EPiC Series in Computing

Volume XXX, 2019, Pages 3–16

GCAI 2019. 5th Global Conference on Artificial Intelligence

Ontology-Mediated Queries from Examples: a Glimpse at the DL-Lite Case*

Magdalena Ortiz

Institute of Logic and Computation, Faculty of Informatics, TU Wien
ortiz@kr.tuwien.ac.at

Abstract

Reverse engineering queries from given data, as in the case of query-by-example and query definability, is an important problem with many applications that has recently gained attention in the areas where symbolic artificial intelligence meets learning. In the presence of ontologies this problem was recently studied for Horn-$\mathcal{ALC}$ and Horn-$\mathcal{ALCI}$. The main contribution of this paper is to take a first look at the case of DL-Lite, to identify cases where the addition of the ontology does not increase the worst-case complexity of the problem. Unfortunately, reverse engineering conjunctive queries is known to be very hard, even for plain databases, since the smallest witness query is known to be exponential in general. In the light of this, we outline some possible research directions for exploiting the ontology in order to obtain smaller witness queries.

1 Introduction

In *reverse engineering* of queries, we are given a dataset together with desired answers, and want to build a query that gives those answers when evaluated. It is often advocated as a way to facilitate query formulation by non-experts, but its applications go far beyond. Since it was formulated by Zloof in the 70s [23], researchers have come across it in data exploration and analysis, usability, data security, the study of expressiveness of query languages, etc., see [15, 16]. It has been studied for different query languages and different types of databases, including relational data [5, 19, 21, 14, 20], graph databases [7, 1], and RDF data [2]. Much work has aimed at finding practicable solutions [15, 20, 14, 11, 9]. Not surprisingly, reverse engineering of queries also arises in AI, where learning from given data how to separate classes is the core goal of entire research fields. Indeed, query reverse engineering can be seen as a form of inductive logic programming [17]. Moreover, the problem has been shown to be useful for exploiting database techniques to improve the feature engineering process for machine learning [4, 13]. It is clearly a core theoretical problem at the interface of learning and symbolic AI.

A setting where remarkably little has been done is query reverse engineering in the presence of *Description Logic (DL)* ontologies. Over the last two decades, *ontology mediated queries (OMQs)*, which enrich standard queries with domain knowledge expressed as a DL ontology, have been extensively studied (see [6, 18] and their references). Among other uses, they are advocated as a powerful tool for querying incomplete data, facilitating access to complex data sources by less experienced users,

*This work was partially supported by the Austrian Science Fund (FWF) projects P30360 and P30873.

and integrating data (see [22] and its references). In reverse engineering queries, the knowledge in the ontology has the potential to help us find more and better solutions, as we argue in this paper. It is then surprising how little this has been addressed. Although some related problems—like learnability of some combinations of ontologies and rules [12]—had been studied before, the very recent [10] is to our knowledge the first work to focus on this topic.

From the theoretical perspective, the study of query reverse engineering focuses on solving two problems. In *query definability (*QDEF*)*, the input is a dataset and a set of tuples that are desired answers or *positive examples*, and we want to find a query that retrieves precisely those tuples. In the more general *query-by-example (*QBE*)* we have both positive and *negative* examples, and we want to find a query that gives all the positive examples and none of the negative ones. We study the variations of these two problems where the input also includes a DL ontology, whose consequences are taken into account for answering the queries.

Example 1. Consider the dataset on the left column. On the right we have a *DL-Lite*$_\mathcal{R}$ ontology expressing that a person that receives chemotherapy takes immunosuppressive (IS) medication, that the range of the relation 'takes IS medication' is the class of IS medications, and that taking an IS medication implies taking a medication.

$$\begin{array}{ll} \mathsf{takesMed}(p_1, \mathit{insulin}) & \mathsf{ReceivesChemo}(p_4) \\ \mathsf{takesISMed}(p_2, \mathit{azasan}) & \mathsf{IsImmunoDef}(p_5) \\ \mathsf{takesISMed}(p_3, \mathit{prednisone}) & \end{array} \qquad \begin{array}{r} \mathsf{ReceivesChemo} \sqsubseteq \exists \mathsf{takesISMed} \\ \exists \mathsf{takesISMed}^- \sqsubseteq \mathsf{ISMed} \\ \mathsf{takesISMed} \sqsubseteq \mathsf{takesMed} \end{array}$$

If we are given positive examples $S^+ = \{p_2, p_3, p_4\}$ and negative example $S^- = \{p_1\}$, we can solve positively the QBE problem, as witnessed by the query $q(x) = \exists y \mathsf{takesMed}(x, y) \wedge \mathsf{ISMed}(y)$. The examples $S^+ = \{p_2, p_3, p_4, p_5\}$ and $S^- = \{p_1\}$ can be separated with a UCQ $q(x) = \mathsf{IsImmunoDef}(x) \vee \big(\exists y \mathsf{takesMed}(x, y) \wedge \mathsf{ISMed}(y)\big)$. Verifying if they can be separated with a CQ requires the rather involved construction in Section 3.2. For a case where no separating query exists, consider the positive examples $S^+ = \{(p_1, \mathit{insulin}), (p_2, \mathit{asazan})\}$ and the negative example $S^- = \{(p_3, \mathit{prednisone})\}$. There is no witness query that can retrieve (p_2, asazan) while avoiding $(p_3, \mathit{prednisone})$. For the same reason, any S^+ including (p_2, asazan) but not $(p_3, \mathit{prednisone})$ is a negative instance of QDEF.

QBE and QDEF are highly relevant in many data management and AI scenarios, but unfortunately they are computationally quite expensive. In the traditional setting with no ontologies, both QBE and QDEF are CONEXPTIME-complete for *conjunctive queries (CQs)*, which are at the core of practically all languages for querying relational data. They remain intractable for *unions of CQs (UCQs)*, but being CONP complete they appear more amenable to be solved using tools for other intractable problems, and for identifying tractable subcases [5]. Taking an ontology into account can increase the complexity even further. The results of [10] show that the complexity increases rather significantly if we add an ontology in the well-known Horn DLs Horn-$\mathcal{ALC}$ or Horn-$\mathcal{ALCI}$. For both QBE and QDEF they prove tight bounds of CONEXPTIME for Horn-$\mathcal{ALC}$ and 2EXPTIME for Horn-$\mathcal{ALCI}$ if one considers CQs, and EXPTIME for Horn-$\mathcal{ALC}$ and 2EXPTIME for Horn-$\mathcal{ALCI}$ in the case of UCQs. If the signature is not restricted, and all concepts and roles are allowed in the query, the bounds for CQs and the bounds for Horn-$\mathcal{ALCI}$ are the same, but the complexity increase of UCQs and Horn-$\mathcal{ALC}$ is only to EXPTIME rather than 2EXPTIME. But even in this case, the news are rather negative. Are there useful scenarios of reverse query engineering in the presence of ontologies that are not so computationally costly?

