Efficient Answer Enumeration in Description Logics with Functional Roles

Carsten Lutz, Marcin Przybyłko

Institute of Computer Science, Leipzig University, Germany
{clu,przybyl}@informatik.uni-leipzig.de

Abstract

We study the enumeration of answers to ontology-mediated queries when the ontology is formulated in a description logic that supports functional roles and the query is a CQ. In particular, we show that enumeration is possible with linear preprocessing and constant delay when a certain extension of the CQ (pertaining to functional roles) is acyclic and free-connex acyclic. This holds both for complete answers and for partial answers. We provide matching lower bounds for the case where the query is self-join free.

1 Introduction

In ontology-mediated querying, a query is combined with an ontology to inject domain knowledge and to facilitate access to incomplete and heterogeneous data (Bienvenu et al. 2014; Calvanese et al. 2009; Calì, Gottlob, and Lukasiewicz 2012). Intense research has been carried out on the complexity of ontology-mediated querying, often focussing on conjunctive queries (CQs) and on description logics and existential rules as ontology languages. Most of the existing studies have concentrated on the basic problem of single-testing which means to decide, given an ontology-mediated query (OMQ) Q, a database $\mathcal{D}$, and a candidate answer $\bar{a}$, whether $\bar{a}$ is indeed an answer to Q on $\mathcal{D}$. From the viewpoint of many practical applications, however, the assumption that a candidate answer is provided is hardly realistic and it seems much more relevant to enumerate, given an OMQ Q and a database $\mathcal{D}$, all answers to Q on $\mathcal{D}$.

The investigation of answer enumeration for OMQs has recently been initiated in (Lutz and Przybylko 2022b) which also introduces useful new notions of minimal partial answers; such answers may contain wildcards to represent objects that are known to exist, but whose exact identity is unknown. If, for example, the ontology stipulates that

$$\begin{aligned} \mathsf{Researcher} &\sqsubseteq \exists \mathsf{worksFor}.\mathsf{University} \\ \mathsf{Unversity} &\sqsubseteq \mathsf{Academia} \end{aligned}$$

and the database $\mathcal{D}$ is $\{\mathsf{Researcher}(\mathsf{mary})\}$, then there are no complete answers to the CQ

$$q(x,y) = \mathsf{worksFor}(x,y) \wedge \mathsf{Academia}(y),$$

but $(\mathsf{mary}, *)$ is a minimal partial answer that conveys information which is otherwise lost. The ontologies in (Lutz and Przybylko 2022b) are sets of guarded existential rules which generalize well-known description logics such as $\mathcal{EL}$ and $\mathcal{ELIH}$. An important feature of description logics that is not captured by guarded rules are functionality assertions on roles, which make it possible to declare that some binary relation symbols must be interpreted as partial functions.

The purpose of this paper is to study the enumeration of answers to OMQs that combine a CQ with an ontology formulated in a description logic with functional roles, in particular $\mathcal{ELIHF}$ and its fragments. We consider both the traditional complete answers and two versions of minimal partial answers that differ in which kind of wildcards are admitted. In one version, there is only a single wildcard symbol '$*$' while in the other version, multiple wildcards '$*_1$', '$*_2$', etc are admitted and multiple occurrences of the same wildcard represent the same unknown constant.

We study enumeration algorithms with a preprocessing phase that takes time linear in the size of $\mathcal{D}$ and with constant delay, that is, in the enumeration phase the delay between two answers must be independent of $\mathcal{D}$. Note that we assume the OMQ Q to be fixed and of constant size, as in data complexity. If such an algorithm exists, then enumeration belongs to the complexity class $\mathsf{DelayC_{lin}}$. If in addition the algorithm writes in the enumeration phase only a constant amount of memory, then it belons to the class CD$\circ$Lin. Enumeration algorithms with these properties have been studied intensely, see (Berkholz, Gerhardt, and Schweikardt 2020; Segoufin 2015) for an overview. It is not known whether $\mathsf{DelayC_{lin}}$ and CD$\circ$Lin coincide (Kazana 2013).

It is an important result for CQs q without ontologies that enumeration is possible in CD$\circ$Lin if q is acyclic and free-connex acyclic (Bagan, Durand, and Grandjean 2007), the latter meaning that q is acyclic after adding an atom that covers all answer variables. If these conditions are not met and q is self-join free (i.e., no relation symbol occurs more than once), then enumeration is not possible in $\mathsf{DelayC_{lin}}$ unless certain algorithmic assumptions fail that pertain to the triangle conjecture, the hyperclique conjecture, and Boolean matrix multiplication (Bagan, Durand, and Grandjean 2007; Brault-Baron 2013). In the presence of functional dependencies, which in their unary version are identical to functionality assertions in description logic, the characteriza-

tion changes: enumeration is in CD∘Lin if a certain extension q^+ of q guided by the functional dependencies is acyclic and free-connex acyclic (Carmeli and Kröll 2020). Notably, adding functional dependencies may result in aditional queries to become enumerable in CD∘Lin.

The main results presented in this paper are as follows. We consider OMQs Q where the ontology $\mathcal{O}$ is formulated in the description logic $\mathcal{ELIHF}$ and the query q is a CQ, and show that enumerating answers to Q is possible in CD∘Lin if q^+ is acyclic and free-connex acyclic. For complete answers, this is achieved by using carefully defined universal models and showing that they can be constructed in linear time via an encoding as a propositional Horn formula. For minimal partial answers (with single or multiple wildcards), we additionally make use of an enumeration algorithm that was given in (Lutz and Przybylko 2022b). Here, we only attain enumeration in $\mathsf{DelayC_{lin}}$.

We also prove corresponding lower bounds for self-join free queries, paralleling those in (Bagan, Durand, and Grandjean 2007; Brault-Baron 2013) and (Carmeli and Kröll 2020). They concern only ontologies formulated in the fragment $\mathcal{ELIF}$ of $\mathcal{ELIHF}$ that disallows role inclusions. The reason is that lower bounds for OMQs with role inclusions entail a characterization of enumerability in CD∘Lin for CQs with self-joins, a major open problem even without ontologies. The lower bounds apply to complete and (both versions of) minimal partial answers and are conditional on the same algorithmic assumptions as in the case without ontologies. Our constructions and correctness proofs are more challenging than the existing ones in the literature since, unlike in the upper bounds, we cannot directly use the query q^+. This is because the transition from q to q^+ changes the signature, extending the arity of relation symbols beyond two, and it is unclear how this can be reflected in the ontology.

We also study the combined complexity of single-testing for minimal partial answers, concentrating on the description logics $\mathcal{EL}$ and $\mathcal{ELH}$ which bear special importance because single-testing complete answers to OMQs based on acyclic CQs and ontologies formulated in these languages is in PTIME. It turns out that this property extends to the single wildcard version of minimal partial answers, but not to the multi-wildcard version. For unrestricted CQs, the complexity raises from NP-complete for complete answers to DP-complete for both versions of minimal partial answers.

Detailed proofs are in the appendix of the extended version, made available at (Lutz and Przybylko 2022a).

