Query Expressibility and Verification in Ontology-Based Data Access

Carsten Lutz, Johannes Marti, Leif Sabellek
Department of Computer Science, University of Bremen, Germany

Abstract

In ontology-based data access, multiple data sources are integrated using an ontology and mappings. In practice, this is often achieved by a bootstrapping process, that is, the ontology and mappings are first designed to support only the most important queries over the sources and then gradually extended to enable additional queries. In this paper, we study two reasoning problems that support such an approach. The expressibility problem asks whether a given source query $q_s$ is expressible as a target query (that is, over the ontology’s vocabulary) and the verification problem asks, additionally given a candidate target query $q_t$, whether $q_t$ expresses $q_s$. We consider UCQs as source and target queries and GAV mappings, showing that both problems are $\Pi^p_2$-complete in DL-Lite, $\text{coNP}\text{-}\text{complete}$ between $\mathcal{ELL}$ and $\mathcal{ECLH\ell}$ when source queries are rooted, and $2\text{ExpTime}$-complete for unrestricted source queries.

Introduction

Ontology-based data access (OBDA) (Poggi et al. 2008) is an instantiation of the classical data integration scenario, that is, a set of data sources is translated into a unifying global schema by means of mappings. The distinguishing feature of OBDA is that the global schema is formulated in terms of an ontology which provides a rich domain model and can be used to derive additional query answers via logical reasoning. When data sources are numerous such as in large enterprises, data integration is often a considerable investment. OBDA is no exception since the construction of both the mappings and the ontology is non-trivial and labour intensive.

In practice, OBDA is thus often approached in an incremental manner (Trisolini, Lenzerini, and Nardi 1999; Kharlamov et al. 2015; Sequeda and Miranker 2017). One starts with a small set of important source queries (typically hand crafted by experts from the enterprise’s IT department) and builds mappings for the involved sources and an initial ontology that support these queries, manually or with the help of extraction tools (Jiménez-Ruiz et al. 2015; Pinkel et al. 2018). The outcome of this first step is then evaluated and, when considered successful, ontology and mappings are extended to support additional queries. This process may proceed for several rounds and in fact forever since new data sources and queries tend to appear as the enterprise develops and existing data sources or the ontology need to be updated (Lembo et al. 2017).

The aim of this paper is to study two reasoning tasks that support such an incremental approach to OBDA. The expressibility problem asks whether a given source query $q_s$ is expressible as a target query $q_t$ over the global schema defined by the ontology. Possible reasons for non-expressibility include that the mappings do not transport all data required for answering $q_s$ to the global schema and that the ontology ‘blurs’ the distinction between different relations from the sources, examples are given in the paper. If $q_s$ is not expressible, one might thus decide to add more mappings or to rework the ontology. The verification problem asks, additionally given a candidate target query $q_t$, whether $q_t$ expresses $q_s$. This is useful for example when a complex $q_t$ has been manually constructed and when the ontology, mappings, or source schemas have been updated, with an unclear impact on $q_t$. The same problems have been considered in the context of open data publishing, there called finding and recognition of s-to-t rewritings (Cima 2017).

We consider UCQs (and sometimes CQs) both for source and target queries, global as view (GAV) mappings, and ontologies that are formulated in DL-Lite or in a description logic (DL) between $\mathcal{EL}$ and $\mathcal{ECL\ell}$, which are all very common choices in OBDA. It follows from results in (Nash, Segoufin, and Vianu 2010; Afrati 2011) that, even without ontologies, additional source UCQs become expressible when full first-order logic (FO) is admitted for the target query rather than only UCQs. In OBDA, however, going beyond UCQs quickly results in undecidability of query answering (Baader et al. 2017) and thus we stick with UCQs.

The expressibility problem in OBDA is closely related to the problem of query expressibility over views, which has been intensively studied in database theory, see for example (Levy et al. 1995; Duschka and Genesereth 1997; Calvanese et al. 2002; Nash, Segoufin, and Vianu 2010; Afrati 2011) and references therein. The problem has occasionally also been considered in a DL context (Calvanese, De Giacomo, and Lenzerini 2000; Haase and Motik 2005; Beeri, Levy, and Roussel 1997; Calvanese et al. 2012). These papers, however, study setups different from the one we consider, both regarding the rôle of the ontology and the
description logics used.

In many classical cases of query expressibility over views, informally stated, \( q_i \) is expressible over a set of mappings \( M \) (representing views) if and only if the UCQ \( M^{-1}(M(q_i)) \) is contained in \( q_i \), where \( M(q_i) \) is the UCQ obtained from \( q_i \) by applying the mappings and \( M^{-1}(M(q_i)) \) is obtained from \( M(q_i) \) by applying the mappings backwards (Nash, Segoufin, and Vianu 2010; Afrati 2011). Our starting point for proving decidability and upper complexity bounds for expressibility in OBDA is the observation that we need to check whether \( M^{-1}(q_i) \) is contained in \( q_i \), where \( q_i \) is a (potentially infinitary) UCQ-rewriting of the UCQ \( M(q_i) \) under the ontology; note that, here, we mean rewriting of an ontology-mediated query into a source query in the classical sense of ontology-mediated querying, see for example (Bienvenu et al. 2016). Verification can be characterized in a very similar way. These characterizations also show that expressibility can be reduced to verification in polynomial time and that if \( q_i \) is expressible, then it is expressed by the polynomial size UCQ \( M(q_i) \).