In this paper, we give the first step towards a positive answer. We consider the so-called *DL-Lite* family of DLs, which not only have the lowest complexity among standard DLs, but they are also the most popular in the context of data access and management. We show that for *DL-Lite*$_\mathcal{R}$ and UCQs,

the complexity of QBE and QDEF is in coNP, and thus not harder than for plain databases. That is, we can add and leverage an ontology at no computational cost. Furthermore, we can implement both problems using existing techniques like *query rewriting* for *DL-Lite*$_\mathcal{R}$. In the last part of the paper, we go beyond just taking the ontology into account as done in [10], and instead advocate to *really exploit the ontology*. Revisiting our example, $S^+ = \{p_2, p_3, p_4, p_5\}$ and $S^- = \{p_1\}$ can be separated with the UCQ $q(x) = \mathsf{IsImmunoDef}(x) \vee \big(\exists y \mathsf{takesISMed}(x, y)\big)$, but we may prefer to introduce a new concept for immunocompromised patients, and use the CQ $q(x) = \mathsf{IsImmunoCompr}(x)$ and while adding to the ontology axioms $\mathsf{IsImmunoDef} \sqsubseteq \mathsf{IsImmunoCompr}$ and $\exists \mathsf{takesISMed} \sqsubseteq \mathsf{IsImmunoCompr}$. We discuss in Section 5 some first ideas towards reverse engineering smaller and more readable queries using more predicates and axioms in the ontology.

2 Preliminaries

As usual in the context of OMQs, we consider datasets whose relations are only unary and binary, that is, concepts and roles, and we focus on conjunctive queries and their unions.

2.1 Datasets and queries

Predicates come from two sets: the set C of *concept names*, and the set R of *role names*; the former have arity 1 and the latter 2. *Terms* are the elements of the set I of individuals and of the set V of variables. All these sets are countably infinite and pairwise disjoint. *Query atoms* are either *concept atoms* of the form $A(x)$ with $A \in \mathsf{C}$ and $x \in \mathsf{V}$, or *role atoms* of the form $r(x, y)$ with $r \in \mathsf{R}$ and $x, y \in \mathsf{V}$. We use $\mathsf{V}(q)$ for the variables occurring in a set of atoms q.

We define two kinds of *queries*. First, *conjunctive queries (CQs)* are defined as pairs $(q, \vec{x})$, where q is a finite set of atoms, and $\vec{x} \subseteq \mathsf{V}(q)$. The variables in $\vec{x}$ are called *answer variables*, and the remaining variables in $\mathsf{V}(q)$ are called *existential variables*. If n is the arity of $\vec{x}$, we call q an n-ary query. We usually call 0-ary queries *Boolean*, and write just q rather than $(q, ())$.

Our second query language are *union of conjunctive queries (UCQs)*. A UCQ is a finite set Q of CQs $\{(q_1, \vec{x}), \ldots, (q_n, \vec{x})\}$ sharing the set $\vec{x}$ of answer variables. The notions of answer and existential variables, arity and Boolean query extend naturally to UCQs.

Assertions are analogous to query atoms, but individuals a, b take the place of variables. A *dataset* $\mathcal{D}$ is a finite set of assertions, and $\mathsf{I}(\mathcal{D})$ denotes the individuals that occur in it.

An n-ary CQ $(q, \vec{x})$ defines a mapping ans from datasets $\mathcal{D}$ to n-ary relations as follows. A *valuation* is a mapping from variables to individuals. An n-ary tuple of individuals $\vec{a}$ is called an *answer to* $(q, \vec{x})$ *over* $\mathcal{D}$ if there is a valuation v such that $v(q) \subseteq \mathcal{D}$ and $v(\vec{x}) = \vec{a}$. Then $ans((q, \vec{x}), \mathcal{D})$ is the set of answers to $(q, \vec{x})$ over $\mathcal{D}$. Naturally, for a UCQ $(q, \vec{x})$, we have that $ans((q, \vec{x}), \mathcal{D})$ is the union of $ans(q_i(\vec{x}), \mathcal{D})$ for all $q_i \in Q$. Note that for Boolean queries the only possible answer is the empty tuple $()$; we may thus say that Q *is true* if $ans(Q, \mathcal{D}) = \{()\}$, and that Q *is false* if $ans(Q, \mathcal{D}) = \{\}$.

2.2 Ontologies

Many DLs can be used for writing ontologies [3]. In this paper, we focus on *DL-Lite*$_\mathcal{R}$ [8].

In *DL-Lite*$_\mathcal{R}$, *roles* take the forms r or r^- with $r \in \mathsf{R}$, and *concepts* B are either $A \in \mathsf{C}$, or $\exists r$ with r a role. *DL-Lite*$_\mathcal{R}$ ontologies $\mathcal{O}$ are finite sets of *axioms* of the following forms,where B_1 and B_2 are concepts and r_1 and r_2 are roles:

$$B_1 \sqsubseteq B_2 \qquad B_1 \sqcap B_2 \sqsubseteq \bot \qquad r_1 \sqsubseteq r_2 \qquad r_1 \sqcap r_2 \sqsubseteq \bot.$$

To give semantics to ontologies, *interpretations* are defined as pairs $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}} \neq \emptyset$ is the *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that maps concept names to subsets of $\Delta^{\mathcal{I}}$, and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We extend the function $\cdot^{\mathcal{I}}$ to roles r^- and concepts $\exists r$:

$$(r^-)^{\mathcal{I}} = \{(d', d) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (d, d') \in r^{\mathcal{I}}\}$$
$$(\exists r)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, d') \in r^{\mathcal{I}} \text{ for some } d' \in \Delta^{\mathcal{I}}\}$$

We say that an interpretation $\mathcal{I}$ *satisfies* an axiom $E_1 \sqsubseteq E_2$ if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$, and that $\mathcal{I}$ *satisfies* an axiom $E_1 \sqcap E_2 \sqsubseteq \bot$ if $E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}} = \emptyset$. We call $\mathcal{I}$ a *model* of an ontology $\mathcal{O}$ and write $\mathcal{I} \models \mathcal{O}$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$.

It will sometimes be convenient to manipulate interpretations as sets of atoms. Given an interpretation $\mathcal{I}$, an $\mathcal{I}$-atom may take the form $A(d)$ or $r(d, d')$, where $A \in \mathsf{C}$, $r \in \mathsf{R}$, and $d, d' \in \Delta^{\mathcal{I}}$. We may write $r^-(d, d')$ to mean the $\mathcal{I}$-atom $r(d', d)$. We identify $\mathcal{I}$ with the set that contains exactly the $\mathcal{I}$-atoms $A(d)$ with $d \in A^{\mathcal{I}}$, and $r(d, d')$ with $(d, d') \in r^{\mathcal{I}}$. Note that, in particular, an interpretation with $\Delta^{\mathcal{I}} \subseteq \mathsf{I}$ is in fact a dataset.

2.3 Ontology-mediated Queries

In this work an *ontology-mediated query* (OMQ) has the form $(Q, \vec{x}, \mathcal{O})$ where $(Q, \vec{x})$ is a CQ or UCQ as above, and $\mathcal{O}$ is a *DL-Lite*$_{\mathcal{R}}$ ontology. Like ordinary queries, OMQs define a mapping ans from datasets to relations of the same arity as the query. However, for the semantics of OMQs we take a *open-world* view, where a dataset $\mathcal{D}$ is viewed as a possibly incomplete world descriptions and the ontology may imply the existence of facts not explicit in $\mathcal{D}$.