2 Preliminaries

Let $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{K}$ be countably infinite sets of *concept names*, *role names*, and *constants*. A *role* R is a role name $r \in \mathbf{R}$ or an *inverse role* r^- with r a role name. If $R = r^-$, then $R^- = r$. An *$\mathcal{ELI}$-concept* is built according to the rule $C, D ::= A \mid C \sqcap D \mid \exists R.C$ where A ranges over concept names and R over roles. An *$\mathcal{ELIHF}$-ontology* is a finite set of *concept inclusions (CIs)* $C \sqsubseteq D$ *role inclusions (RIs)* $R \sqsubseteq S$, and *functionality assertions* $\mathsf{func}(R)$ where (here and in what follows) C, D range over $\mathcal{ELI}$ concepts and R, S over roles. An *$\mathcal{ELIF}$-ontology* is an $\mathcal{ELIHF}$-ontology that does not use RIs.

A *database* is a finite set of *facts* of the form $A(c)$ or $r(c, c')$ where A is a concept name or $\top$, r is a role name, and $c, c' \in \mathbf{K}$. We use $\mathsf{adom}(\mathcal{D})$ to denote the set of constants used in database $\mathcal{D}$, also called its *active domain*. We may write $r^-(a, b) \in \mathcal{D}$ to mean $r(b, a) \in \mathcal{D}$.

A *signature* is a set of concept and role names, uniformly referred to as *relation symbols*. For a syntactic object O such as a concept or an ontology, we use $\mathsf{sig}(O)$ to denote the set of relation symbols used in it and $||O||$ to denote its size, that is, the number of symbols needed to write it as a word using a suitable encoding.

The semantics is given in terms of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set called the *domain* and $\cdot^{\mathcal{I}}$ is the interpretation function, see (Baader et al. 2017) for details. We take the liberty to identify interpretations with non-empty and potentially infinite databases. The interpretation function $\cdot^{\mathcal{I}}$ is then defined as $A^{\mathcal{I}} = \{c \mid A(c) \in \mathcal{I}\}$ for concept names A and $r^{\mathcal{I}} = \{(c, c') \mid r(c, c') \in \mathcal{I}\}$ for role names r. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, a fact $A(c)$ if $c \in A^{\mathcal{I}}$, and a fact $r(c, c')$ if $(c, c') \in r^{\mathcal{I}}$. We thus make the standard names assumption, that is, we interpret constants as themselves. An interpretation $\mathcal{I}$ is a *model* of an ontology (resp. database) if it satisfies all inclusions and assertions (resp. facts) in it.

A database $\mathcal{D}$ is *satisfiable* w.r.t. an ontology $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{D}$. Note that functionality assertions in an ontology $\mathcal{O}$ can result in databases that are unsatisfiable w.r.t. $\mathcal{O}$. We write $\mathcal{O} \models \mathsf{func}(R)$ if every model of $\mathcal{O}$ satisfies the functionality assertion $\mathsf{func}(R)$. In $\mathcal{ELIHF}$, this is decidable and EXPTIME-complete; see appendix.

Queries. A *conjunctive query (CQ)* is of the form $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where $\bar{x}$ and $\bar{y}$ are tuples of variables and $\varphi(\bar{x}, \bar{y})$ is a conjunction of *concept atoms* $A(x)$ and *role atoms* $r(x, y)$, with A a concept name, r a role name, and x, y variables from $\bar{x} \cup \bar{y}$. We call the variables in $\bar{x}$ the *answer variables* of q, and use $\mathsf{var}(q)$ to denote $\bar{x} \cup \bar{y}$. We may write $\alpha \in q$ to indicate that α is an atom in q. For $V \subseteq \mathsf{var}(q)$, we use $q|_V$ to denote the restriction of q to the atoms that use only variables in V. A *homomorphism* from q to an interpretation $\mathcal{I}$ is a function $h : \mathsf{var}(q) \to \Delta^{\mathcal{I}}$ such that $A(x) \in q$ implies $A(h(x)) \in \mathcal{I}$ and $r(x, y) \in q$ implies $r(h(x), h(y)) \in \mathcal{I}$. A tuple $\bar{d} \in (\Delta^{\mathcal{I}})^{|\bar{x}|}$, where $|\bar{x}|$ denotes the length of the tuple $\bar{x}$, is an *answer* to q on interpretation $\mathcal{I}$ if there is a homomorphism h from q to $\mathcal{I}$ with $h(\bar{x}) = \bar{d}$.

Every CQ q is associated with a canonical database $\mathcal{D}_q$ obtained from q by viewing variables as constants and atoms as facts. We associate every database, and via $\mathcal{D}_q$ also every CQ q, with an undirected graph $G_{\mathcal{D}} = (\mathsf{adom}(\mathcal{D}), \{\{a, b\} \mid R(a, b) \in \mathcal{D}$ for some role $R\}$. It is thus clear what we mean by a *path* $c_0, \ldots, c_k$ in a database and a path $x_0, \ldots, x_k$ in a CQ. A CQ q is *self-join free* if every relation symbol occurs in at most one atom in q.

Ontology-Mediated Queries. An *ontology-mediated query (OMQ)* is a pair $Q = (\mathcal{O}, \Sigma, q)$ with $\mathcal{O}$ an ontology, $\Sigma \subseteq \mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q)$ a finite signature called the *data schema*, and q a query. We write $Q(\bar{x})$ to indicate that the answer variables of q are $\bar{x}$. The signature Σ expresses the promise that Q is only evaluated on Σ-databases. Let $\mathcal{D}$ be

such a database. A tuple $\bar{a} \in \mathsf{adom}(\mathcal{D})^{|\bar{x}|}$ is an *answer* to $Q(\bar{x})$ on $\mathcal{D}$, written $\mathcal{D} \models Q(\bar{a})$, if $\mathcal{I} \models q(\bar{a})$ for all models $\mathcal{I}$ of $\mathcal{O}$ and $\mathcal{D}$. We might alternatively write $\mathcal{D}, \mathcal{O} \models q(\bar{a})$. With $Q(\mathcal{D})$ we denote the set of all answers to Q on $\mathcal{D}$. An OMQ $Q = (\mathcal{O}, \Sigma, q)$ is *empty* if $Q(\mathcal{D}) = \emptyset$ for every Σ-database $\mathcal{D}$ that is satisfiable w.r.t. $\mathcal{O}$.

We may assume w.l.o.g. that $\mathcal{ELIHF}$ ontologies used in OMQs are in *normal form*, that is, all CIs in it are of one of the following forms:

$$\top \sqsubseteq A, \quad A_1 \sqcap A_2 \sqsubseteq A, \quad A_1 \sqsubseteq \exists R.A_2, \quad \exists R.A_1 \sqsubseteq A_2$$

where A_1, A_2, A range over concept names. Every $\mathcal{ELIHF}$-ontology $\mathcal{O}$ can be converted into this form in linear time without affecting the answers to OMQs (Baader et al. 2017).

With $(\mathcal{L}, \mathcal{Q})$, we denote the *OMQ language* that contains all OMQs Q in which $\mathcal{O}$ is formulated in DL $\mathcal{L}$ and q in query language $\mathcal{Q}$, such as in $(\mathcal{ELIHF}, \text{CQ})$.