Our main results are that within the setup described above, expressibility and verification are \( \Pi_2 \)-complete in DL-Lite \( ^r \) and in many other dialects of DL-Lite, coNEXPTIME-complete in DLs between \( \text{EL} \) and \( \text{ELHI} \) when the source UCQ is rooted (that is, every variable is reachable from an answer variable in the query graph of every CQ), and 2EXPTime-complete in the unrestricted case. There are some surprises here. First, the \( \Pi_2 \) lower bound already applies when the ontology is empty and the source query is a CQ which means that, in the database theory setting, it is \( \Pi_2 \)-hard to decide the fundamental problem whether a source CQ is expressible as a (U)CQ over a set of UCQ views. For this problem, an NP upper bound was claimed without proof in (Levy et al. 1995). Our results show that the problem is actually \( \Pi_2 \)-complete. A second surprise is that 2EXPTime-respectively coNEXPTIME-hardness applies already in the case that the ontology is formulated in \( \mathcal{EL} \) (and when queries are UCQs). We are not aware of any other reasoning problem for \( \mathcal{EL} \) that has such a high complexity whereas there are several such problems known for \( \mathcal{ELC} \), that is, \( \mathcal{EL} \) extended with inverse roles (Bienvenu et al. 2016). There is a clear explanation, though: the mappings make it possible to introduce just enough inverse roles in the backwards translation \( \mathcal{M}^{-1} \) mentioned above so that hardness proofs can be made work.

Detailed proofs are deferred to the appendix that is included in the extended version of the paper available at http://www.informatik.uni-bremen.de/tkdi/research/papers.html.

### Preliminaries

We use a mix of standard DL notation (Baader et al. 2017) and standard notation from database theory.

**Databases and Queries.** A schema \( S \) is a set of relation names with associated arities. An \( S \)-database \( D \) is a set of facts \( R(a_1, \ldots, a_n) \) where \( R \in S \) is a relation name of arity \( n \) and \( a_1, \ldots, a_n \) are constants. We use \( \text{adom}(D) \) to denote the set of constants that occur in \( D \).

A conjunctive query (CQ) \( q(x) \) over schema \( S \) takes the form \( \exists y \varphi(x, y) \), where \( x \) are the answer variables, \( y \) are the quantified variables, and \( \varphi \) is a conjunction of relational atoms \( R(z_1, \ldots, z_n) \) and equality atoms \( z_1 = z_2 \) where \( R \in S \) is of arity \( n \) and \( z_1, \ldots, z_n \) are variables from \( x \cup y \). Contrary to the usual setup and to avoid dealing with special cases in some technical constructions, we do not require that all variables in \( x \) actually occur in \( \varphi(x, y) \). We sometimes confuse \( q \) with the set of atoms in \( \varphi \), writing for example \( R(x, y, z) \in q \). We use \( \var{q} \) to denote \( x \cup y \). The arity of a CQ is the number of variables in \( x \) and \( q \) is Boolean if it has arity zero. A homomorphism from \( q \) to a database \( D \) is a function \( h : \var{q} \rightarrow \text{adom}(D) \) such that \( R(h(x)) \in D \) for every relational atom \( R(x) \in q \) and \( h(x) = h(y) \) for every relational atom \( x = y \in q \). Note that \( h \) needs to be defined also for answer variables that do not occur in \( \varphi(x, y) \). A tuple \( a \in \text{adom}(D) \) is an answer to \( q \) on \( D \) if there is a homomorphism \( h \) from \( q \) to \( D \) with \( h(x) = a \). A union of conjunctive queries (UCQ) is a disjunction of CQs that all have the same answer variables. Answers to UCQs are defined in the expected way. We use \( \text{ans}_{D}(D) \) to denote the set of all answers to UCQ \( q \) on database \( D \).

Let \( q_1(x_1), q_2(x_2) \) be UCQs of the same arity and over the same schema \( S \). We say that \( q_1 \) is contained in \( q_2 \), denoted \( q_1 \subseteq_S q_2 \), if for every \( S \)-database \( D \), \( \text{ans}_{D}(q_1) \subseteq_S \text{ans}_{D}(q_2) \). It is well-known that, when \( q_1 \) and \( q_2 \) are CQs, then \( q_1 \subseteq_S q_2 \) iff there is a homomorphism from \( q_2 \) to \( q_1 \), that is, a function \( h : \var{q_2} \rightarrow \var{q_1} \) such that \( R(h(x)) \in q_1 \) for every relational atom \( R(x) \in q_2 \), \( (h(x), h(y)) \) in the equivalence relation generated by the equality atoms in \( q_1 \) for every \( x = y \in q_2 \), and \( h(x_2) = x_1 \). We indicate the existence of such a homomorphism with \( q_2 \rightarrow q_1 \). When \( q_1 \) is a UCQ with disjuncts \( q_1, \ldots, q_i, i \in \{1, 2\} \), then \( q_1 \subseteq_S q_2 \) iff for every \( q_{1,i} \), there is a \( q_{2,i} \) with \( q_{2,i} \rightarrow q_{1,i} \).

We shall frequently view UCQs as databases, for which we merely need to read variables as constants of the same name and drop equality atoms. Conversely, we shall also view a tuple \( (D, a) \) with \( D \) a database and \( a = a_1 \cdots a_n \in \text{adom}(D) \) as an \( n \)-ary CQ; note that repeated elements are admitted in \( a \). We do this by reserving \( n \) fresh answer variables \( x_1, \ldots, x_n \), viewing \( D \) as a CQ by reading all constants (including those in \( a \) as quantified variables, and adding the equality atom \( x_i = a_i \) for \( 1 \leq i \leq n \).

