $\mathcal{R}$ is a set of deterministic rules, then there is only one branch in every possible chase tree of a program $\langle \mathcal{R}, \mathcal{I} \rangle$ which can be computed in double-exponentially many steps. Then, a query is entailed by $\langle \mathcal{R}, \mathcal{I} \rangle$ iff such query is entailed by the set of facts in the branch. If $\mathcal{R}$ only contains nondeterministic rules, membership in coN2ExpTime follows from the fact that BCQ non-entailment can be shown by guessing some branch of the tree, and then checking that the set of facts in such branch does not entail the query. □

Theorem 13. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a) and (f) is in co*NExpTime*. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is in* ExpTime*.*

Proof. We determine that the maximal number of facts that may occur in the chase of $\langle \mathcal{R}, \mathcal{I} \rangle$ is exponential in the size of the program. The remainder of the proof is analogous to that of Theorem 12.

Let n be the number of skolem functions in $sk(\mathcal{R})$ which, by (f), have an arity of at most 1. The maximal nesting depth of ground terms in the chase is n, since every term of greater depth is cyclic and, by (a), such terms may not occur in the chase of $\langle \mathcal{R}, \mathcal{I} \rangle$. Ground terms then correspond to sequences of depth at most n and, since each element in the sequence is assigned a constant or function symbol, there are at most $T = (C + n)^n$ ground terms, overall. In turn, the maximal number of facts in the chase is $A = kT^\ell = k(C + n)^{\ell \cdot n}$ with k the number of predicates and ℓ the maximal arity of a predicate in $\langle \mathcal{R}, \mathcal{I} \rangle$. □

Theorem 14. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a), (b), (w), and (p) is co*N2ExpTime*-hard. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is* 2ExpTime*-hard.*

Proof. For the first result, we present a reduction of the word problem of double-exponentially time-bounded non deterministic Turing machines (TMs) to BCQ non-entailment. Given such reduction, it is clear how to produce a similar reduction to prove the second result stated in the theorem.

Consider a N2ExpTime Turing Machine (TM) M. We simulate the computation of M on an input string I by constructing a program $\langle \mathcal{R}, \mathcal{I} \rangle$ such that $\langle \mathcal{R}, \mathcal{I} \rangle$ does not entail some nullary predicate *Reject* iff M accepts I. To address computation steps and tape cells, we recall a construction by [4] to (deterministically) construct a chain of double exponentially many elements. Let $\mathcal{I} = \{r_0(0), r_0(a), Scc_0(0,1), Min_0(0), Max_0(a)\}$. For each $i \in \{0, \ldots, n-1\}$, with n the length of the input I, we add the rules in $\{R_i(x) \wedge R_i(y) \rightarrow \exists v_i . S_i(x, y, v_{i+1}) \wedge R_{i+1}(v_{i+1}), S_i(x,y,z) \wedge S_i(x,y',z') \wedge Scc_i(y,y') \rightarrow Scc_{i+1}(z,z'), S_i(x,y,z) \wedge S_i(x',y',z') \wedge Max_i(y) \wedge Min_i(y') \wedge Scc_i(x,x') \rightarrow Scc_{i+1}(z,z'), Min_i(x) \wedge S_i(x,x,y) \rightarrow Min_{i+1}(y), Max_i(x) \wedge S_i(x,x,y) \rightarrow Max_{i+1}(y)\}$ It can be shown, by induction on i, that in any path of any chase tree of $\langle \mathcal{R}, \mathcal{I} \rangle$, the relation r_n contains 2^{2^n} elements, which are linearly ordered by Scc_n.

The remaining TM simulation follows standard constructions (cf. [11]), using elements of the r_n chain to refer to specific time points and tape cells when encoding a run of the TM. Nondeterministic transitions are captured using rules with disjunction. Assuming that the state of M at step s is captured with facts $State_q(s)$ for all states Q, we can complete the simulation by adding rules $State_q(x) \wedge Max_n(x) \rightarrow Reject$ for all non-accepting states q of M. We can assume without loss of generality that M runs for the maximum double-exponential number of steps on all rejecting runs, so that the query *Reject* is entailed iff there are no accepting runs.