To give semantics to OMQs we extend modelhood to datasets. We say that an interpretation $\mathcal{I}$ satisfies an assertion $A(a)$ if $a \in \Delta^{\mathcal{I}}$ and $a \in A^{\mathcal{I}}$. Similarly, it satisfies an assertion $r(a, b)$ if $\{a, b\} \subseteq \Delta^{\mathcal{I}}$ and $(a, b) \in r^{\mathcal{I}}$. Then $\mathcal{I}$ is a model of a dataset $\mathcal{D}$ (in symbols, $\mathcal{I} \models \mathcal{D}$) if it satisfies every assertion in $\mathcal{D}$. Given an ontology $\mathcal{O}$, we write $\mathcal{I} \models \mathcal{O}, \mathcal{D}$ if $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{D}$. For a concept B and an individual a, we write $\mathcal{O}, \mathcal{D} \models B(a)$ if $a \in B^{\mathcal{I}}$ for every $\mathcal{I}$ with $\mathcal{I} \models \mathcal{O}, \mathcal{D}$.

We remark that we make the standard name assumption (SNA) because it is convenient and closer to standard databases, but this assumption is not necessary for our results.

We extend the notion of valuations to interpretations. Rather than that mapping V to I, $\mathcal{I}$-valuations now map variables in V to objects in $\Delta^{\mathcal{I}}$. Given an interpretation $\mathcal{I}$ and a CQ $(q, \vec{x})$, answers are defined similarly as above: an n-ary tuple of individuals $\vec{a}$ is called an *answer to* $(q, \vec{x})$ *over* $\mathcal{I}$ if there is an $\mathcal{I}$-valuation v such that $v(q) \subseteq \mathcal{I}$ and $v(\vec{x}) = \vec{a}$. For a UCQ $(Q, \vec{x})$, we have that $ans((Q, \vec{x}), \mathcal{I})$ is the union of $ans(q_i(\vec{x}), \mathcal{I})$ for all $q_i \in Q$. Finally, we can now define *certain answers*: given an n-ary OMQ $(Q, \vec{x}, \mathcal{O})$ and a dataset $\mathcal{D}$, we define

$$ans((Q, \vec{x}, \mathcal{O}), \mathcal{D}) = \{\vec{a} \in \mathsf{I}(\mathcal{D})^n \mid \vec{a} \in ans((Q, \vec{x}), \mathcal{I}) \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \mathcal{O}, \mathcal{D}\}.$$

We remark that, given a dataset $\mathcal{D}$ and an ontology $\mathcal{O}$, it can be the case that there is no $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O}, \mathcal{D}$. In this case, we say that $(\mathcal{D}, \mathcal{O})$ *is inconsistent*, and have that $ans((Q, \vec{x}, \mathcal{O}), \mathcal{D}) = \mathsf{I}(\mathcal{D})^n$ for every n-ary query $(Q, \vec{x})$.

3 Reverse Engineering of Queries

We define two central query reverse-engineering problems: *query-by-example (QbE)* and *query definability (QDef)*. In what follows we use $\mathcal{Q} \in \{CQ, UCQ\}$ to denote query languages.

PROBLEM:	**QBE($\mathcal{Q}$)**
INPUT:	A dataset $\mathcal{D}$ and two sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$.
QUESTION:	Is there a query $(Q, \vec{x})$ in $\mathcal{Q}$ such that $S^+ \subseteq ans((Q, \vec{x}), \mathcal{D})$ and $S^- \cap ans((Q, \vec{x}), \mathcal{D}) = \emptyset$?

PROBLEM:	**QDEF($\mathcal{Q}$)**
INPUT:	A dataset $\mathcal{D}$ and a set S^+ of tuples from $\mathsf{I}(\mathcal{D})^n$.
QUESTION:	Is there a query $(Q, \vec{x})$ in $\mathcal{Q}$ such that $S^+ = ans((Q, \vec{x}), \mathcal{D})$?

If if exists, we call $(Q, \vec{x})$ a *witness (query)*. Note that the QBE case trivializes if $S^+ \cap S^- \neq \emptyset$, so we will quietly assume the two sets are disjoint in the paper.

QBE and QDEF are rather well understood for the standard setting of CQs and UCQs over plain databases. We review a few central results, mostly from [1, 19, 21]. For an in-depth discussion, please refer to [5] (whose presentation we follow closely) and its references.

Theorem 1 ([1, 19, 21])**.** QBE*(CQ) and* QDEF*(CQ) are* coNEXPTIME *complete, while* QBE*(UCQ) and* QDEF*(UCQ) are* coNP *complete.*

In all four cases, the upper bounds are obtained by testing for the existence of homomorphisms between structures constructed from $\mathcal{D}$, S^+ and S^-. As we will see below, these structures also give us witness queries.

In general, by a *structure* we mean a set of atoms, which may be query atoms (which use only variables), assertions (which use only individuals), or $\mathcal{I}$-atoms (which use objects of some domain containing the individuals). So, we have three types of structures S, and all of them have a *domain* of objects that we denote $dom(S)$:

- A database $\mathcal{D}$ is a finite structure whose domain is $\mathsf{I}(\mathcal{D})$.
- A set of query atoms Q is finite a structure whose domain is $\mathsf{V}(Q)$.
- An interpretation $\mathcal{I}$ is a possibly infinite structure whose domain is $\Delta^{\mathcal{I}}$.

All these structures are similar, and their atoms use only concepts and roles. We are sometimes interested in the following structures:

- Given a pair $(\mathcal{D}, \vec{a})$, we denote by $query(\mathcal{D}, \vec{a})$ the CQ obtained by replacing in $\mathcal{D}$ each individual $a \in \vec{a}$ by a variable $x_a \in \mathsf{V}$.
- Given an interpretation $\mathcal{I}$, we denote by $\mathcal{I}|_{\mathsf{I}}$ the result of restricting its domain to the individuals. Note that $\mathcal{I}|_{\mathsf{I}}$ is a dataset whenever it is finite.

A *homomorphism* from structure S to structure S' is a mapping h from $dom(S)$ to $dom(S')$ such that $h(\alpha) \in S'$ for every atom $\alpha \in S$. We write $S \to S'$ if such a homomorphism exists. It is sometimes important to distinguish a tuple $\vec{e}$ of elements in a structure S. We do so by writing $(S, \vec{e})$, and for two such pairs we write $(S, \vec{e}) \to (S', \vec{e'})$ if there is a homomorphism from S to S' with $h(\vec{e}) = \vec{e'}$.

We can now describe how the upper bounds of Theorem 1 are obtained, together with a witness query for a given input if it exists.

3.1 The case of UCQs

First, for deciding QBE(UCQ) we can use the following test:

QBE(UCQ) Test
INPUT: A dataset $\mathcal{D}$ and two sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$.