**Ontology-Based Data Access.** Let \( \mathbb{N}_C \) and \( \mathbb{N}_R \) be countably infinite sets of concept names and role names. An \( \mathcal{EL} \)-concept is formed according to the syntax rule

\[
C, D ::= T \mid A \mid C \cap D \mid \exists r.C \mid \exists r^{-}.C
\]

where \( A \) ranges over concept name and \( r \) over role names. An expression \( r^{-} \) is called an inverse role and a role is either a role name or an inverse role. As usual, we let \( (r^{-})^{-} \) denote \( r \). An \( \mathcal{EL} \)-ontology is a finite set of concept inclusions (CIs) \( C \subseteq D \), \( C, D \) \( \mathcal{EL} \)-concepts, and role inclusions \( r \subseteq s \) and \( r \subseteq s^{-} \). The semantics is defined in terms of interpretation \( I = (\Delta_I, -) \) as usual. An \( \mathcal{EL} \)-concept is an \( \mathcal{EL} \)-concept that does not use the constructor \( \exists r^{-}.C \). An \( \mathcal{EL} \)-ontology is an \( \mathcal{EL} \)-ontology that uses only \( \mathcal{EL} \)-concepts and contains no role inclusions. A basic concept is a concept name or of one of the forms \( T, \bot, \exists r.T, and \)
\( \exists \gamma \cdot \forall \cdot \). A DL-Lite\(_{\text{form}}\)-ontology is a finite set of statements of form

\[
B_1 \land \cdots \land B_n \subseteq B \quad \text{where } B_1, \ldots, B_n, B \text{ range over basic concepts and } r, s, r_1, \ldots, r_n \text{ range over role names.}
\]

A DL schema is a schema that uses only unary and binary relation names, which we identify with \( N_C \) and \( N_R \) respectively. An S-ABox is a database over DL schema \( S \). An interpretation \( \mathcal{I} \) is a model of an ABox \( \mathcal{A} \) if \( A(a) \subseteq \mathcal{A} \) implies \( a \in A^2 \) and \( r(a, b) \in \mathcal{A} \) implies \( (a, b) \in r^2 \). In contrast to standard DL terminology, we speak of constants and facts also in the context of ABoxes, instead of individuals and assertions.

An ontology-mediated query (OMQ) is a tuple \( Q = (O, S, q) \) with \( O \) an ontology, \( S \) a DL schema, and \( q \) a query such as a CQ. Let \( \mathcal{A} \) be an S-ABox. A tuple \( a \in \text{adm}(A) \) is a certain answer to \( Q \) on \( A \) if \( a \in \text{ans}(Q, \mathcal{I}) \) for every model \( \mathcal{I} \) of \( A \) (viewed as a potentially infinite S-database). We use \( \text{cert}(Q, \mathcal{A}) \) to denote the set of all certain answers to \( Q \) on \( A \) and sometimes write \( A \models \varphi(a) \) when \( a \in \text{cert}(Q, \mathcal{A}) \). For OMQs \( Q_1 = (O_1, \Sigma, q_1) \) and \( Q_2 = (O_2, \Sigma, q_2) \) of the same arity, we say that \( Q_1 \) is contained in \( Q_2 \), denoted \( Q_1 \subseteq Q_2 \), if for every \( \Sigma \)-ABox \( \mathcal{A} \), \( \text{cert}(Q_1, \mathcal{A}) \subseteq \text{cert}(Q_2, \mathcal{A}) \). Containment between an OMQ and a UCQ are defined in the expected way, and so is the converse containment.

A global as view (GAV) mapping over a schema \( S \) takes the form \( \varphi(x, y) \rightarrow \psi(x) \) where \( \varphi(x, y) \) is a conjunction of relational atoms over \( S \) and \( \psi(x) \) is of the form \( A(x) \), \( r(x, y) \), or \( r(x, x) \) with \( A \) a concept name and \( r \) a role name. We call \( \varphi(x, y) \) the body of the mapping and \( \psi(x) \) its head. Every variable that occurs in the head must also occur in the body. Let \( M \) be a set of GAV mappings over a schema \( S \). For every \( S \)-database \( D \), the mappings in \( M \) produce an ABox \( M(D) \), defined as follows:

\[
\{ R(a) \mid D \models \varphi(a, b) \text{ and } \varphi(x, y) \rightarrow R(x) \in M \}.
\]

This ABox can be physically materialized or left virtual; we do not make any assumptions regarding this issue.

An OBDA specification is a triple \( S = (O, M, S) \) where \( S \) is the source schema, \( M \) a finite set of mappings over \( S \), and \( O \) an ontology.\(^1\) We use \( \text{sch}(M) \) to denote the schema that consists of all relation names that occur in the heads of mappings in \( M \). Informally, \( S \) is addressing source data in schema \( S \); translated into an ABox in schema \( \text{sch}(M) \) in terms of the mappings from \( M \) and then evaluated under the ontology \( O \). Note that \( O \) can use the relation names in \( \text{sch}(M) \) as well as additional concept and role names, and so can queries that are posed against the ABox.

We use \( \mathcal{L}, \mathcal{M} \) to denote the set of all OBDA specifications \( (O, M, S) \) where \( O \) is formulated in the ontology language \( \mathcal{L} \) and all mappings in \( M \) are formulated in the mapping language \( \mathcal{M} \) and call \( \mathcal{L}, \mathcal{M} \) an OBDA language. An example of an OBDA language is \( \mathcal{ELHI}, \mathcal{GAV} \). In this paper, we shall concentrate on GAV mappings. While other types of mappings such as LAV and GLAV are also interesting (Poggi et al. 2008; Cima 2017), they are outside the scope of this paper.