The rules in $\mathcal{R}$ satisfy (a), (b), (w), and (p): $G(\mathcal{R})$ is the smallest graph containing $v_i \rightarrow v_{i+1}$ for every $i = 1, \ldots, n$ and hence, this graph is acyclic and does not contain any braids. Also, both the arity of the predicates, and treewidth of the rules in $\mathcal{R}$ is fixed. Finally, it can be checked that the rules added to finalize the reduction (cf. [11]) do not violate (a), (b), (w), nor (p). □

Theorem 15. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I} \rangle$ with $\mathcal{R}$ a rule set satisfying (a), (f), (w), and (p) is co*NExpTime*-hard. If $\mathcal{R}$ is a set of deterministic rules, then it is* ExpTime*-hard.*

Proof. We show that, using a set of rules satisfying (a), (f), (w), and (p), we can define a program that, given some n, can generate an exponentially long chain of terms sorted by some binary predicate. The remainder of the proof is analogous to that of Theorem 14.

Let $\mathcal{R}$ be the set containing the rules in $\{S_i(x) \rightarrow \exists y_{i+1}, z_{i+1}.L_i(x,y) \wedge R_i(x,z_{i+1}) \wedge Scc_{i+1}(y_{i+1},z_{i+1}), R_i(x,z) \wedge Scc_i(x,y) \wedge L_i(y,w) \rightarrow Scc_{i+1}(z,w)\}$ for each $i \in \{0,\ldots,n-1\}$. We can show, by induction on i, that in any path of any chase tree of $\langle \mathcal{R}, \{S_0(c)\}\rangle$, the relation S_n contains 2^n elements, which are linearly ordered by Scc_n.

The rules in $\mathcal{R}$ satisfy (a), (f), (w), and (p): $G(\mathcal{R})$ is the smallest graph containing $y_i \rightarrow y_{i+1}$, $y_i \rightarrow z_{i+1}$, $z_i \rightarrow y_{i+1}$, and $z_i \rightarrow z_{i+1}$ for every $i = 1,\ldots,n$, and hence, this graph is acyclic. Also, the arity of every function symbol in $sk(\mathcal{R})$ is 1, and both the arity of the predicates and treewidth of the rules is fixed. □

Theorem 16. *Deciding BCQ entailment for programs $\langle \mathcal{R}, \mathcal{I}\rangle$ with $\mathcal{R}$ a rule set satisfying (a), (f), and (b) is co*NExpTime*-hard. Moreover, if $\mathcal{R}$ is a set of deterministic rules, then it is* ExpTime*-hard.*

Proof. The first and second parts of the theorem follow from the hardness of fact entailment over disjunctive and non-disjunctive Datalog [11], respectively. Note that, every (possibly disjunctive) Datalog program – a program containing only deterministic rules without existential variables – satisfies (a), (f) and (b). □

4 Tractable Reasoning for Ontologies

Across this section we discuss how to employ the chase to reason over DL ontologies and then, using some of the results from the previous section, we define *tractable acyclicity*, an acyclicity condition tailored for DL ontologies which ensures tractability of BCQ entailment.

We consider the $\mathcal{SRI}$ fragment of the description logic $\mathcal{SROIQ}$, which is the logical basis of OWL 2 DL. We present this DL using a normal form close to that of [8]. Note that, in such a normal form, occurrences of the negation logical constructor are normalized into axioms of the form (3) in Figure 4. Moreover, we do not consider number restrictions nor nominals in our definition of DL, as the use of these logical constructors would require *equality reasoning*. There are well-known techniques to axiomatize the meaning of equality – e.g., singularization [10, 21] – but these are not our focus.

Let $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{I}$ be some infinite countable and pairwise disjoint sets of *concepts*, *roles*, and *individuals*, respectively. Moreover, let $\mathbf{R}^- = \mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$; and, for every $R \in \mathbf{R}^-$, $R^{--} = R$. A *TBox axiom* is a formula of one of the forms given on the left hand side of Figure 4. An *ABox axiom* or *assertion* is a formula of the form $A(a)$ or $R(a,b)$ with $A \in \mathbf{C}$, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$. A *ontology* is a tuple $\langle \mathcal{T}, \mathcal{A}\rangle$ with $\mathcal{T}$ a set of TBox axioms and $\mathcal{A}$ a set of assertions.