Partial Answers. Fix a wildcard symbol '$*$' that is not in $\mathbf{K}$. A *wildcard tuple* for a database $\mathcal{D}$ takes the form $(c_1, \ldots, c_n) \in (\mathsf{adom}(\mathcal{D}) \cup \{*\})^n$, $n \geq 0$. For wildcard tuples $\bar{c} = (c_1, \ldots, c_n)$ and $\bar{c}' = (c'_1, \ldots, c'_n)$, we write $\bar{c} \preceq \bar{c}'$ if $c'_i \in \{c_i, *\}$ for $1 \leq i \leq n$. Moreover, $\bar{c} \prec \bar{c}'$ if $\bar{c} \preceq \bar{c}'$ and $\bar{c} \neq \bar{c}'$. For example, $(a, b) \prec (a, *)$ and $(a, *) \prec (*, *)$ while $(a, *)$ and $(*, b)$ are incomparable w.r.t. '$\prec$'. Informally, $\bar{c} \prec \bar{c}'$ expresses that tuple $\bar{c}$ is preferred over tuple $\bar{c}'$ as it carries more information.

A *partial answer* to OMQ $Q(\bar{x}) = (\mathcal{O}, \Sigma, q)$ on an $\mathbf{S}$-database $\mathcal{D}$ is a wildcard tuple $\bar{c}$ for $\mathcal{D}$ of length $|\bar{x}|$ such that for each model $\mathcal{I}$ of $\mathcal{D}$ and $\mathcal{O}$, there is a $\bar{c}' \in q(\mathcal{I})$ such that $\bar{c}' \preceq \bar{c}$. Note that some positions in $\bar{c}'$ may contain constants from $\mathsf{adom}(\mathcal{I}) \setminus \mathsf{adom}(\mathcal{D})$, and that the corresponding position in $\bar{c}$ must then have a wildcard. A partial answer $\bar{c}$ to Q on a Σ-database $\mathcal{D}$ is a *minimal partial answer* if there is no partial answer $\bar{c}'$ to Q on $\mathcal{D}$ with $\bar{c}' \prec \bar{c}$. We use $Q(\mathcal{D})^*$ to denote the set of all minimal partial answers to Q on $\mathcal{D}$. An example is provided in the introduction. Note that $Q(\mathcal{D}) \subseteq Q(\mathcal{D})^*$. To distinguish them from partial answers, we also refer to the answers in $Q(\mathcal{D})$ as *complete* answers.

We also define a second version of minimal partial answers where multiple wildcards are admitted, from a countably infinite set $\mathcal{W} = \{*_1, *_2, \ldots\}$ disjoint from $\mathbf{K}$. Multiple occurrences of the same wildcard then represent the same unknown constant while different wildcards may or may not represent different constants. We use $Q(\mathcal{D})^{\mathcal{W}}$ to denote the set of minimal partial answers with multiple wildcards. A precise definition is provided in the appendix, here we only give an example.

Example 1. *Let $Q(x, y, z) = (\mathcal{O}, \Sigma, q)$ where $\mathcal{O}$ contains*

$$\mathsf{Company} \sqsubseteq \exists \mathsf{hasEmployee}.\mathsf{Person}$$
$$\mathsf{TechCompany} \sqsubseteq \mathsf{Company}, \quad \mathsf{CarCompany} \sqsubseteq \mathsf{Company},$$
$$\mathsf{TechFactory} \sqsubseteq \exists \mathsf{hasOwner}.\mathsf{TechCompany}$$
$$\mathsf{CarFactory} \sqsubseteq \exists \mathsf{hasOwner}.\mathsf{CarCompany}$$

and $\mathsf{func}(\mathsf{hasOwner})$*, Σ contains all symbols from $\mathcal{O}$, and*

$$q(x, y, z) = \mathsf{Person}(x) \wedge \mathsf{hasEmployee}(y, x) \wedge \mathsf{TechCompany}(y) \wedge \mathsf{hasEmployee}(z, x) \wedge \mathsf{CarCompany}(z).$$

Further consider the database $\mathcal{D}$ with facts

$$\mathsf{CarFactory}(\mathsf{gigafactory1}), \mathsf{TechFactory}(\mathsf{gigafactory1}).$$

Then $Q^{\mathcal{W}}(\mathcal{D}) = \{(*_1, *_2, *_2)\}$. If we extend $\mathcal{D}$ with $\mathsf{hasOwner}(\mathsf{gigafactory1}, \mathsf{tesla})$, then this changes to $Q^{\mathcal{W}}(\mathcal{D}) = \{(*_1, \mathsf{tesla}, \mathsf{tesla})\}$.

Enumeration. We are interested in enumerating the complete and minimal partial answers to a given OMQ $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{L}, \mathcal{Q})$ on a given Σ-database D. An *enumeration algorithm* has a *preprocessing phase* where it may produce data structures, but no output. In the subsequent *enumeration phase*, it enumerates all tuples from $Q(D)$, without repetition. Answer enumeration for an OMQ language $(\mathcal{L}, \mathcal{Q})$ is possible *with linear preprocessing and constant delay*, or *in* $\mathsf{DelayC_{lin}}$, if there is an enumeration algorithm for $(\mathcal{L}, \mathcal{Q})$ in which preprocessing takes time $f(||Q||) \cdot O(||D||)$, f a computable function, while the delay between the output of two consecutive answers depends only on $||Q||$, but not on $||D||$. Enumeration in $\mathsf{CD{\circ}Lin}$ is defined likewise, except that the total amount of additional memory used in the enumeration phase must be independent of $||D||$.

The above definition only becomes precise when we fix a concrete machine model. We use RAMs under a uniform cost measure (Cook and Reckhow 1973), see (Grandjean 1996) for a formalization. A RAM has a one-way read-only input tape, a write-only output tape, and an unbounded number of registers that store non-negative integers of $O(\log n)$ bits, n the input size. In this model, which is standard in the $\mathsf{DelayC_{lin}}$ context, sorting is possible in linear time and we can access in constant time lookup tables indexed by constants from $\mathsf{adom}(D)$ (Grandjean 1996).

We also consider *single-testing* which means to decide, given an OMQ $Q(\bar{x}) = (\mathcal{O}, \Sigma, q)$, a Σ-database D, and an answer candidate $\bar{c} \in \mathsf{adom}(D)^{|\bar{x}|}$, whether $\bar{c} \in Q(D)$.

Acyclic CQs. Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be a CQ. A *join tree* for $q(\bar{x})$ is an undirected tree $T = (V, E)$ where V is the set of atoms in φ and for each $x \in \mathsf{var}(q)$, the set $\{\alpha \in V \mid x \text{ occurs in } \alpha\}$ is a connected subtree of T. A CQ $q(\bar{x})$ is *acyclic* if it has a join tree. If q contains only unary and binary relations (which shall not always be the case), then q being acyclic is equivalent to $G_{\mathcal{D}_q}$ being a tree, potentially with multi-edges and self-loops. A CQ $q(\bar{x})$ is *free-connex acyclic* if adding a *head atom* $H(\bar{x})$ that 'guards' the answer variables, where H is a fresh relation symbol of arity $|\bar{x}|$, results in an acyclic CQ. Acyclicity and free-connex acyclicity are independent properties, that is, neither of them implies the other.

3 Upper Bounds

We identify cases that admit answer enumeration in $\mathsf{CD{\circ}Lin}$ and $\mathsf{DelayC_{lin}}$, considering both complete answers and minimal partial answers. From now on, we also use relation symbols of arity exceeding two, identifying concept names and role names with relations symbols of arity one and two.