**Definition 1.** Let \( Q_s \) and \( Q_t \) be query languages and \( \mathcal{L}, \mathcal{M} \) an OBDA language.

1. \( Q_s \)-to-\( Q_t \) verification problem in \( \mathcal{L}, \mathcal{M} \) is to decide, given an OBDA specification \( S = (O, M, S) \in \mathcal{L}, \mathcal{M} \), a source query \( q_s \in Q_s \), and a target query \( q_t \in Q_t \) of the same arity, whether \( q_t \) is a realization of \( q_s \) in \( S \), that is, whether \( \text{ans}_s(D) = \text{cert}(Q(M(D))) \) for all \( S \)-databases \( D \), where \( Q = (O, \text{sch}(M), q_t) \).

2. The \( Q_s \)-to-\( Q_t \) expressibility problem in \( \mathcal{L}, \mathcal{M} \) is to decide, given an OBDA specification \( S = (O, M, S) \in \mathcal{L}, \mathcal{M} \) and a source query \( q_s \in Q_s \), whether there is a realization \( q_t \) of \( q_s \) in \( Q_t \). We then say that \( q_s \) is \( Q_t \)-expressible in \( S \).

Note that an alternative definition is obtained by quantifying only over those \( S \)-databases \( D \) such that \( D \cup O \) is satisfiable. This does not make a difference for most of the setups studied in this paper since the involved DLs cannot express inconsistency.

**Example 2.** Assume that \( S \) contains a binary relation \( \text{Man}(m, d) \) meaning that department \( d \) is managed by manager \( m \) and a ternary relation \( \text{Emp}(e, d, o) \) meaning that employee \( e \) works for department \( d \) in office \( o \). Let \( M \) contain the GAV mappings

\[
\begin{align*}
\text{Man}(x, z) \land \text{Emp}(y, z, u) & \rightarrow \text{manages}(x, y) \\
\text{Emp}(x, y, z) & \rightarrow \text{Employee}(x)
\end{align*}
\]

Then the source query \( q_s(x) = \exists y \exists z \text{Man}(x, y) \) is not expressible because the mappings do not provide sufficient data from the source. It trivially becomes expressible as \( q_t(x) = \text{Manager}(x) \) when we add the mapping

\[
\text{Man}(x, y) \rightarrow \text{Manager}(x).
\]

Next, we further add the following \( \mathcal{EL} \)-ontology \( O \):

\[
\begin{align*}
\text{Manager} & \subseteq \text{Employee} \\
\text{Manager} & \subseteq \exists \text{manages.} \text{Secretary}
\end{align*}
\]

Then the source query \( q_s(x, y) = \exists z \exists m \text{Man}(x, y, z) \land \text{Emp}(y, z, u) \) is no longer expressible due to the first CI in \( O \). Informally, all the required data is there, but it is mixed with other data and we have no way to separate. The source query \( q_s(x, y) = \exists m \exists \text{Man}(x, z) \land \text{Emp}(y, z, u) \) is expressible as \( \text{manages}(x, y) \) despite the second CI in \( O \), intuitively because the additional data mixed into manages by that CI always involves an anonymous constant introduced through the existential quantifier and is thus never returned as a certain answer.

\(^1\)For readability, we consider a single data source, only. Multiple source databases can be represented as a single one by assuming that their schemas are disjoint and taking the union.
The size of any syntactic object $X$ such as a UCQ or an ontology, denoted $|X|$, is the number of symbols needed to write it, with names of concepts, roles, variables, etc. counting as one.

Characterizations

We characterize when a UCQ $q_1$ over $\text{sch}(M)$ is a realization of a UCQ $q_2$ over the source schema $S$ and then lift this characterization to the expressibility of $q_1$. This serves as a basis for deciding the expressibility and verification problems later on. The characterization applies the mappings from $M$ forwards and backwards, as also done in query rewriting under views (Nash, Segoufin, and Vianu 2010; Afrati 2011), and suitably mixes in UCQ-rewritings of certain emerging OMQs.

Let $S = (O, M, S)$ be an OBDA specification and $Q = (O, \text{sch}(M), q)$ an OMQ. A rewriting of $Q$ is a query $q_1$ over $\text{sch}(M)$ of the same arity as $q$ such that for all $\text{sch}(M)$-ABoxes $A$, $\text{ans}_{q_1}(A) = \text{cert}_{Q}(A)$. We speak of a UCQ rewriting if $q_1$ is a UCQ, of an infinitary UCQ rewriting if $q_1$ is a potentially infinite UCQ, and so on. Note that there always exists a canonical infinitary UCQ rewriting that is obtained by taking all $\text{sch}(M)$-ABoxes $A$ and answers $a \in \text{cert}_{Q}(A)$ and including $(A, a)$ viewed as a CQ as a disjoint. This even holds when $O$ is formulated in FO without equality. In fact, this follows from the definition of rewritings and the fact that OMQs with $O$ formulated in FO without equality are preserved under homomorphisms (Bienvenu et al. 2014).

Let $S = (O, M, S)$ be an OBDA specification and $A$ an ABox that uses only concept and role names from $\text{sch}(M)$. We say that a mapping $\varphi(x, y) \rightarrow \psi(x)$ from $M$ is suitable for a fact $\alpha \in A$ if $\psi(x)$ and $\alpha$ are unifiable. We write $M^-(\alpha)$ to denote the set of all $S$-databases obtained from $A$ as follows: for every fact $\alpha \in A$, choose a suitable mapping $\varphi(x, y) \rightarrow \psi(x)$ from $M$ and include $R(\sigma(x))$ in $M^-(\alpha)$ whenever $R(z)$ is an atom in $\varphi(x, y)$ and where $\sigma$ is the most general unifier of $\psi(x)$ and $\alpha$ extended to replace every variable from $y$ with a fresh constant. For example, for a fact $r(a, a) \in A$ we can choose a mapping $R(x, y, z) \rightarrow r(x, y)$ and include $R(a, a, b)$ in $M^-(\alpha)$, where $b$ is fresh. Both $M$ and $M^-$ lift to sets of databases and ABoxes as expected, that is, if $S$ is a set of $S$-databases, then $M(S) = \{M(D) \mid D \in S\}$ and if $S$ is a set of ABoxes over $\text{sch}(M)$, then $M^-(S) = \bigcup_{A \in S} M^-(A)$.