We do not consider any structural restrictions, such as role regularity [15], in our definition of ontologies. These restrictions are unnecessary for preserving correctness when using the chase and hence, we ignore them.

$$\prod_{i=1}^{n} C_i \sqsubseteq \bigsqcup_{i=1}^{m} D_i \;\mapsto\; \bigwedge_{i=1}^{n} C_i(x) \rightarrow \bigvee_{i=1}^{m} D_i(x) \tag{3}$$

$$C \sqsubseteq \forall R.D \;\mapsto\; C(x) \wedge R\langle x, y\rangle \rightarrow D(y) \tag{4}$$

$$C \sqsubseteq \exists R.Self \;\mapsto\; C(x) \rightarrow R\langle x, x\rangle \tag{5}$$

$$\exists S.Self \sqsubseteq D \;\mapsto\; R\langle x, x\rangle \rightarrow D(x) \tag{6}$$

$$\prod_{i=1}^{n} S_i \sqsubseteq R \;\mapsto\; \bigwedge_{i=1}^{n} S_i\langle x, y\rangle \rightarrow R\langle x, y\rangle \tag{7}$$

$$S_1 \circ \ldots \circ S_n \sqsubseteq R \;\mapsto\; S_1\langle x_0, x_1\rangle \wedge \ldots \wedge S_n\langle x_{n-1}, x_n\rangle \rightarrow R\langle x_0, x_n\rangle \tag{8}$$

$$C \sqsubseteq \exists R.D \;\mapsto\; C(x) \rightarrow \exists y.R\langle x, y\rangle \wedge D(y) \tag{9}$$

Fig. 4. Mapping Ψ. In the above, $C_{(i)}, D \in \mathbf{C}$, $R, S_{(i)} \in \mathbf{R}^-$, and $m, n \geq 1$. Moreover, for every $R \in \mathbf{R}$, $R^-\langle t, u\rangle = R(u, t)$ and $R\langle t, u\rangle = R(t, u)$.

The semantics of ontologies are given by means of a mapping into programs.

Fact 17 *An ontology $\mathcal{O}$ entails some query Q iff $\langle \Psi(\mathcal{R}), \mathcal{I}\rangle \models Q$ with Ψ the function mapping axioms to rules defined in Figure 4.*

Due to the close correspondence between DL axioms and rules highlighted by the previous result, we identify an axiom α with the rule $\Psi(\alpha)$, a TBox $\mathcal{T}$ with the set of rules $\Psi(\mathcal{R})$, and an ontology $\langle \mathcal{T}, \mathcal{A}\rangle$ with the program $\langle \Psi(\mathcal{R}), \mathcal{A}\rangle$.

By definition, every TBox $\mathcal{T}$ satisfies restrictions (f) and (w) and hence, we only need to determine whether $\mathcal{T}$ satisfies (a) and (b) to guarantee tractability of reasoning over a deterministic ontology $\langle \mathcal{T}, \mathcal{A}\rangle$. Unfortunately, the dependency graph of a TBox – which needs to be checked in order to verify (a) and (b) – cannot be computed in polynomial time.

Lemma 18. *Given a TBox $\mathcal{T}$, the computation of $G(\mathcal{T})$ is* ExpTime*-hard.*

Proof. The lemma follows from the fact that entailment of concept subsumptions by a TBox (which is ExpTime-hard) can be decided by computing the dependency graph of another TBox $\mathcal{T}'$.

Consider a TBox $\mathcal{T}$ and two concepts C and D. Moreover, let $\mathcal{T}' = \mathcal{T} \cup \{\alpha_1 = C \sqsubseteq \exists R_Y.C \sqcap Y, \alpha_2 = Y \sqcap D \sqsubseteq \exists R_Z.Z\}$ with R_Y and R_Z, and Y and Z some fresh roles and concepts, respectively. Then, $\mathcal{T} \models C \sqsubseteq D$ iff $y \rightarrow z \in G(\mathcal{T}')$ with y and z the variables occurring in the rules $\Psi(\alpha_1)$ and $\Psi(\alpha_2)$. □

Since the computation of the dependency graph of a TBox is rather expensive, we define an over-approximation of this graph based on the definition of model-summarizing acyclicity (MSA) [10] which can be computed more efficiently.