Is it the case that $(\mathcal{D}, \vec{a}) \nrightarrow (\mathcal{D}, \vec{b})$ for all $\vec{a} \in S^+$ and all $\vec{b} \in S^-$?

The answer to this test is positive iff we have a positive instance of QBE(UCQ), and in that case, we also have a *canonical witness* UCQ: the union of the CQs $query(\mathcal{D}, \vec{a})$ with $\vec{a} \in S^+$ (after unifying the answer variables via renaming). A very similar procedure gives us a solution for QDEF(UCQ), now the only difference is that, instead of testing $(\mathcal{D}, \vec{a}) \nrightarrow (\mathcal{D}, \vec{b})$ for each $\vec{b}$ in S^+, we must test every n-ary tuple from $\mathsf{I}(\mathcal{D})$ that is not in S^+.

QDEF(UCQ) Test
INPUT: A dataset $\mathcal{D}$ and a set S^+ of tuples from $\mathsf{I}(\mathcal{D})^n$.

Is it the case that $(\mathcal{D}, \vec{a}) \nrightarrow (\mathcal{D}, \vec{b})$ for all $\vec{a} \in S^+$, and all $\vec{b} \in \mathsf{I}(\mathcal{D})^n \setminus S^+$?

The canonical witness query for the QDEF(UCQ) instance is exactly the same in this case.

Our canonical witness query has size $O(|\mathcal{D}| \times |S^+|)$, and is hence polynomial in the input. Deciding $(\mathcal{D}, \vec{a}) \rightarrow (\mathcal{D}, \vec{b})$ is a well-known NP-complete problem. Hence the coNP upper bound for both QBE(UCQ) and QDEF(UCQ) easily follows: if the answer to either test is false, we can guess the tuples $\vec{a}$ and $\vec{b}$ and check $(\mathcal{D}, \vec{a}) \rightarrow (\mathcal{D}, \vec{b})$ in NP. Not surprisingly, this is tight [1].

Example 2. In a dataset storing suppliers of some company we have two roles R_A and R_B, which respectively store suppliers of products of type A and of products of type B, and two roles L_{EU} and L_{nEU} that store the locations of the suppliers in European Union countries and in non-European Union ones. Consider the following dataset $\mathcal{D}_s$:

$$R_A(s_1, p_1) \qquad R_B(s_2, p_2) \qquad L_{\text{EU}}(s_2, lit) \qquad L_{\text{nEU}}(s_1, tur)$$

If we have $S^+ = \{(s_1, p_1), (s_2, lit)\}$ and $S^- = \{(s_2, p_2), (s_1, tur)\}$, our canonical witness query (written in logic notation for readability) is:

$$\begin{aligned} q(x, y) = &\big(\exists x_{s_2}, x_{p_2}, x_l, x_t \; R_A(x, y) \wedge R_B(x_{s_2}, x_{p_2}) \wedge L_{\text{EU}}(x_{s_2}, x_l) \wedge L_{\text{nEU}}(x, x_t)\big) \vee \\ &\big(\exists x_{s_1}, x_{p_1}, x_{p_2}, x_t \; R_A(x_{s_1}, x_{p_1}) \wedge R_B(x, x_{p_2}) \wedge L_{\text{EU}}(x, y) \wedge L_{\text{nEU}}(x_{s_1}, x_t)\big)\} \end{aligned}$$

Observe that there is no homomorphism h of this query into $\mathcal{D}_s$ that would give us $h(x) = s_2$ and $h(y) = p_2$, or $h(x) = s_1$ and $h(y) = tur$. Canonical witnesses may contain unnecessary atoms, as in this case, where we can easily see that the simpler query $q(x, y) = R_A(x, y) \vee L_{\text{EU}}(x, y)$ is a more intuitive witness.

3.2 The case of CQs

For the case of CQs we can use the same intuitions, but we need to test for all $\vec{a} \in S^+$ at once, and for a witness query we need a single CQ that captures the union of all $query(\mathcal{D}, \vec{a})$. This is achieved by *synchronizing* all the $(\mathcal{D}, \vec{a})$ using the well-known *direct product* construction.

Direct product of structures The *direct product* of two n-ary tuples $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_n)$ is the n-ary tuple $\vec{a} \otimes \vec{b} = ((a_1, b_1), \ldots, (a_n, b_n))$. For two structures S_1 and S_2, their direct product is defined as

$$S_1 \otimes S_2 = \{P(\vec{a} \otimes \vec{b}) \mid P(\vec{a}) \in S_1 \text{ and } P(\vec{b}) \in S_2\}.$$

We let $(S_1, \vec{a}) \otimes (S_2, \vec{b}) = (S_1 \otimes S_2, \vec{a} \otimes \vec{b})$, and relying on the associativity of $\otimes$, we use $\Pi_{1 \leq i \leq m}(S_i, \vec{a_i})$ as a shorthand for $(S_1, \vec{a_1}) \otimes \cdots \otimes (S_m, \vec{a_m})$. We call the pair $\Pi_{1 \leq i \leq m}(S_i, \vec{a_i})$ *safe* if all the elements in the tuple $\vec{a_1} \otimes \cdots \otimes \vec{a_m}$ appear in the structure $S_1 \otimes \cdots \otimes S_m$.

The direct product synchronizes into a single structure all the $(\mathcal{D}, \vec{a})$ that we need to test for the QBE(CQ) problem and it is, in general, the minimal structure that does this. It is the least upper bound on the lattice of objects $(S, \vec{a})$ defined by the relation $\rightarrow$, that is:

- $\Pi_{1 \leq i \leq m}(S_i, \vec{a_i}) \rightarrow (S_i, \vec{a_i})$ for every $1 \leq i \leq m$, and
- if $(S, \vec{a}) \rightarrow (S_i, \vec{a_i})$ for every $1 \leq i \leq m$, then $(S, \vec{a}) \rightarrow \Pi_{1 \leq i \leq m}(S_i, \vec{a_i})$.

The direct product is used for the QBE(CQ) and QDEF(CQ) tests that characterize the QBE(CQ) and QDEF(CQ) problems. Note that we must check for safety of the product.

QBE(CQ) Test
INPUT: A dataset $\mathcal{D}$ and two sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$.
Is it the case that: - $\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a})$ is safe, and - $\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a}) \not\rightarrow (\mathcal{D}, \vec{b})$ for all $\vec{b} \in S^-$?

QDEF(CQ) Test
INPUT: A dataset $\mathcal{D}$ and a set S^+ of tuples from $\mathsf{I}(\mathcal{D})^n$.
Is it the case that: - $\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a})$ is safe, and - $\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a}) \not\rightarrow (\mathcal{D}, \vec{b})$ for all $\vec{b} \in \mathsf{I}(\mathcal{D})^n \setminus S^+$?