In what follows, we shall often apply a M to a CQ $q(x)$ viewed as a database, and view the result (which formally is a database) again as a CQ. In this case, the answer variables are again $x$ and the equality atoms from $q(x)$ are readded to $M(q)$. The same applies to UCQs and sets of databases, and to $M^-(q)$ (where we also preserve answer variables and readd equality atoms). Note that $M^-(q)$ gives a UCQ even when $q$ was a CQ.

Example 3. Consider the CQ $q(x, y, z) = \exists u r(x, y) \land s(x, z) \land s(z, u) \land x = y$ and let $M$ consist of the single mapping $r(x, y) \rightarrow r(x, y)$, that is, the role name $r$ is simply copied and the role name $s$ is dropped. Then $M(q)$ viewed as a CQ is $p(x, y, z) = r(x, y) \land x = y$. Note that the answer variable $z$ does not occur in an atom.

The following fundamental lemma describes the (non)-effect of applying $M$ and $M^-$ on query containment. It is explicit or implicit in many papers concerned with query rewriting under views or with query determinacy, see for example (Nash, Segoufin, and Vianu 2010; Afrati 2011).

**Lemma 4.** Let $M$ be a set of GAV mappings, $q_1$ and $q_2$ UCQs over $S$ and $r$, $r_1$ and $r_2$ UCQs over $\text{sch}(M)$, then:

1. If $q_1 \subseteq S q_2$, then $M(q_1) \subseteq \text{sch}(M) M(q_2)$.
2. If $r_1 \subseteq S \text{sch}(M) r_2$, then $M^-(r_1) \subseteq S M^-(r_2)$.
3. $q \subseteq S M^-(r)$ iff $M(q) \subseteq \text{sch}(M) r$.

The next theorem characterizes realizations in terms of UCQ rewritings and $M^-$. It can thus serve as a basis for deciding the verification problem.

**Theorem 5.** Let $S = (O, M, S)$ be an OBDA specification from $(\text{FO}(=), \text{GA V}), q_1$ a UCQ over $S$, $q_2$ a UCQ over $\text{sch}(M)$, and $q_1$ an infinitary UCQ rewriting of the OMQ $Q = (O, \text{sch}(M), q_1)$. Then $q_1$ is a realization of $q_2$ iff $q_2 \equiv S M^-(q_1)$.

**Proof.** Assume that $q_2 \equiv S M^-(q_1)$. We have to show that $q_1$ is a realization of $q_2$. Since $q_1$ is a rewriting of $Q$, it suffices to prove that $\text{ans}_{q_1}(D) = \text{ans}_{q_2}(M(D))$ for all $S$-databases $D$.

For $\subseteq$, assume that $a \in \text{ans}_{q_1}(D)$. Let $p$ be $(D, a)$ viewed as a CQ. From $a \in \text{ans}_{q_1}(D)$, we obtain $p \subseteq q_1$, and $q_1 \subseteq S M^- (q_2)$ yields $p \subseteq S M^- (q_2)$. With Point 3 of Lemma 4, it follows that $M(p) \subseteq q_1$, which by construction of $p$ implies $a \in \text{ans}_{q_2}(M(D))$.

For $\supseteq$, assume that $a \in \text{ans}_{q_2}(M(D))$. Let $p$ be $(M(D), a)$ viewed as a UCQ. Then, $p \subseteq \text{sch}(M) q_1$ and Point 3 of Lemma 4 yields $p' \subseteq S M^- (q_2)$, where $p'$ is $(D, a)$ viewed as a CQ. Together with $M^- (q_1) \subseteq q_1$, we obtain $p' \subseteq q_2$, which implies that $a \in \text{ans}_{q_1}(D)$.

The "only if" direction, assume that $q_1$ is a realization of $q_2$. We have to show that $q_2 \equiv S M^- (q_1)$. Thus, let $D$ be an $S$-database and $a$ a tuple from $\text{adm}(D)$ whose length matches the arity of $q_2$. Further, let $p$ be $(D, a)$ viewed as a database. Since $q_1$ is a realization of $q_2$ and $q_1$ a rewriting of the OMQ $Q$, $a \in \text{ans}_{q_1}$ (iff $a \in \text{ans}_{q_2}(M(D))$). The latter is the case iff $M(p) \subseteq \text{sch}(M) q_1$, which by Point 3 of Lemma 4 holds iff $p \subseteq S M^- (q_2)$. This in turn is the case if $a \in \text{ans}_{M^- (q_1)(D)}$.

The next theorem characterizes the expressibility of source queries in an OBDA specification. It has several interesting consequences. First, it implies that the UCQ $M(q_1)$ is a realization of a UCQ $q_2$ over $S$ if there is any such realization. This is well known in the case without an ontology (Nash, Segoufin, and Vianu 2010; Afrati 2011) and is implicit in (Cima 2017) for a rather special case of OBDA. Second, the theorem provides a polynomial time reduction of expressibility to verification: $q_1$ is expressible in $S$ iff $M(q_1)$ is a realization of $q_1$ in $S$. And third, it shows that if $q_2$ is a CQ, then CQ-expressibility coincides with UCQ-expressibility. Thus, all lower bounds for CQ-to-CQ expressibility also apply to (U)CQ-to-UCQ expressibility and all upper bounds
for UCQ-to-UCQ verification and expressibility also apply to the corresponding CQ-to-(U)CQ case.