We start with some preliminaries. Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \text{CQ})$. Fix a linear order on the variables in q. A *path* $y_0, \ldots, y_k$ in q is *functional* if for

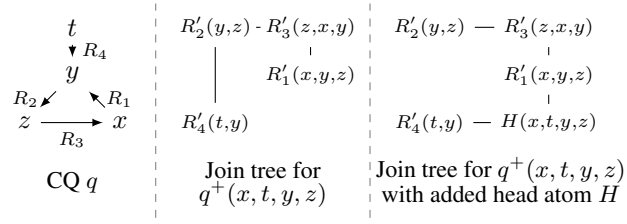

Figure 1: Illustration of Example 2

$0 \leq i < k$, there is an atom $R(y_i, y_{i+1}) \in q$ such that $\mathcal{O} \models \mathsf{func}(R)$. For a tuple $\bar{x}$ of variables, we use $\bar{x}^+$ to denote the tuple $\bar{x}\bar{y}$ where $\bar{y}$ consists of all variables y, in the fixed order, that are reachable from a variable in $\bar{x}$ on a functional path in q and that are not part of $\bar{x}$.

The *FA-extension of* q is the CQ $q^+(\bar{x}^+)$ that contains an atom $R'(\bar{y}^+)$ for every atom $R(\bar{y})$ in q, where R' is a fresh relation symbol of arity $|\bar{y}^+|$. Note that q and q^+ are over different signatures. Moreover, q^+ may contain symbols of high arity and is self-join free. We also consider the CQ $q^+(\bar{x})$ with a non-extended set of answer variables. Our FA-extensions are a variation of the notion of FD-extension used in (Carmeli and Kröll 2020).

Example 2. *Let* $\mathcal{O}$=$\{\mathsf{func}(R_2^-), \mathsf{func}(R_3^-), \mathsf{func}(R_4)\}$ *and* $q(x,t)$=$R_1(x,y) \wedge R_2(y,z) \wedge R_3(z,x) \wedge R_4(t,y)$. *Fix the variable order* $x < y < z < t$. *Then the CQ* $q^+((x,t)^+)$ *is*

$$q^+(x,t,y,z) = R'_1(x,y,z), R'_2(y,z), R'_3(z,x,y), R'_4(t,y),$$

which is acyclic and free-connex acyclic while q *is neither. See Figure 1 for a graphical representation.*

In contrast, the version of q^+ *that has a non-extended set of answer variables* $\{x,t\}$ *(but is otherwise identical) is acyclic, but not free-connex acyclic.*

We state our upper bound for complete answers.

Theorem 1. *Let* $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \mathit{CQ})$ *such that* $q^+(\bar{x}^+)$ *is acyclic and free-connex acyclic. Then the complete answers to* Q *can be enumerated in* CD∘Lin.

The proof of Theorem 1 relies on the following.

Proposition 1. *Given an OMQ* $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \mathit{CQ})$ *and a* Σ*-database* $\mathcal{D}$ *that is satisfiable w.r.t.* $\mathcal{O}$, *one can compute in time* $2^{\mathsf{poly}(||Q||)} \cdot O(||\mathcal{D}||)$ *a database* $\mathcal{U}_{\mathcal{D},Q} \supseteq \mathcal{D}$ *that satisfies all functionality assertions in* $\mathcal{O}$ *and such that* $Q(\mathcal{D}) = q(\mathcal{U}_{\mathcal{D},Q}) \cap \mathsf{adom}(\mathcal{D})^{|\bar{x}|}$.

The database $\mathcal{U}_{\mathcal{D},Q}$ from Proposition 1 should be thought of as a univeral model for the ontology $\mathcal{O}$ and database $\mathcal{D}$ that is tailored specifically towards the query q. Such 'query-directed' universal models originate from (Bienvenu et al. 2013). We compute $\mathcal{U}_{\mathcal{D},Q}$ in time $2^{\mathsf{poly}(||Q||)} \cdot O(||\mathcal{D}||)$ by constructing a suitable propositional Horn logic formula θ, computing a minimal model for θ in linear time (Dowling and Gallier 1984), and then reading off $\mathcal{U}_{\mathcal{D},Q}$ from that model. Details are provided in the appendix.

To prove Theorem 1, let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \mathrm{CQ})$ with $q^+(\bar{x}^+)$ acyclic and free-connex, and let $\mathcal{D}$ be a Σ-database. We first replace q by the CQ q_0 that is obtained from q by choosing a fresh concept name D and adding $D(x)$ for every answer variable x. We also replace $\mathcal{D}$ by the database $\mathcal{D}_0 = \mathcal{U}_{\mathcal{D},Q}$ from Proposition 1 extended with $D(c)$ for all $c \in \mathsf{adom}(\mathcal{D})$. Clearly, $Q(\mathcal{D}) = q_0(\mathcal{D}_0)$ and thus it suffices to enumerate the latter.

We next replace $q_0(\bar{x})$ by $q_0^+(\bar{x}^+)$ and $\mathcal{D}_0$ by a database $\mathcal{D}_0^+$ that reflects the change in signature which comes with the transition from q_0 to q_0^+. More precisely if atom $R(\bar{y})$ is replaced with $R'(\bar{y}^+)$ in the construction of q_0^+ and h is a homomorphism from $q_0|_{\bar{y}^+}$ to $\mathcal{D}_0$, then $\mathcal{D}_0^+$ contains the fact $R'(h(\bar{y}^+))$. The following implies that $q_0(\mathcal{D}_0)$ is the projection of $q_0^+(\mathcal{D}_0^+)$ to the first $|\bar{x}|$ components.

Lemma 1. *Every homomorphism from* q_0 *to* $\mathcal{D}_0$ *is also a homomorphism from* q_0^+ *to* $\mathcal{D}_0^+$ *and vice versa. Moreover,* $\mathcal{D}_0^+$ *can be constructed in time linear in* $||\mathcal{D}||$.

To enumerate $q_0(\mathcal{D}_0)$, we may thus enumerate $q^+(\mathcal{D}_0^+)$ and project to the first $|\bar{x}|$ components. The former can be done in CD∘Lin: since $q^+(\bar{x}^+)$ is acyclic and free-connex, so is $q_0^+(\bar{x}^+)$, and thus we may apply the CD∘Lin enumeration procedure from (Bagan, Durand, and Grandjean 2007; Berkholz, Gerhardt, and Schweikardt 2020). Clearly, projection can be implemented in constant time. To argue that the resulting algorithm produces no duplicates, it remains to observe that the answers to $q_0(\mathcal{D}_0)$ and to $q^+(\mathcal{D}_0^+)$ are in a one-to-one correspondence, that is, every $\bar{c} \in q_0(\mathcal{D}_0)$ extends in a unique way to a $\bar{c}' \in q^+(\mathcal{D}_0^+)$. This, however, is an immediate consequence of Lemma 1, the definition of q_0^+ and the fact that $\mathcal{D}_0$ saties all functionality assertions in $\mathcal{O}$.

We now turn to minimal partial answers. Here, we cannot expect a result as general as Theorem 1, a counterexample is presented in Section 4. We thus resort to the stronger condition that $q^+(\bar{x})$ rather than $q^+(\bar{x}^+)$ is acyclic and free-connex acyclic. The difference between the two conditions is related to the interplay of answer variables and functional roles. In particular, $q^+(\bar{x})$ and $q^+(\bar{x}^+)$ are identical for OMQs $Q = (\mathcal{O}, \Sigma, q)$ such that answer variables have no functional edges to quantified variables, that is, for every atom $R(x,y)$ in q, if $\mathcal{O} \models \mathsf{func}(R)$ and x is an answer variable, then y is also an answer variable.