Definition 19. *Given a set of rules $\mathcal{R}$, let $\mathcal{R}_S$ be the set of rules that results from replacing every rule $\rho \in \mathcal{R}$ of the form (1) by the following rule.*

$$B \rightarrow \bigwedge_{1 \leq i \leq n} \Big(H_i \wedge \bigwedge_{x \in \boldsymbol{x}} \bigwedge_{v \in \boldsymbol{v}_i} Scc(x, v) \Big)\theta \tag{10}$$

In the above, Scc is a fresh binary predicate and θ is the substitution mapping every variable in $v \in \boldsymbol{v}_i$ to a fresh constant c_v (which, by (†) and the definition of a rule, is also specific to the i-th disjunct in the head of the rule ρ).

The summarizing dependency graph $G_S(\mathcal{R})$ *of a rule set $\mathcal{R}$ is the smallest graph containing an edge $y \to z$ if $\langle \mathcal{R}_S, \mathcal{I}^\star_\mathcal{R} \rangle \models Scc(c_y, c_z)$ where $\mathcal{I}^\star_\mathcal{R}$ is the* critical instance of $\mathcal{R}$*; i.e., the set of all facts that can be constructed using the predicates in $\mathcal{R}$ and the special constant $\star$.*

Lemma 20. *Consider a rule set $\mathcal{R}$. Then, the summarizing dependency graph of $\mathcal{R}$ is a superset of the dependency graph of $\mathcal{R}$.*

Proof. Consider some chase tree T of a program $\langle \mathcal{R}, \mathcal{I} \rangle$; and a function h mapping every constant to $\star$, and every skolem term of the form $f_y(\boldsymbol{t})$ to the constant c_y. Then, for every set of facts $\mathcal{F}$ associated to some node α in T, $h(\mathcal{F})$ is contained in the result of the chase of $\langle \mathcal{R}_S, \mathcal{I}^\star_\mathcal{R} \rangle$. The previous claim can be verified by induction on the path from the root of T to α.

Let us assume that there is some edge $y \to z \in G(\mathcal{R})$. Then, by the definition of the dependency graph, there must be some terms $f_z(\boldsymbol{t})$ and $f_y(\boldsymbol{s})$ with $f_y(\boldsymbol{s}) \in \boldsymbol{t}$ occurring in some set of facts $\mathcal{F}$ in some chase tree of a program $\langle \mathcal{R}, \mathcal{I} \rangle$. Let $B[\boldsymbol{x}, \boldsymbol{y}] \to \bigvee_{i=1}^n \exists \boldsymbol{v}_i . H_i[\boldsymbol{x}, \boldsymbol{v}_i]$ be the only rule in $\mathcal{R}$ containing z in some disjunct in the head. Then, $B[\boldsymbol{x}/\boldsymbol{t}] \subseteq \mathcal{F}$, and hence, $h(B[\boldsymbol{x}/\boldsymbol{t}])$ is contained in the result of the chase of $\langle \text{MSA}(\mathcal{R}), \mathcal{I}^\star_\mathcal{R} \rangle$. Since $B \to \bigwedge_{i=1}^n (H'_i \wedge \bigwedge_{x \in \boldsymbol{x}} \bigwedge_{v \in \boldsymbol{v}_i} Scc(x, v))\theta \in \text{MSA}(\mathcal{R})$, then $Scc(c_y, c_z)$ is also in the result of the chase of $\langle \text{MSA}(\mathcal{R}), \mathcal{I}^\star_\mathcal{R} \rangle$. In turn, this implies that $y \to z \in G_S(\mathcal{R})$. □

We proceed with the definition of tractable acyclicity, and thereafter establish the complexity of checking this condition and reasoning over such ontologies

Definition 21. *A TBox $\mathcal{T}$ is* k-tractable acyclic *(TA_k) if its summarizing graph is acyclic and the length of every braid in this graph is at most k.*