**Theorem 6.** Let \( S = (O, M, S) \) be an OBDA specification from \((\text{FO}(=), \text{GAV})\), \( q_s \) a UCQ over \( S \), and \( q_r \) an infinitary UCQ rewriting of the OMQ \( Q = (O, \text{sch}(M), M(q_s)) \). Then \( q_s \) is UCQ-expressible in \( S \) iff \( M^-(q_r) \subseteq S \). Moreover, if this is the case then \( M(q_s) \) is a realization of \( q_s \) in \( S \).

**Proof.** We first observe that

(a) if \( M^-(q_r) \subseteq S \) then \( M(q_s) \) is a realization of \( q_s \) in \( S \).

This actually follows from Theorem 5 because \( q_s \subseteq S \) \( M^-(q_r) \) always holds. In fact, since \( M(q_s) \) is the actual query in \( Q \) and since \( q_r \) is a rewriting of \( Q \), we have \( M(q_s) \subseteq \text{sch}(M) q_r \); applying Point 3 of Lemma 4 then yields \( q_s \subseteq S \).

Note that (a) establishes the “if” part of Theorem 6. In view of Theorem 5 and by (a), we can prove both the “only if” and the “Moreover” part by showing that if there is any realization \( q_t \) of \( q_s \) in \( S \), then \( M(q_s) \) is a realization of \( q_s \).

Thus assume that \( q_s \) is such a realization and let \( Q' \) be the OMQ \((O, \text{sch}(M), q_r) \) and \( q_r' \) a UCQ-rewriting of \( Q' \). We aim to show that

(b) \( M^-(q_r) \subseteq S \).

This suffices since Theorem 5 yields \( M^-(q_r') \subseteq S \) as and composing (b) with this containment gives \( M^-(q_r') \subseteq S \), which yields the desired result because of (a).

To establish (b), by Point 2 of Lemma 4 it suffices to show \( q_r \subseteq \text{sch}(M) q_r' \). From Theorem 5, we get \( q_s \subseteq S \) \( q_r' \). Point 3 of Lemma 4 gives \( M(q_s) \subseteq \text{sch}(M) q_r' \). By the semantics of certain answers, this implies \((O, \text{sch}(M), M(q_s)) \subseteq \text{sch}(M) (O, \text{sch}(M), q_r') \). Since the former OMQ is just \( Q \) and \( q_r' \) is a rewriting of \( Q \), we get \( q_r \subseteq \text{sch}(M) (O, \text{sch}(M), q_r') \). It thus remains to show \((O, \text{sch}(M), q_r') \subseteq q_r' \), which is exactly the statement of Lemma 24 in the appendix.

The following corollary of Theorem 6 shows that while making the ontology logically stronger might make some source queries inexpressible (see Example 3), it never results in additional such queries becoming expressible.

**Corollary 7.** Let \( S_i = (O_i, M, S_i) \) \( i \in \{1, 2\} \) be OBDA specifications from \([\text{FO}, \text{GAV}]\) with \( O_1 \models O_2 \), \( Q \in \{\text{CQ}, \text{UCQ}\} \) and \( q_s \) from \( Q \). Then \( Q\)-expressibility of \( q_s \) in \( S_i \) implies \( Q\)-expressibility of \( q_s \) in \( S_1 \).

**Proof.** Assume that \( q_s \) is \( Q\)-expressible in \( S_1 \). Then Theorem 6 gives that \( M(q_s) \) is a realization, and this query is also from \( Q \). We show that \( M(q_s) \) is also a realization of \( q_s \) in \( S_2 \). Let \( q_r, i \) be the canonical infinitary UCQ rewriting of the OMQ \( Q_i = (O_i, \text{sch}(M), M(q_s)) \), \( i \in \{1, 2\} \). By Theorem 5, \( q_s \equiv S M^-(q_r,i) \). Since \( O_1 \models O_2 \), we have \( Q_2 \subseteq \text{sch}(M) Q_1 \). This clearly implies that every UCQ in \( Q_2 \) is also in \( Q_1 \). Thus \( q_s \equiv S M^-(q_r,1) \) implies \( q_s \equiv S M^-(q_r,2) \). It remains to argue that \( q_s \equiv S M^-(q_r,2) \). Since \( q_r,2 \) is a rewriting of \( Q_2 \), we have \( Q_2 \subseteq \text{sch}(M) q_r,2 \). By the semantics and definition of \( Q_2 \), \( M(q_s) \subseteq \text{sch}(M) Q_2 \) and thus \( M(q_s) \subseteq \text{sch}(M) q_r,2 \). Point 3 of Lemma 4 yields \( q_s \subseteq S M^-(q_r,2) \) as desired.

---

**Expressibility and Verification in DL-Lite**

We consider OBDA specifications in which the ontology is formulated in a dialect of DL-Lite. The distinguishing feature of logics from this family is that finite UCQ rewritings of OMQs always exist. Therefore, Theorems 5 and 6 immediately imply decidability of the verification and expressibility problem, respectively. It is, however, well known that UCQ rewritings can become exponential in size (Gottlob et al. 2014) and thus optimal complexity bounds are not immediate.

We consider the dialect DL-Lite\(^\text{horn} \) as a typical representative of the DL-Lite family of logics. However, our results also apply to many other dialects since their proof rests only on the following properties, established in (Artale et al. 2009).

**Theorem 8.** In DL-Lite\(^\text{horn} \):

1. all OMQs \( Q \) have a UCQ-rewriting in which all CQs are of size polynomial in \(|Q|\);
2. OMQ evaluation is in NP in combined complexity.