Theorem 2. *Let* $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \mathit{CQ})$ *such that* $q^+(\bar{x})$ *is acyclic and free-connex acyclic. Then the minimal partial answers to* Q *can be enumerated in* $\mathsf{DelayC_{lin}}$, *both with multi-wildcards and with a single wildcard.*

To prove Theorem 2, we make use of a recent result regarding OMQs in which the ontologies are sets of guarded existential rules. We refer to the class of such ontologies as $\mathbb{G}$. It was shown in (Lutz and Przybylko 2022b) that minimal partial answers to OMQs $Q = (\mathcal{O}, \Sigma, q) \in (\mathbb{G}, \mathrm{CQ})$ can be enumerated in $\mathsf{DelayC_{lin}}$ if q is acyclic and free-connex acyclic, both with multi-wildcards and with a single wildcard. The enumeration algorithms presented in (Lutz and Przybylko 2022b) are non-trivial and we use them as a blackbox. To achieve this, we need a slightly more 'low-level' formulation of the results from (Lutz and Przybylko 2022b). In what follows, we restrict our attention to minimal partial answers with a single wildcard. The multi-wildcard case is analogous, details are in the appendix.

Fix a countably infinite set $\mathbf{N}$ of *nulls* that is disjoint from $\mathbf{K}$ and does not contain the wildcard symbol '$*$'. In what follows, we assume that databases may use nulls in place of constants. Let $\mathcal{D}$ be a database and $q(\bar{x})$ a CQ. For an answer $\bar{a} \in q(\mathcal{D})$, we use $\bar{a}^*_{\mathbf{N}}$ to denote the unique wildcard tuple for $\mathcal{D}$ obtained from $\bar{a}$ by replacing all nulls with '$*$'. We call $\bar{a}^*_{\mathbf{N}}$ a *partial answer* to q on $\mathcal{D}$ and say that it is *minimal* if there is no $\bar{b} \in q(\mathcal{D})$ with $\bar{b}^*_{\mathbf{N}} \prec \bar{a}^*_{\mathbf{N}}$. With $q(\mathcal{D})^*_{\mathbf{N}}$, we denote the set of minimal partial answers to q on $\mathcal{D}$.

A database $\mathcal{E}$ is *chase-like* if there are databases $\mathcal{D}_1, \dots, \mathcal{D}_n$ such that

1. $\mathcal{E} = \mathcal{D}_1 \cup \dots \cup \mathcal{D}_n$,
2. $\mathcal{D}_i$ contains exactly one fact that uses no nulls, and that fact contains all constants in $\mathsf{adom}(\mathcal{D}_i) \setminus \mathbf{N}$,
3. $\mathsf{adom}(\mathcal{D}_i) \cap \mathsf{adom}(\mathcal{D}_j) \cap \mathbf{N} = \emptyset$ for $1 \leq i < j \leq n$.

We call $\mathcal{D}_1, \dots, \mathcal{D}_n$ a *witness* for $\mathcal{E}$ being chase-like. The term 'chase-like' refers to the chase, a well-known procedure for constructing universal models (Johnson and Klug 1982). The query-directed universal models $\mathcal{U}_{\mathcal{D},Q}$ from Proposition 1 are chase-like when the elements of $N = \mathsf{adom}(\mathcal{U}_{\mathcal{D},Q}) \setminus \mathsf{adom}(\mathcal{D})$ are viewed as nulls. A witness $\mathcal{D}_1, \dots, \mathcal{D}_n$ is obtained by removing from $\mathcal{U}_{\mathcal{D},Q}$ all atoms $r(a, b)$ with $a, b \in \mathsf{adom}(\mathcal{D})$ and taking the resulting maximally connected components. The domain sizes $|\mathsf{adom}(\mathcal{D}'_i)|$ then only depend on Q, but not on $\mathcal{D}$. The following is Proposition E.1 in (Lutz and Przybylko 2022c).

Theorem 3. *For every CQ $q(\bar{x})$ that is acyclic and free-connex ayclic, enumerating the answers $q(\mathcal{D})^*_{\mathbf{N}}$ is in $\mathsf{DelayC_{lin}}$ for databases $\mathcal{D}$ and sets of nulls $N \subseteq \mathsf{adom}(\mathcal{D})$ such that $\mathcal{D}$ is chase-like with witness $\mathcal{D}_1, \dots, \mathcal{D}_n$ where $|\mathsf{adom}(\mathcal{D}_i)|$ does not depend on $\mathcal{D}$ for $1 \leq i \leq n$.*

The strategy for proving Theorem 2 is now similar to the case of complete answers. Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIHF}, \text{CQ})$ with $q^+(\bar{x})$ acyclic and free-connex acyclic, and let $\mathcal{D}$ be a Σ-database. It is shown in the appendix that the query-directed universal model $\mathcal{U}_{\mathcal{D},Q}$ is also universal for partial answers with a single wildcard in the sense that $Q(\mathcal{D})^* = q(\mathcal{U}_{\mathcal{D},Q})^*_{\mathbf{N}}$. We thus first replace $\mathcal{D}$ with $\mathcal{U}_{\mathcal{D},Q}$, aiming to enumerate $Q(\mathcal{D})^* = q(\mathcal{U}_{\mathcal{D},Q})^*_{\mathbf{N}}$. We next replace $q(\bar{x})$ with $q^+(\bar{x})$ and $\mathcal{D}_0 = \mathcal{U}_{\mathcal{D},Q}$ with $\mathcal{D}^+_0$. It follows from Lemma 1 that $q(\mathcal{D}_0)^*_{\mathbf{N}} = q^+(\mathcal{D}^+_0)^*_{\mathbf{N}}$. Note that no projection is needed since q^+ has answer variables $\bar{x}$ here, in contrast to answer variables $\bar{x}^+$ in the case of complete answers. It remains to invoke Theorem 3.

4 Lower Bounds

The main aim of this section is to establish lower bounds that (partially) match the upper bounds stated in Theorems 1 and 2. First, however, we show that Theorem 2 cannot be strengthened by using $q^+(\bar{x}^+)$ in place of $q^+(\bar{x})$. All results presented in this section are conditional on algorithmic conjectures and assumptions. One of the conditions concerns Boolean matrix multiplication.

A Boolean $n \times n$ matrix is a function $M : [n]^2 \to \{0, 1\}$ where $[n]$ denotes the set $\{1, \dots, n\}$. The *product* of two Boolean $n \times n$ matrices M_1, M_2 is the Boolean $n \times n$ matrix $M_1 M_2 := \sum_{c=1}^{n} M_1(a, c) \cdot M_2(c, b)$ where sum and product are interpreted over the Boolean semiring. In (non-sparse) *Boolean matrix multiplication (BMM)*, one wants to compute $M_1 M_2$ given M_1 and M_2 as $n \times n$ arrays. In *sparse Boolean matrix multiplication (spBMM)*, input and output matrices M are represented as lists of pairs (a, b) with $M(a, b) = 1$. The currently best known algorithm for BMM achieves running time $n^{2.37}$ (Alman and Williams 2021) and it is open whether running time n^2 can be achieved; this would require dramatic advances in algorithm theory. Regarding spBMM, it is open whether running time $O(|M_1|+|M_2|+|M_1 M_2|)$ can be attained, that is, time linear in the size of the input and the output (represented as lists). This clearly implies BMM in time n^2, but the converse is not known.

The following implies that Theorem 2 cannot be strengthened by using $q^+(\bar{x}^+)$ in place of $q^+(\bar{x})$.