The canonical witness query for both problems is now $query(\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a}))$. Although the UCQ case and the CQ case are conceptually similar, there is a major difference between taking the union of all $(\mathcal{D}, \vec{a})$ as a UCQ, or taking their direct product as a CQ: the size is now $O(|\mathcal{D}|^{|S^+|})$ rather than $O(|\mathcal{D}| \times |S^+|)$. Since we have to test $\rightarrow$ with an exponentially larger structure, the upper bound jumps to coNEXPTIME. Unfortunately, both of these bounds are tight: the complexity of deciding QBE(CQ) is coNEXPTIME-hard, and there are instances where any witness query is of necessarily exponential size. This was first shown by Willard [21], but a more recent proof by ten Cate and Dalmau [19] shows that this holds even if we restrict the examples to unary relations, that is, to concepts only.

Example 3. (ctd.) With the same $\mathcal{D}_s$ and $S^+ = \{(s_1, p_1), (s_2, lit)\}$ as in Example 2, the product $S = (\mathcal{D}_s \otimes \mathcal{D}_s, (s_1 s_2, p_1 lit))$ is not safe. Indeed, $\mathcal{D}_s \otimes \mathcal{D}_s$ contains

$$R_A(s_1 s_1, p_1 p_1) \qquad R_B(s_2 s_2, p_2 p_2) \qquad L_{\text{EU}}(s_2 s_2, lit\,lit) \qquad L_{\text{nEU}}(s_1 s_1, tur\,tur)$$

and neither $(s_1 s_2)$ nor $(p_1 lit))$ occur in it. Hence there is no witness CQ.

4 Reverse Engineering OMQs

We now define variants of the $\mathcal{Q}$-QBE and $\mathcal{Q}$-QDEF problems for OMQs, where also an ontology $\mathcal{O}$ in some DL language $\mathcal{L}$ is given as an input.

PROBLEM:	**QBE($\mathcal{Q}$, $\mathcal{L}$)**
INPUT:	A dataset $\mathcal{D}$, an $\mathcal{L}$ ontology $\mathcal{O}$, and two sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$.
QUESTION:	Is there a query $(Q, \vec{x})$ in $\mathcal{Q}$ such that $S^+ \subseteq ans((Q, \vec{x}, \mathcal{O}), \mathcal{D})$ and $S^- \cap ans((Q, \vec{x}, \mathcal{O}), \mathcal{D}) = \emptyset$?

PROBLEM:	**QDEF($\mathcal{Q}$, $\mathcal{L}$)**
INPUT:	A dataset $\mathcal{D}$, an $\mathcal{L}$ ontology $\mathcal{O}$, and a set S^+ of tuples from $\mathsf{I}(\mathcal{D})^n$.
QUESTION:	Is there a query $(Q, \vec{x})$ in $\mathcal{Q}$ such that $S^+ = ans((Q, \vec{x}, \mathcal{O}), \mathcal{D})$?

These definitions are essentially the same as in [10], except that we omit the signature Σ, and allow all concepts and roles to occur in the query. In that work, the authors characterize the precise complexity of $\mathcal{Q}$-QBE and $\mathcal{Q}$-QDEF problems for the two DLs called Horn-$\mathcal{ALC}$ and Horn-$\mathcal{ALCI}$ [10]. The latter is a strict extension of *DL-Lite*$_\mathcal{R}$, while the former is orthogonal to it. Their results (without restricted signatures) are summarized in the following theorem.

Theorem 2. *[10] The following hold:*

1. QBE*(CQ,Horn-$\mathcal{ALC}$) and* QDEF*(CQ,Horn-$\mathcal{ALC}$) are* coNEXPTIME*-complete.*
2. QBE*(UCQ,Horn-$\mathcal{ALC}$) and* QDEF*(UCQ,Horn-$\mathcal{ALC}$) are* EXPTIME*-complete.*
3. QBE*(CQ,Horn-$\mathcal{ALCI}$) and* QDEF*(CQ,Horn-$\mathcal{ALCI}$) are* 2EXPTIME*-complete.*
4. QBE*(UCQ,Horn-$\mathcal{ALCI}$) and* QDEF*(UCQ,Horn-$\mathcal{ALCI}$) are* EXPTIME*-complete.*

The case where $\mathcal{L}$ is *DL-Lite*$_\mathcal{R}$ has not been studied yet. From these results, since *DL-Lite*$_\mathcal{R}$ supports inverse roles r^- and Horn-$\mathcal{ALC}$ does not, we can only transfer the upper bounds for the third and fourth items. These together with the results of Section 3 give us:

Corollary 1. The following hold:

1. QBE(CQ,*DL-Lite*$_\mathcal{R}$) and QDEF(CQ,*DL-Lite*$_\mathcal{R}$) are coNEXPTIME-hard, and in 2EXPTIME.
2. QBE(UCQ,*DL-Lite*$_\mathcal{R}$) and QDEF(UCQ,*DL-Lite*$_\mathcal{R}$) are coNP-hard and in EXPTIME.

Our main contribution is to close the second gap, obtaining the best possible bound:

Theorem 3. QBE*(UCQ,DL-Lite*$_\mathcal{R}$*) and* QDEF*(UCQ,DL-Lite*$_\mathcal{R}$*) are* coNP*-complete*

We prove Theorem 3 in the rest of this section. For this, we rely on some of the results of [10], and on well-known properties of *DL-Lite*$_\mathcal{R}$ [8]. We focus on the case of QBE; the case of QDEF is proved similarly by simply taking the corresponding set of $\vec{b}$-tuples.

In what follows, we restrict our attention to instances of QBE($\mathcal{Q}$,$\mathcal{L}$) and QDEF($\mathcal{Q}$,$\mathcal{L}$) for which $(\mathcal{O}, \mathcal{D})$ is consistent and $S^+ \neq \emptyset$; both problems trivialize if any of those two conditions fails, see [10]. We remark that for *DL-Lite*$_\mathcal{R}$ ontologies, testing if $(\mathcal{O}, \mathcal{D})$ is consistent can be done in NLOGSPACE with existing algorithms.

Universal Model We start by recalling the notion of *universal models*. For *DL-Lite*$_\mathcal{R}$ and other so-called *Horn* DLs, we can build a *universal model* for any consistent $\mathcal{O}$ and $\mathcal{D}$. This model can be homomorphically embedded into any other model of $\mathcal{O}, \mathcal{D}$, and is therefore sufficient for answering any query that is preserved under homomorphisms, as is the case for CQs and UCQs. The following result is well-known in the OMQ literature.

Claim 1. *[8] For every consistent pair $(\mathcal{O}, \mathcal{D})$ with $\mathcal{O}$ in DL-Lite*$_\mathcal{R}$*, there exists an interpretation $\mathcal{U}_{\mathcal{O},\mathcal{D}}$ such that $\mathcal{U}_{\mathcal{O},\mathcal{D}} \models \mathcal{O}, \mathcal{D}$, and $\mathcal{I} \rightarrow \mathcal{U}_{\mathcal{O},\mathcal{D}}$ for every model $\mathcal{I}$ of $\mathcal{O}, \mathcal{D}$.*

Adapting the characterization The QBE and QDEF tests in Section 3 cannot be used directly in the presence of ontologies. For instance, it fails for the example given in the introduction: the product $\Pi_{\vec{a} \in S^+}(\mathcal{D}, \vec{a})$ is not safe ((p_2, p_3, p_5) does not occur in $\mathcal{D} \otimes \mathcal{D} \otimes \mathcal{D}$), but a CQ does exist. This is not surprising: the characterizations in the previous section does not take into account the ontology, which can of course have an effect on QBE. Therefore, we have an alternative characterization that, in a nutshell, replaces $\mathcal{D}$ by the universal model $\mathcal{U}_{\mathcal{O},\mathcal{D}}$.