Theorem 22. *Deciding TA_k membership of a TBox $\mathcal{T}$ is* P*-complete.*

Proof. To verify membership, we propose a polynomial procedure to determine if $G_S(\mathcal{T})$ is acyclic and then compute the length of the longest braid in $G_S(\mathcal{T})$. Let $\mathcal{P} = \langle \mathcal{R}, \mathcal{I} \rangle$ be the program where $\mathcal{I}$ is the instance containing $E(c_y, c_z)$ for every $y \to z \in G_S(\mathcal{T})$, and $Neq(c_y, c_z)$ for every pair of nodes y and z in $G_S(\mathcal{T})$ with $y \neq z$; and $\mathcal{R} = \{ \to P(x,x), E(x,y) \to P(x,y), P(x,y) \wedge P(y,z) \to P(x,z), P(x,y) \wedge P(y,z) \to P(x,z), P(x,y) \wedge P(x,z) \wedge Neq(y,z) \wedge E(y,w) \wedge E(z,w) \to B(x,w), B(x,y) \wedge B(y,z) \to B(x,z) \}$. Then, there is a braid starting in y and ending in z in $G_S(\mathcal{R})$ if and only if $\mathcal{P} \models B(c_y, c_z)$. Thus, to determine the maximum length of a braid in $G_S(\mathcal{R})$, we simply have to look for the largest path over the binary predicate B in the result of the chase of $\mathcal{P}$. Moreover, $G_S(\mathcal{R})$ is acyclic if and only if $\mathcal{P}$ does not entail the query $\exists x . P(x,x)$. Note that, the program $\mathcal{P}$ can be constructed in polynomial time since the computation of $G_S(\mathcal{T})$ is tractable. Moreover, as the number of variables per rule in $\mathcal{R}$ is at most 4 and the maximum arity of a predicate is 2, the chase of such a program can be computed in polynomial time.

Hardness of the TA_k membership check can be readily ascertained via reduction from propositional horn entailment. □

Theorem 23. *Deciding BCQ entailment for TA_k ontologies $\langle \mathcal{T}, \mathcal{A} \rangle$ is co*NP*-complete provided the size of the query is fixed. Moreover, if $\mathcal{T}$ is a deterministic TBox, then it is* P*-complete.*

Proof. If $\mathcal{T}$ is TA_k, then $G_S(\mathcal{T})$ is acyclic and every braid in $G_S(\mathcal{T})$ is of length at most k. In turn, this implies that $G(\mathcal{T})$ is acyclic and every braid in $G(\mathcal{T})$ is of length at most k by Lemma 20. Since the TBox $\mathcal{T}$ satisfies restrictions (a), (b), (f), and (w), the theorem follows from Theorems 7 and 23. □

5 Evaluation

To assess the empirical generality of TA_k, we analyzed ontologies from MOWLCorp [22] and Oxford Ontology Library,[2] two large corpora of real-world OWL ontologies. These ontologies were transformed into the normal form defined in Figure 4 using standard normalization techniques [8]. After this step, we disregarded ontologies with nominals and number restrictions; and also ontologies without any axiom of type (9), as these are trivially TA_0. Since the MOWLCorp is rather large, we only considered ontologies in this corpus with up to 1,000 axioms of type (9). The final set contained 1,576 TBoxes from MOWLCorp and 225 TBoxes from the Oxford Ontology Library.

To determine TA_k membership, we first constructed the summarizing dependency graphs of the TBoxes. For this, we transformed axioms to rules using the mapping in Figure 4 and derived the programs described in Definition 19, over which we reasoned using the RDFox [24] datalog rule engine. Out of the obtained graphs, we found 974 (61.8%) acyclic ones from MOWLCorp and 171 (76%) from Oxford Library. Then, we determined TA_k membership of acyclic graphs by counting the length of their longest braid. We did this by constructing the program defined in Theorem 22, over which we reasoned using RDFox.

As our results show in Table 1, 78.3% of acyclic ontologies from MOWLCorp are TA_1, 90.8% are TA_2, 95.5% are TA_3, 98.8% are TA_4 and 99% are TA_5. In the Oxford Library, 51.4% of the acyclic ontologies are TA_1, 69.5% are TA_2, 81.2% are TA_3, 92.3% are TA_4, 97.6 % are TA_5 and 98.2 % are TA_6. There was only one ontology from the Oxford corpus (00477.owl), containing more than 150,000 rules of type (9), for which computing TA_k membership did not terminate.

Our acyclicity notion is theoretically equivalent to MSA and as general as MFA with respect to the evaluated ontologies: In our test set, there are no MFA ontologies which were not MSA. This validates the claims from [7, 10], where it was observed that MFA (the most general known acyclicity criterion for the skolem chase) is not empirically more general than MSA. Moreover, our results show that almost all acyclic ontologies are TA_k with a small k: TA_5 characterizes 97% of the ontologies in both corpora.