Theorem 4. *There is an OMQ $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIF}, \text{CQ})$ such that $q^+(\bar{x}^+)$ is acyclic and free-connex acyclic, but the minimal partial answers to Q cannot be enumerated in $\mathsf{DelayC_{lin}}$ unless spBMM is possible in time $O(|M_1|+|M_2|+|M_1 M_2|)$. This holds both for single wildcards and multi-wildcards.*

Proof. Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q)$ where

$$\mathcal{O} = \{A \sqsubseteq \exists f^-.\top, \mathsf{func}(f)\}$$
$$\Sigma = \{A, r_1, r_2, f\}$$
$$q(x, z, y) = r_1(x, u_1) \wedge f(z, u_1) \wedge f(z, u_2) \wedge r_2(u_2, y).$$

It is easy to see that $q^+(\bar{x}^+)$ is just q, except that now all variables are answer variables. Thus, $q^+(\bar{x}^+)$ is acyclic and free-connex acyclic, as required.

Assume to the contrary of what is to be shown that there is an algorithm that given a database $\mathcal{D}$, enumerates $Q(\mathcal{D})^*$ in $\mathsf{DelayC_{lin}}$ (the case of multi-wildcards is identical). Then this algorithm can be used, given two Boolean matrices $M_1 M_2$ in list representation, to compute $M_1 M_2$ in time $O(|M_1| + |M_2| + |M_1 M_2|)$. This is done as follows.

Given M_1 and M_2, we construct a database $\mathcal{D}$ by adding facts $r_1(a, c)$ and $A(c)$ for every (a, c) with $M_1(a, c) = 1$ and facts $r_2(c, b)$ and $A(c)$ for every (c, b) with $M_2(c, b) = 1$. It is easy to verify that $Q(\mathcal{D})^* = \{(a, *, b) \mid (a, b) \in M_1 M_2\}$. We may thus construct a list representation of $M_1 M_2$ by enumerating $Q(\mathcal{D})^*$. Since $|\mathcal{D}| = |M_1| + |M_2|$ and we can enumerate in $\mathsf{DelayC_{lin}}$, the overall time spent is $O(|M_1| + |M_2| + |M_1 M_2|)$. □

We now consider lower bounds for Theorems 1 and 2. Here, we need two additional algorithmic conjectures that are closely related, both from fine-grained complexity theory. Recall that a *k-regular hypergraph* is a pair $H = (V, E)$ where V is a finite set of vertices and $E \subseteq 2^V$ contains only sets of cardinality k. Consider the following problems:

- The *triangle detection problem* is to decide, given an undirected graph $G = (V, E)$ as a list of edges, whether G contains a 3-clique (a "triangle").
- The $(k + 1, k)$*-hyperclique problem*, for $k \geq 3$, is to decide whether a given k-uniform hypergraph H contains a hyperclique of size $k + 1$, that is, a set of $k + 1$ vertices such that each subset of size k forms a hyperedge in H.

The *triangle conjecture* states that there is no algorithm for triangle detection that runs in linear time (Abboud and Williams 2014) and the *hyperclique conjecture* states that every algorithm that solves the $(k+1,k)$-hyperclique problem, for some $k \geq 3$, requires running time at least $n^{k+1-o(1)}$ with n the number of vertices (Lincoln, Williams, and Williams 2018). Note that triangle detection is the same as $(k+1,k)$-hyperclique for $k=2$, but the formulation of the two conjectures differs in that the former refers to the number of edges and the latter to the number of nodes. The following theorem summarizes our lower bounds.

Theorem 5. *Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIF}, CQ)$ be non-empty with q self-join free and connected.*

1. *If q^+ is not acyclic, then enumerating complete answers to Q is not in $\mathsf{DelayC_{lin}}$ unless the triangle conjecture fails or the hyperclique conjecture fails.*
2. *If $q^+(\bar{x}^+)$ is acyclic, but not free-connex acyclic, then enumerating complete answers to Q is not in $\mathsf{DelayC_{lin}}$ unless spBMM is possible in time $O(|M_1| + |M_2| + |M_1 M_2|)$.*

The same is true for least partial answers, both with a single wildcard and with multi-wildcards.

Recall that we use different versions of q^+, namely $q^+(\bar{x}^+)$ and $q^+(\bar{x})$ in Theorems 1 and 2. The difference is moot for Point 1 of Theorem 2 as $q^+(\bar{x}^+)$ is acyclic if and only if $q^+(\bar{x})$ is.

The proof of Theorem 5 is inspired by proofs from (Bagan, Durand, and Grandjean 2007; Brault-Baron 2013; Carmeli and Kröll 2020) and uses similar ideas. However, the presence of ontologies and the fact that we want to capture minimal partial answers makes our proofs much more subtle. In particular, the constructions in (Carmeli and Kröll 2020) first transition from q to q^+ and then work purely on q^+, but we cannot do this due to the presence of the ontology, which is formulated in the signature of q, not of q^+. We (partially) present the proof of Point 1 and refer to the appendix for full detail.

The proof of Point 1 of Theorem 5 splits into two cases. Recall that the Gaifman graph of a CQ q is the undirected graph that has the atoms of q as its nodes and an edge between any two nodes/atoms that share a variable. It is known that if q is not acyclic, then its Gaifman graph is not chordal or not conformal (Beeri et al. 1983). Here, chordal means that every cycle of length at least 4 has a chord and conformal means that for every clique C in the Gaifman graph, there is an atom in q that contains all variables in C. The first case of the proof of Point 1 of Theorem 5 is as follows.

Lemma 2. *Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIF}, CQ)$ be non-empty such that q is self-join free and connected and the hypergraph of q^+ is not chordal. Then enumerating complete answers to Q is not in $\mathsf{DelayC_{lin}}$ unless the triangle conjecture fails. The same is true for least partial answers, both with a single wildcard and with multi-wildcards.*

The second case is formulated similarly, but refers to non-conformality and the hyperclique conjecture. We give the proof of Lemma 2.

Let $Q(\bar{x}) = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELIF}, \text{CQ})$ be as in Lemma 2, and let $y_0, \ldots, y_k$ be a chordless cycle in the Gaifman graph of q^+ that has length at least 4. Let $Y = \{y_0, \ldots, y_k\}$ and for easier reference let $y_{k+1} = y_0$. For every variable x in q, we use Y_x to denote the set of variables $y \in Y$ such that q contains a functional (possibly empty) path from x to y.

Let $G = (V, E)$ be an undirected graph. We may assume w.l.o.g. that G does not contain isolated vertices. Our aim is to construct a database $\mathcal{D}$, in time linear in $|E|$, such that G contains a triangle if and only if $Q(\mathcal{D}) \neq \emptyset$. Clearly, a $\mathsf{DelayC_{lin}}$ enumeration algorithm for Q lets us decide the latter in linear time and thus we have found an algorithm for triangle detection that runs in time linear in $|E|$, refuting the triangle conjecture.