QBE(UCQ,$\mathcal{L}$) Test
INPUT: A dataset $\mathcal{D}$, an $\mathcal{L}$ ontology $\mathcal{O}$, and two sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$.

Is it the case that $(\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{a}) \not\rightarrow (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$ for all $\vec{a} \in S^+$ and all $\vec{b} \in S^-$?

Since Horn-$\mathcal{ALCI}$ contains *DL-Lite*$_\mathcal{R}$, we can use the following result, shown in [10] for the DLs Horn-$\mathcal{ALC}$ and Horn-$\mathcal{ALCI}$:

Proposition 1. *An instance $\mathcal{D}, \mathcal{O}, S^+, S^-$ of the* QBE*(UCQ,DL-Lite*$_\mathcal{R}$*) problem is positive iff it is also a positive instance of the* QBE*(UCQ,DL-Lite*$_\mathcal{R}$*) Test.*

However, unlike the case of standard databases, this test does not result in an immediate decision procedure. Indeed, for most ontology languages, and in particular for *DL-Lite*$_\mathcal{R}$, $\mathcal{U}_{\mathcal{O},\mathcal{D}}$ can be infinite. The authors of [10] provide sophisticated techniques to test for existence of homomorphisms between possibly infinite universal models, taking into account possibly restricted signatures. For the case of UCQs and no signature restriction, however, they make a key observation that will be useful for us, namely:

Lemma 1. *[10] For every dataset $\mathcal{D}$, DL-Lite*$_\mathcal{R}$ *ontology $\mathcal{O}$, and n-ary tuples $\vec{a}, \vec{b} \in \mathsf{I}(\mathcal{D})^n$,*

$$(\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{a}) \rightarrow (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b}) \quad \textit{iff} \quad (\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}) \rightarrow (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$$

where $\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}$ is the restriction of $\mathcal{U}_{\mathcal{O},\mathcal{D}}$ to have as domain the individuals in $\mathcal{O}$ only.

This is useful, since now the source for testing homomorphism existence is a small dataset. The target structure, however, is still infinite. Here is where we make use of the *DL-Lite*$_\mathcal{R}$ machinery, and exploit *query rewriting*.

Query rewriting for *DL-Lite*$_\mathcal{R}$ *DL-Lite*$_\mathcal{R}$ was designed with the deliberate intention of supporting scalable OMQ answering leveraging existing database technologies. This goal is realized in the crucial property of *UCQ rewritability* enjoyed by OMQs that comprise a CQ and a *DL-Lite*$_\mathcal{R}$ ontology. In a nutshell, such an OMQ can be effectively translated into an equivalent UCQ that, without an ontology, gives the same answers over any dataset.

Proposition 2. *[8] Given an OMQ $(q, \vec{x}, \mathcal{O})$ where $(q, \vec{x})$ is a CQ and $\mathcal{O}$ a DL-Lite*$_\mathcal{R}$ *ontology, we can obtain a UCQ $rew(q, \vec{x}, \mathcal{O}) = (\{q_1, \ldots, q_n\}, \vec{x})$ such that for every dataset $\mathcal{D}$*

$$ans((q, \vec{x}, \mathcal{O}), \mathcal{D}) = ans(rew(q, \vec{x}, \mathcal{O}), \mathcal{D})$$

The number n of CQs in $rew(q, \vec{x}, \mathcal{O})$ may be exponential in the number of axioms in $\mathcal{O}$, but each q_i is polynomial in the combined sizes of q and $\mathcal{O}$, and can be computed with a non-deterministic algorithm in polynomial time.

To leverage this for QBE, we simple take $query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a})$, and the existence of homomorphisms reduces to testing if $\vec{b}$ is an answer to its rewriting. This reduces QBE to a few ordinary UCQ evaluations, yielding the desired decision procedure.

Proposition 3. *An input $(\mathcal{D}, \mathcal{O}, S^+, S^-)$ is a positive instance of* QBE*(UCQ,DL-Lite$_\mathcal{R}$) iff for every $\vec{a} \in S^+$ and every $\vec{b} \in S^-$ we have*

$$\vec{b} \notin ans(rew(q_a, \mathcal{O}), \mathcal{D}), \quad \text{where } q_a = query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}).$$

Proof. From Proposition 1 we have that $(\mathcal{D}, \mathcal{O}, S^+, S^-)$ is a positive instance of QBE(UCQ,*DL-Lite*$_\mathcal{R}$) iff $(\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{a}) \not\to (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$ for all $\vec{a} \in S^+$ and $\vec{b} \in S^-$. We also have, for all $\vec{a} \in S^+$ and $\vec{b} \in S^-$:

$(\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{a}) \to (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$	iff $(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}) \to (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$	by Lemma 1,
	iff $q_a \to (\mathcal{U}_{\mathcal{O},\mathcal{D}}, \vec{b})$	since $query(\cdot)$ preserves $\to$,
	iff $q_a \to (\mathcal{I}, \vec{b})$ for all $\mathcal{I} \models \mathcal{O}, \mathcal{D}$	by Claim 1,
	iff $\vec{b} \in ans((q_a, \mathcal{O}), \mathcal{D})$	by the semantics of OMQs,
	iff $\vec{b} \in ans(rew(q_a, \mathcal{O}), \mathcal{D})$	by Proposition 2.

Hence $(\mathcal{D}, \mathcal{O}, S^+, S^-)$ is a positive instance of QBE(UCQ,*DL-Lite*$_\mathcal{R}$) iff $\vec{b} \notin ans(rew(q_a, \mathcal{O}), \mathcal{D})$ for all $\vec{a} \in S^+$ and $\vec{b} \in S^-$. □

Now we are ready to conclude the proof of Theorem 3.

Proof of Theorem 3. The lower bound follows from Theorem 1 (see also Corollary 1). To show the upper bound for QBE, consider the following non-deterministic polynomial time algorithm:

(1) Guess some $\vec{a} \in S^+$ and some $\vec{b} \in S^-$.
(2) Build $\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}$.
(3) Take $q_a = query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a})$.
(4) Non-deterministically obtain some CQ $(q, \vec{x}) \in rew(q_a, \mathcal{O})$.
(5) Verify that $\vec{b} \in ans(q, \mathcal{D})$ by finding a homomorphism from $(q, \vec{x})$ into $(\mathcal{D}, \vec{b})$.