We remark that the results presented in this section are related to those obtained in (Cima 2017), where the DL-Lite\(_{A, id}\) dialect of DL-Lite is considered, mappings are GLAV, and queries CQs. A main difference is Cima’s technical results concern rewritings that are complete but not necessarily sound, which corresponds to replacing ‘\( \text{ans}_{q_s}(D) = \text{cert}_Q(M(D)) \)’ in Definition 1 with ‘\( \text{ans}_{q_s}(D) \subseteq \text{cert}_Q(M(D)) \)’. Some of his technical constructions are similar to ours. Note that DL-Lite\(_{A, id}\) also satisfies the conditions from Theorem 8 and thus our results apply to [DL-Lite\(_{A, id}\), GAV] as well.

For an OMQ \( Q = (O, S, q) \), with \( O \) formulated in FO(=) and \( q \) a UCQ, the canonical UCQ-rewriting of size \( n \) is the UCQ \( q_s \) that consists of all pairs \((A, a)\) viewed as a CQ where \( a \in \text{cert}_Q(A) \) and \(|A| \leq n \). The following lemma is interesting in connection with Point 1 of Theorem 8 as it allows us to concentrate on canonical UCQ rewritings of polynomial size.

**Lemma 9.** Let \( Q = (O, S, q) \) be an OMQ with \( O \) formulated in FO(=) and \( q \) a UCQ. If \( Q \) has a UCQ-rewriting \( q_r \) in which all CQs are of size at most \( n \), then the canonical UCQ-rewriting \( q_s \) of size \( n \) is also a rewriting of \( Q \).

We are now ready to establish the upper bound.

**Theorem 10.** In [DL-Lite\(^\text{horn} \), GAV], the UCQ-to-UCQ expressibility and verification problems are in \( \Pi^2_2 \).

**Proof.** As remarked before Theorem 6, expressibility polynomially reduces to verification and thus it suffices to consider the latter. Hence let the following be given: an OBDA specification \( S = (O, M, S) \) from [DL-Lite\(^\text{horn} \), GAV], a UCQ \( q_s \) over schema \( S \), and a UCQ \( q_t \) over the schema \( \text{sch}(M) \). Let \( n \) be the size of this input.

Let \( Q = (O, \text{sch}(M), q_t) \) and note that the size of \( Q \) is polynomial in \( n \). By Point 1 of Theorem 8, we can assume that \( Q \) has a UCQ-rewriting in which all CQs are of size \( P(n) \), \( P \) a polynomial. By Lemma 9, we can even assume that this rewriting is the canonical UCQ-rewriting \( q_s \) of size \( P(n) \). By Theorem 5, \( q_t \) is thus a realization of \( q_s \) if \( q_s \equiv S \).
to prove the following result.

First, consider the inclusion \( q_s \subseteq M^-(q_s) \). It holds iff for every \( q \) in \( q_s \), there is a \( p \) in \( M^-(q_s) \) and a homomorphism \( p \rightarrow q \). This condition can be checked even in NP: iterate over all CQs \( q \) in \( q_s \) (of which there are at most \( n \)), guess a disjunct \( p \) from \( M^-(q_s) \), and verify in NP that \( p \rightarrow q \).

To guess a \( p \) in \( M^-(q_s) \), it suffices to guess a pair \((A,a)\) in \( q_s \) and suitable mappings from \( M \) for every fact in \( A \), which determine \( p \). Then, \( p \) can be computed in polynomial time from \( A \) and these suitable mappings. We guess the pair \((A,a)\) from \( q_s \) by guessing an arbitrary ABox \( A \) of size at most \( P(n) \) and then verifying that \( a \in cert_Q(A) \). By Point 2 of Theorem 8 this verification is possible in NP.

We next consider the inclusion \( M^-(q_s) \subseteq q_s \). This holds iff for every \( p \) in \( M^-(q_s) \), there is a CQ \( q \) in \( q_s \) such that \( q \rightarrow p \). We can thus universally guess a \( p \) in \( M^-(q_s) \), then iterate over all CQs \( q \) in \( q_s \), and for each such \( q \) check in NP whether \( q \rightarrow p \). For universally guessing \( p \), we actually guess a CQ \( p \) of size at most \( P(n) \) and then verify that it is in \( M^-(q_s) \). It has already been argued above that this is possible in NP. Overall, we obtain a \( \Pi^P_2 \)-algorithm, as desired.

We next show that the expressibility problem in [DL-Lite\textsubscript{horn}, GAV] is \( \Pi^P_2 \)-hard, and thus the same holds for the verification problem. Interestingly, the lower bound already applies when the ontology is empty and the source query is a CQ. As noted in the introduction, this shows that expressibility of a source CQ as a (U)CQ over UCQ views is \( \Pi^P_2 \)-hard, and in fact it is \( \Pi^P_2 \)-complete by Theorem 10. This corrects a (very likely) erroneous statement of NP-completeness in (Levy et al. 1995).

**Theorem 11.** The CQ-to-CQ expressibility problem is \( \Pi^P_2 \)-hard for GAV mappings and the empty ontology.

The proof is by reduction of validity of \( \forall \exists \)-QBFs. By Theorem 6, expressibility in the absence of an ontology amounts to checking the containment \( M^-(M(q_s)) \subseteq q_s \), which is equivalent to the \( \forall \exists \)-statement that for all \( p \) in \( M^-(M(q_s)) \) there is a homomorphism \( q_s \rightarrow p \). Hence we encode a \( \forall \exists \)-quantified Boolean formula such that the outer universal quantifiers correspond to the different choices of mappings when taking a \( p \) in \( M^-(M(q_s)) \), whereas the inner existential quantifiers of the formulas correspond to homomorphisms \( q_s \rightarrow p \).