The construction proceeds in two steps. In the first step, we define a database $\mathcal{D}_0$ that encodes the graph G. The constants in $\mathcal{D}_0$ are pairs $\langle x, f\rangle$ with $x \in \mathsf{var}(q)$ and f a partial function from Y to V. For every variable x in q and word $w = a_0 \ldots a_k \in V^*$ we use f_x^w denote the function that maps each variable $y_i \in Y_x$ to a_i and is undefined on all other variables. We may treat E as a symmetric (directed) relation, writing e.g. $(a,b) \in E$ and $(b,a) \in E$ if $\{a,b\} \in E$. For every atom $r(x,y)$ in q with $r \in \Sigma$, add the following facts to $\mathcal{D}_0$:

1. if $y_0 \in Y_x \cup Y_y$: $r(\langle x, f_x^{ab^k}\rangle, \langle y, f_y^{ab^k}\rangle)$ for all $(a,b) \in E$,
2. if $y_k \in Y_x \cup Y_y$: $r(\langle x, f_x^{a^k b}\rangle, \langle y, f_y^{a^k b}\rangle)$ for all $(a,b) \in E$,
3. if neither is true: $r(\langle x, f_x^{b^{k+1}}\rangle, \langle y, f_y^{b^{k+1}}\rangle)$ for all $a \in V$.

In addition, we add the fact $A(c)$ for every concept name $A \in \Sigma$ and every constant c introduced above.

To provide an intuition for the reduction, let us start with a description that is relatively simple, but inaccurate. Consider a homomorphism h from q to $\mathcal{D}_0$. It can be shown that h must map every variable y_i to a constant of the form $\langle y_i, f_{y_i}\rangle$ and that the domain of the function f_{y_i} is $\{y_i\}$. Since $f_{y_i}(y_i)$ is a node from G, the homomorphism h thus identifies a sequence of nodes $a_0, \ldots, a_k$ from G, with $a_i = f_{y_i}(y_i)$. The construction of $\mathcal{D}_0$ ensures that $a_1 = \cdots = a_{k-1}$ and a_0, a_1, a_k forms a triangle in G. Conversely, every triangle in G gives rise to a homomorphism from q to $\mathcal{D}_0$ of the described form. For other variables x from q, the use of the function f_x in constants $\langle x, f_x\rangle$ serves the purpose of ensuring that all functionality assertions in $\mathcal{O}$ are satisfied in $\mathcal{D}_0$. A concrete example for the construction of $\mathcal{D}_0$ is provided in the appendix.

The above description is inaccurate for several reasons. First, instead of homomorphisms into $\mathcal{D}_0$, we need to consider homomorphisms into the universal model $\mathcal{U}_{\mathcal{D}_0,\mathcal{O}}$ (defined in the appendix). Then variables y_i need not be mapped to a constant $\langle y_i, f_{y_i}\rangle$, but can also be mapped to elements outside of $\mathsf{adom}(\mathcal{D}_0)$. This does not break the reduction but complicates the correctness proof. Another difficulty arises from the fact that $\mathcal{O}$ and q may use symbols that do not occur in Σ and and we need these to be derived by $\mathcal{O}$ at the relevant points in $\mathcal{D}_0$. This is achieved in the second step of the construction of $\mathcal{D}_0$, described next.

Informally, we want $\mathcal{O}$ to derive, at every constant $c \in \mathsf{adom}(\mathcal{D}_0)$, anything that it could possibly derive at any con-

stant in any database. This is achieved by attaching certain tree-shaped databases to every constant in $\mathcal{D}_0$. We next make this precise. Let $\mathcal{R}_\Sigma$ be the set of all role names from Σ and their inverses. The infinite tree-shaped Σ-database D_ω has as its active domain $\mathsf{adom}(\mathcal{D}_\omega)$ the set of all (finite) words over alphabet $\mathcal{R}_\Sigma$ and contains the following facts:

- $A(w)$ for all $w \in \mathsf{adom}(\mathcal{D}_\omega)$ and concept names $A \in \Sigma$;
- $r(w, w')$ for all $w, w' \in \mathsf{adom}(\mathcal{D}_\omega)$ with $w' = wr$;
- $r(w', w)$ for all $w, w' \in \mathsf{adom}(\mathcal{D}_\omega)$ with $w' = wr^-$.

We cannot directly use $\mathcal{D}_\omega$ in the construction of $\mathcal{D}$ since it is infinite. Consider all concept names A such that $\mathcal{D}_\omega, \mathcal{O} \models A(\varepsilon)$. We prove in the appendix that these are precisely the concept names A that are *non-empty*, that is, $\mathcal{D}, \mathcal{O} \models A(c)$ for some database $\mathcal{D}$ and some $c \in \mathsf{adom}(\mathcal{D})$. Clearly the number of such concept names A is finite. By compactness, there is thus a finite database $\mathcal{D}_{\mathsf{tree}} \subseteq \mathcal{D}_\omega$ such that $\mathcal{D}_{\mathsf{tree}}, \mathcal{O} \models A(\varepsilon)$ for all non-empty concept names A. We may w.l.o.g. assume that $\mathcal{D}_{\mathsf{tree}}$ is the initial piece of $\mathcal{D}_\omega$ of some finite depth $k \geq 1$.

In principle, we would like to attach a copy of $\mathcal{D}_{\mathsf{tree}}$ at every constant in $\mathcal{D}_0$. This, however, might violate functionality assertions in $\mathcal{O}$ and thus we have to be a bit more careful. For a role $R \in \{r, r^-\}$ with $r \in \Sigma$, let $\mathcal{D}_R \subseteq \mathcal{D}_{\mathsf{tree}}$ be the database that consists of the fact $R(\varepsilon, R)$ and the subtree in $\mathcal{D}_{\mathsf{tree}}$ rooted at R. Now, the final database $\mathcal{D}$ used in the reduction is obtained from $\mathcal{D}_0$ as follows: for every $c \in \mathsf{adom}(\mathcal{D}_0)$ and every role $R \in \{r, r^-\}$ with $r \in \Sigma$ such that there is no fact $R(c, c') \in \mathcal{D}_0$, add a disjoint copy of $\mathcal{D}_R$, glueing the copy of ε to c.

It is easy to see that $\mathcal{D}$ can be computed in time $O(||E||)$. In particular, the database $\mathcal{D}_{\mathsf{tree}}$ can be constructed (in time independent of $\mathcal{D}$) by generating initial pieces of $\mathcal{D}_\omega$ of increasing depth and checking whether all non-empty concept names are implied at ε. In the appendix, we show that $\mathcal{D}$ satisfies all functionality assertions in $\mathcal{O}$ and is derivation complete at $\mathsf{adom}(\mathcal{D}_0)$. We then use a rather subtle analysis to prove the following.

Lemma 3.

TD1 *If there is a minimal partial answer to Q on $\mathcal{D}$ (with a single wildcard or with multiple wildcards), then there is a triangle in G.*

TD2 *If there is a triangle in G then there is a complete answer to Q on $\mathcal{D}$.*

5 Combined Complexity of Single-Testing

The results on enumeration provide a (mild) indication that partial answers can be computationally more challenging than complete ones: the condition used in Theorem 1 is weaker than that in Theorem 2, and Theorem 1 achieves $\mathsf{CD}\circ\mathsf{Lin}$ while Theorem 2 achieves only $\mathsf{DelayC_{lin}}$. Other cases in point may be found in (Lutz and Przybylko 2022b). This situation prompts us to study the effect of answer partiality on the combined complexity of single-testing.