The correctness of the algorithm follows from Proposition 3. For the complexity bound, the only step we need to explain how to do in polynomial time is item (2), that is, building $\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}$. But for *DL-Lite*$_\mathcal{R}$ this is also easy. We simply take all the axioms $E_1 \sqsubseteq E_2$ in $\mathcal{O}$, and take let $\sqsubseteq^*_\mathcal{O}$ to be the transitive closure of this relation. Then add to $\mathcal{D}$ the necessary assertions to close it under the following rules:

$$\text{If } B_1(a) \in \mathcal{D} \text{ and } B_1 \sqsubseteq^*_\mathcal{O} B_2, \text{ then } B_2(a) \in \mathcal{D}.$$
$$\text{If } r_1(a,b) \in \mathcal{D} \text{ and } r_1 \sqsubseteq^*_\mathcal{O} r_2, \text{ then } r_2(a,b) \in \mathcal{D}.$$

It is easy to see that this can be done deterministically in polynomial time.

For QDEF, we use the same argument, but the only difference is that instead of guessing some $\vec{b} \in S^-$, the algorithm guesses some $\vec{b} \in \mathsf{I}(\mathcal{D})^n \setminus S^+$. □

Note that we do not only have a tight upper bound on the complexity, but in case of a positive answer, we also have a witness query:

$$\bigcup_{\vec{a} \in S^+} rew(query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}))$$

Note that the witness query can be constructed using standard techniques for reasoning in *DL-Lite*$_\mathcal{R}$, which may be useful towards a practicable implementation.

5 Ontology axioms for smaller queries

In the characterizations and results so far we have taken the ontology into account, but we have not really exploited it. Naturally, we have silently assumed the worst case: that the ontology does not help us much. However, this need not always be the case. Recall our example from the introduction, where we could use an ontology with axioms

$$\begin{array}{rcl} \mathsf{ReceivesChemo} & \sqsubseteq & \exists \mathsf{takesISMed} \\ \exists \mathsf{takesISMed}^- & \sqsubseteq & \mathsf{ISMed} \\ \mathsf{takesISMed} & \sqsubseteq & \mathsf{takesMed} \end{array} \qquad \begin{array}{rcl} \mathsf{IsImmunoDef} & \sqsubseteq & \mathsf{IsImmunoCompr} \\ \exists \mathsf{takesISMed} & \sqsubseteq & \mathsf{IsImmunoCompr} \end{array}$$

to separate $S^+ = \{p_2, p_3, p_4, p_5\}$ and $S^- = \{p_1\}$ with the very natural CQ

$$q(x) = \mathsf{IsImmunoCompr}(x).$$

In fact, the *DL-Lite*$_\mathcal{R}$ rewriting algorithm rewrites this CQ into the UCQ

$$q(x) = \mathsf{IsImmunoCompr}(x) \vee \mathsf{IsImmunoDef}(x) \vee \big(\exists y\ \mathsf{takesISMed}(x, y)\big) \vee \mathsf{RecievesChemo}(x)$$

which is not very different from the canonical witness UCQ $\bigcup_{\vec{a} \in S^+} rew(query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}))$ (except that it contains less unnecessary atoms). This suggests a natural question: when can we use the ontology, possibly by adding axioms, to obtain a small witness CQ? As we have discussed above, witness CQs are known to be of exponential size in general. However, a pair of a small CQ and a DL ontology could potentially express this CQ more succinctly, in a similar way as it succinctly represents the possibly exponentially larger UCQ that results from its rewriting in the case of *DL-Lite*$_\mathcal{R}$. It would be nice if we could *always* leverage an ontology for finding small CQs, but we conjecture this is not possible. And even if it was, in practice we may not always be allowed to modify the original ontology, or we may want to modify it in very limited ways to avoid affecting other aspects of the domain conceptualization. We therefore formulate the following decision problem, where we only allow to add axioms that to not modify any entailments over the original signature.

PROBLEM:	**succinct-OQbE($\mathcal{L}$)**
INPUT:	A dataset $\mathcal{D}$, two sets S^+, S^- from $\mathsf{I}(\mathcal{D})^n$, and a (possibly empty) $\mathcal{L}$ ontology $\mathcal{O}$ such that $(\mathcal{D}, S^+, S^-, \mathcal{O})$ is a positive instance of QBE(UCQ,$\mathcal{L}$).
QUESTION:	Are there a CQ $(q, \vec{x})$ and an $\mathcal{L}$ ontology $\mathcal{O}'$ such that: – $(q, \vec{x})$ is a witness for QBE(CQ,$\mathcal{L}$) on input $(\mathcal{D}, S^+, S^-, \mathcal{O} \cup \mathcal{O}')$, – the size of $(q, \vec{x}, \mathcal{O}')$ is polynomially bounded by $(\mathcal{D}, S^+, S^-, \mathcal{O})$, and – $\mathcal{O} \models \alpha$ iff $\mathcal{O} \cup \mathcal{O}' \models \alpha$ for every assertion α whose predicate occurs in $\mathcal{O}$?

When the input ontology is empty we are in the case where we want to add an ontology to transform a positive instance of QBE(UCQ) into a positive instance of QBE(CQ,$\mathcal{L}$), as we did in the example above. QDEF can also be generalized similarly. We expect these problems to be highly non-trivial for all standard DLs, and leave them for future research.

A family of positive instances

As the focus of this work is on feasible and useful cases, we finish by identifying a simple family of positive instances of succinct-OQbE(*DL-Lite*$_\mathcal{R}$) that covers our motivating example. Before giving the characterization, we remark that it is not insensitive to variable renamings, and some instances not satisfying the conditions below may be transformable into instances that do by renaming variables.

However, this is not critical. On the one hand, the characterization is not intended to be complete: the conditions are sufficient, not necessary. On the other hand, our construction relies on the witness $rew(query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}))$ which can be assumed to have analogous variable occurrences over the CQs by using variables canonically during rewriting.

Assume that we are given a dataset $\mathcal{D}$, sets S^+ and S^- of tuples from $\mathsf{I}(\mathcal{D})^n$, and a *DL-Lite*$_\mathcal{R}$ ontology $\mathcal{O}$, so that $(\mathcal{D}, S^+, S^-, \mathcal{O})$ is a positive instance of QBE(UCQ,*DL-Lite*$_\mathcal{R}$). Let $(Q, \vec{x}) = \bigcup_{\vec{a}\in S^+} rew(query(\mathcal{U}_{\mathcal{O},\mathcal{D}}|_{\mathsf{I}}, \vec{a}))$ with $\vec{x} = (x_1, \ldots x_n)$ be its canonical witness UCQ, and let $Q^\cap = \bigcap_{q_i \in Q} q_i$. Moreover, let the sets of atoms $q'_1, \ldots, q'_n$ denote the differences $q'_i = q_i \setminus Q^\cap$, and let $Q' = \bigcup_{q_i \in Q} q'_i$. For each variable x, we define

$$concepts(x) = \{A \mid A(x) \in Q'\} \cup \{\exists r \mid r(x, y) \in Q'\} \cup \{\exists r^- \mid r(y, x) \in Q'\}.$$

Then we call the instance $(\mathcal{D}, S^+, S^-, \mathcal{O})$ *DL-Lite*$_\mathcal{R}$ *abbreviatable* if the following hold:

1. Each variable in each q'_i occurs only once, that is, there are no joins in the differences q'_i.
2. For every individual a that occurs in the i-th position of some tuple in S^-, we have that $(\mathcal{O}, \mathcal{D}) \not\models B(a)$ for every $B \in concepts(x_i)$.