[2] http://www.cs.ox.ac.uk/isg/ontologies/

MOWLCorp	TA_1	TA_2	TA_3	TA_4	TA_5	TA_{22}	TA_{23}	TA_{25}	Total
	763	122	36	42	2	2	6	1	974

Oxford Onto. Library	TA_1	TA_2	TA_3	TA_4	TA_5	TA_6	TA_{11}	TA_{23}	Total
	88	31	20	19	9	1	1	1	170

Table 1. Histogram of TA_k on ontologies from MOWL and Oxford corpora, where for TA_k we only count ontologies that do not also belong to TA_j for all $j < k$.

6 Conclusions and Future Work

To the best of our knowledge, this is the first systematic study of tractability of CQ answering with disjunctive existential rules. An important application is tractable query answering over OWL ontologies, a task which in general is known to be intractable [25]. We have shown that our restrictions do indeed apply, for small bounds of the related parameters, to many practical ontologies.

Our work therefore suggests a new approach to efficient reasoning that might be applicable to many realistic ontologies, and which might be natural to implement in existing reasoners such as HermiT [23], which use chase-like procedures already. The extension of our work with more general conditions for restricted chase termination, which was recently shown to work well with many OWL ontologies [7], may further help to extend the applicability of this approach.

Acknowledgements Supported by the DFG within the cfaed Cluster of Excellence, CRC 912 (HAEC), and Emmy Noether grant KR 4381/1-1.

References

1. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)
2. Bienvenu, M., Hansen, P., Lutz, C., Wolter, F.: First order-rewritability and containment of conjunctive queries in Horn description logics. In: Proc. 25th Int. Joint Conf. on Artificial Intelligence (IJCAI'16). pp. 965–971. IJCAI/AAAI Press (2016)
3. Bourhis, P., Morak, M., Pieris, A.: The impact of disjunction on query answering under guarded-based existential rules. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. pp. 796–802. IJCAI/AAAI (2013)
4. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/-. In: Proc. 4th Int. Conf. on Web Reasoning and Rule Systems. LNCS, vol. 6333, pp. 1–17. Springer (2010)
5. 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)
6. 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. Automated Reasoning (JAR) 39(3), 385–429 (2007)

7. Carral, D., Dragoste, I., Krötzsch, M.: Restricted chase (non)termination for existential rules with disjunctions. In: IJCAI 2017, Melbourne, Australia, August 2017 (2017), to appear
8. Carral, D., Feier, C., Cuenca Grau, B., Hitzler, P., Horrocks, I.: $\mathcal{EL}$-ifying ontologies. In: IJCAR. LNCS, vol. 8562, pp. 464–479. Springer (2014)
9. Carral, D., Feier, C., Hitzler, P.: A practical acyclicity notion for query answering over Horn-SRIQ ontologies. In: The 15th International Semantic Web Conference, Kobe, Japan, 2016, Proceedings, Part I. LNCS, vol. 9981, pp. 70–85 (2016)
10. Cuenca Grau, B., 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. JAIR 47, 741–808 (2013)
11. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys 33(3), 374–425 (2001)
12. Feier, C., Carral, D., Stefanoni, G., Grau, B.C., Horrocks, I.: The combined approach to query answering beyond the OWL 2 profiles. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 2971–2977. AAAI Press (2015), http://ijcai.org/Abstract/15/420
13. Gottlob, G., Pichler, R., Wei, F.: Bounded treewidth as a key to tractability of knowledge representation and reasoning. Artif. Intell. 174(1), 105–132 (2010)
14. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
15. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, United Kingdom, June 2-5, 2006. pp. 57–67. AAAI Press (2006)
16. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany (2006)
17. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in dl-lite. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010. AAAI Press (2010), http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1282
18. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI. pp. 2656–2661 (2011)
19. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proceedings 22nd IJCAI. pp. 963–968. AAAI Press (2011)
20. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: ISWC. pp. 314–330 (2013)
21. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June, 2009, USA. pp. 13–22. ACM (2009)
22. Matentzoglu, N., Bail, S., Parsia, B.: A snapshot of the OWL Web. In: Proc. 12th Int. Semantic Web Conf. (ISWC'13). LNCS, vol. 8218, pp. 331–346. Springer (2013)
23. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research (JAIR) 36(1), 165–228 (2009)
24. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of Datalog programs in centralised, main-memory RDF systems. In: AAAI (2014)
25. Stefanoni, G., Motik, B., Krötzsch, M., Rudolph, S.: The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. J. of Art. Int. Research 51, 645–705 (2014)