We concentrate on the fragments $\mathcal{EL}$ and $\mathcal{ELH}$ of $\mathcal{ELIHF}$ that do not admit inverse roles and functionality assertions and, in the case of $\mathcal{EL}$, also no role inclusions. These DLs bear special importance as single-testing complete answers to OMQs $Q = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELH}, \text{CQ})$ is in PTIME if q is acyclic and NP-complete otherwise, both in combined complexity, and thus no harder than without ontologies (Krötzsch, Rudolph, and Hitzler 2007; Bienvenu et al. 2013). We show that making answers partial may have an adverse effect on these complexities, starting, however, with a positive result. It is proved by a Turing-reduction to single-testing complete answers.

Theorem 6. *For OMQs $Q = (\mathcal{O}, \Sigma, q) \in (\mathcal{ELH}, \mathit{CQ})$ with q acyclic, single-testing minimal partial answers with a single-wildcard is in* PTIME *in combined complexity.*

Partial answers with multi-wildcards are less well-behaved. The lower bound in the next result is proved by a reduction from 1-in-3-SAT and only needs a very simple ontology that consists of a single CI of the form $A \sqsubseteq \exists r.\top$.

Theorem 7. *For OMQs $Q = (\mathcal{O}, \Sigma, q) \in (\mathcal{EL}, \mathit{CQ})$ with q acyclic, single-testing minimal partial answers with multi-wildcards is* NP*-complete in combined complexity. The same is true in $(\mathcal{ELH}, \mathit{CQ})$*

We now move from acyclic to unrestricted CQs. This makes the complexity increase further, and the difference between single and multi-wildcards vanishes.

Theorem 8. *For OMQs $Q = (\mathcal{O}, \Sigma, q) \in (\mathcal{EL}, \mathit{CQ})$ single-testing minimal partial answers is* DP*-complete in combined complexity. This is true both for single wildcards and multi-wildcards, and the same holds also in $(\mathcal{ELH}, \mathit{CQ})$.*

For most other OMQ languages, we do not expect a difference in complexity between single-testing complete answers and single-testing partial answers. As an example, we consider $\mathcal{ELIHF}$ where single-testing complete answers is EXPTIME-complete (Eiter et al. 2008).

Theorem 9. *In $(\mathcal{ELIHF}, \mathit{CQ})$, single-testing minimal partial answers is* EXPTIME*-complete in combined complexity, both with single wildcards and multi-wildcards.*

We remark that the *data* complexity of single-testing minimal partial answers in $(\mathcal{ELIHF}, \text{CQ})$ is in PTIME, both with a single wildcard and with multi-wildcards. This can be shown by using essentially the same arguments as in the proof of Theorem 6.

6 Conclusion

It would be interesting to extend our results to $\mathcal{ELIHF}$ with local functionality assertions, that is, with concepts of the form $(\leqslant 1\ R)$ or even $(\leqslant 1\ R\ C)$. This is non-trivial as it is unclear how to define the CQ extension q^+. It would also be interesting and non-trivial to get rid of self-join freeness in the lower bounds, see (Berkholz, Gerhardt, and Schweikardt 2020; Carmeli and Segoufin 2022). Another natural question is whether answers can be enumerated in some given order, see e.g. (Carmeli et al. 2021). Note that it was observed in (Lutz and Przybylko 2022b) that when enumerating $Q(\mathcal{D})^*$ or $Q(\mathcal{D})^{\mathcal{W}}$, it is possible to enumerate the complete answers before the truely partial ones.

Acknowledgements

The authors were supported by the DFG project LU 1417/3-1 QTEC.

References

Abboud, A.; and Williams, V. V. 2014. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *Proc. of FOCS 2014*, 434–443. IEEE Computer Society.

Alman, J.; and Williams, V. V. 2021. A Refined Laser Method and Faster Matrix Multiplication. In *SODA 2021*, 522–539. SIAM.

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

Bagan, G.; Durand, A.; and Grandjean, E. 2007. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *Proc. of CSL 2007*, volume 4646 of *LNCS*, 208–222. Springer.

Beeri, C.; Fagin, R.; Maier, D.; and Yannakakis, M. 1983. On the Desirability of Acyclic Database Schemes. *J. ACM*, 30: 479–513.

Berkholz, C.; Gerhardt, F.; and Schweikardt, N. 2020. Constant delay enumeration for conjunctive queries: a tutorial. *ACM SIGLOG News*, 7(1): 4–33.

Bienvenu, M.; Ortiz, M.; Simkus, M.; and Xiao, G. 2013. Tractable Queries for Lightweight Description Logics. In *Proc. of IJCAI13*, 768–774. IJCAI/AAAI.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4): 33:1–33:44.

Brault-Baron, J. 2013. *De la pertinence de l'énumération: complexité en logiques propositionnelle et du premier ordre. (The relevance of the list: propositional logic and complexity of the first order)*. Ph.D. thesis, University of Caen Normandy, France.

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

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; and Rosati, R. 2009. Ontologies and Databases: The DL-Lite Approach. In *Proc. of Reasoning Web*, volume 5689 of *LNCS*, 255–356. Springer.

Carmeli, N.; and Kröll, M. 2020. Enumeration Complexity of Conjunctive Queries with Functional Dependencies. *Theory Comput. Syst.*, 64(5): 828–860.

Carmeli, N.; and Segoufin, L. 2022. Conjunctive queries with self-joins, towards a fine-grained complexity analysis. *CoRR*, abs/2206.04988.

Carmeli, N.; Tziavelis, N.; Gatterbauer, W.; Kimelfeld, B.; and Riedewald, M. 2021. Tractable Orders for Direct Access to Ranked Answers of Conjunctive Queries. In *Proc. of PODS*, 325–341. ACM.

Cook, S. A.; and Reckhow, R. A. 1973. Time Bounded Random Access Machines. *J. Comput. Syst. Sci.*, 7(4): 354–375.

Dowling, W. F.; and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3): 267–284.

Eiter, T.; Gottlob, G.; Ortiz, M.; and Simkus, M. 2008. Query Answering in the Description Logic Horn-$\mathcal{SHIQ}$. In *Proc. of JELIA*, volume 5293 of *LNCS*, 166–179. Springer.

Grandjean, E. 1996. Sorting, linear time and the satisfiability problem. *Annals of Mathematics and Artificial Intelligence*, 16: 183–236.

Johnson, D. S.; and Klug, A. C. 1982. Testing Containment of Conjunctive Queries Under Functional and Inclusion Dependencies. In *Proc. of PODS*, 164–169. ACM.

Kazana, W. 2013. *Query evaluation with constant delay. (L'évaluation de requêtes avec un délai constant)*. Ph.D. thesis, École normale supérieure de Cachan, Paris, France.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2007. Conjunctive Queries for a Tractable Fragment of OWL 1.1. In *Proc. of ISWC*, volume 4825 of *LNCS*, 310–323. Springer.

Lincoln, A.; Williams, V. V.; and Williams, R. R. 2018. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *Proc. of SODA 2018*, 1236–1252. SIAM.

Lutz, C.; and Przybylko, M. 2022a. Efficient Answer Enumeration in Description Logics with Functional Roles – Extended Version. *CoRR*, abs/2211.15248.

Lutz, C.; and Przybylko, M. 2022b. Efficiently Enumerating Answers to Ontology-Mediated Queries. In *Proc. of PODS*, 277–289. ACM.

Lutz, C.; and Przybylko, M. 2022c. Efficiently Enumerating Answers to Ontology-Mediated Queries. *CoRR*, abs/2203.09288.

Segoufin, L. 2015. Constant Delay Enumeration for Conjunctive Queries. *SIGMOD Rec.*, 44(1): 10–17.