The *succinct OMCQ witness* of such an abbreviatable instance is $(q, \vec{x}, \mathcal{O} \cup \mathcal{O}_Q)$, where: *(i)* $\mathcal{O}_Q$ contains, for each x with $|concepts(x)| > 0$ and for some fresh concept A_x, all axioms $B \sqsubseteq A_x$ with $B \in concepts(x)$, and *(ii)* $q = Q^\cap \cup \{A_x(x) \mid |concepts(x)| > 0\}$.

The restricted shape of the axioms we added guarantees that $\mathcal{O}$ and $\mathcal{O} \cup \mathcal{O}_Q$ entail the same assertions over the signature of $\mathcal{O}$. It can be verified that every homomorphism from some $q_i \in Q$ into $\mathcal{U}_{\mathcal{O},\mathcal{D}}$ is still a homomorphism when we extend $\mathcal{U}_{\mathcal{O},\mathcal{D}}$ into $\mathcal{U}_{\mathcal{O}\cup\mathcal{O}_Q,\mathcal{D}}$, and we still have $(\mathcal{U}_{\mathcal{O}\cup\mathcal{O}_Q,\mathcal{D}}, \vec{a}) \nrightarrow (\mathcal{U}_{\mathcal{O}\cup\mathcal{O}_Q,\mathcal{D}}, \vec{b})$ for all $\vec{a} \in S^+$ and all $\vec{b} \in S^-$. Therefore, $(q, \vec{x})$ is a witness to the QBE(CQ,*DL-Lite*$_\mathcal{R}$) instance $(\mathcal{D}, S^+, S^-, \mathcal{O} \cup \mathcal{O}_Q)$ as desired, and its size is polynomial in $(\mathcal{D}, S^+, S^-, \mathcal{O})$.

6 Conclusions

In this paper, we have revisited the problem of reverse engineering queries in the presence of ontologies, with emphasis on showing that in the case of UCQs, the addition of a *DL-Lite*$_\mathcal{R}$ ontology does not increase the worst-case complexity. Our results are for the case where all concepts and roles can be used in the query. We mentioned that in [10], the authors consider both restricted and unrestricted signature. Both settings are important, but in this work choose to adopt the latter for two reasons. On the one hand, we are interested in keeping the complexity increase to a minimum, and in identifying manageable settings. Imposing additional restrictions can only make the problem harder [10]. On the other hand, we are interested in exploiting the vocabulary and knowledge of the ontology as much as possible, and use it to obtain smaller and more readable queries. Indeed, we may even prefer to *expand* rather than to *restrict* the signature in some cases, as we discussed in the last section, where we advocated for a more active role for the ontology and hinted that it could help us obtain smaller and more readable queries. We proposed a naïve algorithm for restricted instances that obtains small witness CQs by adding axioms to the ontology. Despite being simple, we hope it will help motivate the search for better and smarter algorithms to learn the query along with ontology axioms. We plan to explore this direction in future work, probably building on ideas that have been used for practical reverse engineering of queries in the database literature [11, 15, 9, 14, 20]. An interactive system where the user is shown candidate concepts or instances to be generalized, and proposes suitable axioms, may be particularly promising.

References

[1] Timos Antonopoulos, Frank Neven, and Frédéric Servais. Definability problems for graph query languages. In *Proc. Joint 2013 EDBT/ICDT Conferences, ICDT 2013*, pages 141–152. ACM, 2013.

[2] Marcelo Arenas, Gonzalo I. Diaz, and Egor V. Kostylev. Reverse engineering SPARQL queries. In *Proc. 25th International Conference on World Wide Web*, WWW '16, pages 239–249, 2016.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Pablo Barceló, Alexander Baumgartner, Victor Dalmau, and Benny Kimelfeld. Regularizing conjunctive features for classification. In *Proc. 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019*, pages 2–16. ACM, 2019.

[5] Pablo Barceló and Miguel Romero. The complexity of reverse engineering problems for conjunctive queries. In *Proc. 20th International Conference on Database Theory, ICDT 2017*, volume 68 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[6] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, pages 218–307. Springer, 2015.

[7] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. Learning path queries on graph databases. In *Proc. 18th International Conference on Extending Database Technology, EDBT 2015*, pages 109–120. OpenProceedings.org, 2015.

[8] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[9] Gonzalo I. Diaz, Marcelo Arenas, and Michael Benedikt. Sparqlbye: Querying RDF data by example. *Proc. VLDB Endowment*, 9(13):1533–1536, 2016.

[10] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. Reverse engineering queries in ontology-enriched systems: The case of expressive horn description logic ontologies. In *Proc. 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 1847–1853. ijcai.org, 2018.

[11] Dmitri V. Kalashnikov, Laks V.S. Lakshmanan, and Divesh Srivastava. Fastqre: Fast query reverse engineering. In *Proc. 2018 International Conference on Management of Data*, SIGMOD '18, pages 337–350, New York, NY, USA, 2018. ACM.

[12] Jörg-Uwe Kietz. Learnability of description logic programs. In *Proc. 12th International Conference on Inductive Logic Programming*, ILP'02, pages 117–132, Berlin, Heidelberg, 2003. Springer-Verlag.

[13] Benny Kimelfeld and Christopher Ré. A relational framework for classifier engineering. *SIGMOD Record*, 47(1):6–13, 2018.

[14] Hao Li, Chee-Yong Chan, and David Maier. Query from examples: An iterative, data-driven approach to query construction. *Proc. VLDB Endowment*, 8(13):2158–2169, September 2015.

[15] Denis Mayr Lima Martins. Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities. *Information Systems*, 83:89 – 100, 2019.

[16] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. New trends on exploratory methods for data analytics. In *Proc. VLDB Endowment*, volume 10, pages 1977–1980, 2017. Cited By :12.

[17] Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629 – 679, 1994. Special Issue: Ten Years of Logic Programming.

[18] Magdalena Ortiz and Mantas Simkus. Reasoning and query answering in description logics. In *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012*, volume 7487 of *Lecture Notes in Computer Science*, pages 1–53. Springer, 2012.

[19] Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In *Proc. 18th International Conference on Database Theory, ICDT 2015*, volume 31 of *LIPIcs*, pages 161–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[20] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. Query reverse engineering. *The VLDB Journal*, 23(5):721–746, October 2014.

[21] Ross Willard. Testing expressibility is hard. In *Proc. Principles and Practice of Constraint Programming - CP 2010*, volume 6308 of *Lecture Notes in Computer Science*, pages 9–23. Springer, 2010.

[22] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: A survey. In *Proc. 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 5511–5519. ijcai.org, 2018.

[23] Moshé M. Zloof. Query-by-example: The invocation and definition of tables and forms. In *Proc. 1st International Conference on Very Large Data Bases*, VLDB '75, pages 1–24. ACM, 1975.