**Expressibility in \( \mathcal{ELHI} \): Upper Bound for Rooted Queries**

We show that the expressibility problem in \( \mathcal{ELHI}, GAV \) is in \( \text{coNEXPTIME} \) when the source query is a rooted UCQ. Here, a CQ \( q \) is rooted if every variable from \( q \) is reachable from an answer variable in the hypergraph \( H_q := (\text{var}(q), \{x_1, \ldots, x_n\} | R(x_1, \ldots, x_n) \in q) \) and a UCQ is rooted or an \( rUCQ \) if every CQ in it is rooted. In practice, many relevant queries are rooted. Our aim is thus to prove the following result.

**Theorem 12.** In \( \mathcal{ELHI}, GAV \), the \( rUCQ \)-to-UCQ expressibility problem is in \( \text{coNEXPTIME} \).

To prepare for lifting the result from expressibility to verification later, we actually establish a slightly more general result as needed. Note that the following implies Theorem 12 since, by Theorem 6, we can simply use \( M(q_s) \) for \( q_s \).

**Theorem 13.** Given an OBDA setting \( S = (\mathcal{O}, M, S) \) from \( \mathcal{ELHI}, GAV \), an \( rUCQ \) \( q_s \) over \( S \), and a UCQ \( q_t \) over \( sch(M) \), it is in \( \text{coNEXPTIME} \) to decide whether \( M^-(q_s) \subseteq q_s \), where \( q_s \) is an infinitary UCQ-rewriting of the OMQ \( Q = (\mathcal{O}, sch(M), q_s) \).

To prove Theorem 13, we now describe a \( \text{NEXPTIME} \) algorithm for deciding the complement of the problem described there: we want to check whether \( M^-(q_s) \nsubseteq q_s \), that is, whether there is a \( CQ \) \( p \) in the \( UCQ \) \( M^-(q_s) \) such that \( q \nrightarrow p \) for all CQs \( q \) in \( q_s \). Because all rewritings of \( Q \) are equivalent, it suffices to prove the theorem for any particular infinitary UCQ-rewriting \( q_s \) of \( Q \). We choose to work with the canonical one introduced at the beginning of the characterizations section. The algorithm is as follows:

1. Guess a \( \text{sch}(M)\)-ABox \( A \) such that \( |\text{adom}(A)| \leq |q_s| + |q_t| \cdot |\mathcal{O}|^{q_s+1} \) and a tuple \( a \in A \) of the same arity as \( q_t \).
2. Verify that \( a \in \text{cert}(A) \) to make sure that \( (A,a) \) viewed as a CQ is in the UCQ \( q_s \). This can be done by an algorithm that is exponential in \( |\mathcal{O}| \) and \( |q_t| \), but only polynomial in \( |A| \); see for example (Krishnadhi and Lutz 2007).
3. Hence, the overall running time is single exponential in the size of the original input.
4. Guess a disjunct \( p \) from the UCQ \( M^-(A,a) \) by guessing, for each fact \( \alpha \) in \( A \), a suitable mapping from \( M \). Note that both \( A \) and \( p \) are of single exponential size.
5. Verify that \( q \nrightarrow p \) for all CQs \( q \) in the \( rUCQ \) \( q_s \). This can be done in single exponential time using brute force.

This is clearly a \( \text{NEXPTIME} \) algorithm.

**Lemma 14.** The algorithm decides the complement of the problem in Theorem 13.

It is easy to verify the soundness part: a successful run of the algorithm identifies \( p \) as a CQ in \( M^-(q_s) \) such that for any disjunct \( q \) of \( q_s \), \( q \nrightarrow p \). Completeness is less obvious, mainly because of the magical size bound used in Step 1. We need some preliminaries.

An ABox \( A \) is tree-shaped if the undirected graph \( G_A = (\text{adom}(A), \{ (a,b) | r(a,b) \in A \}) \) is acyclic, connected and \( r(a,b) \in A \) implies that \( s(a,b) \notin A \) for all role names \( s \neq r \) and that \( s(b,a) \notin A \) for all role names \( s \). An ABox \( A \) is pseudo tree-shaped with core \( C \subseteq A \) if there is a tree-shaped ABox \( A_o \) with root \( a \) for every \( a \in \text{adom}(C) \), with mutually disjoint domains, such that \( A = C \cup \bigcup_{a \in \text{adom}(C)} A_o \). The outerdegree of \( A \) is the maximal outerdegree of the trees underlying any of the \( A_o \).

The following lemma is an adaptation of Proposition 23 in Appendix B of (Bienvenu et al. 2016):

**Lemma 15.** Consider an \( \mathcal{ELHI} \)-ontology \( \mathcal{O} \), an OMQ \( Q = (\mathcal{O}, S, q) \) with a UCQ, an \( S \)-ABox \( A \), and an answer \( a \in \text{cert}(A) \). Then there is a pseudo tree-shaped \( S \)-ABox \( A' \) and a tuple \( a' \) in the core of \( A' \) such that
1. the core of $\mathcal{A}'$ is not larger than $|q|$ and the outdegree of $\mathcal{A}$ is not larger than $|Q|$;
2. $a' \in \text{cert}_Q(A')$;
3. there is a homomorphism $h$ from $\mathcal{A}'$ to $\mathcal{A}$ with $h(a') = a$.

We are now ready for the completeness part of Lemma 14. Assume that there is a $\text{CQ}$ $p$ in $M^-(q_r)$ such that for any disjunct $q$ of $q_r$, $q \not\rightarrow p$. Since $q_r$ is the canonical infinitary UCQ-rewriting of $Q$, $p$ is of the form $M^-(\mathcal{A},a)$ where $\mathcal{A}$ is a $\text{sch}(\Gamma)$-ABox with $a \in \text{cert}_Q(A)$. Let $\mathcal{A}'$ be the pseudo tree-shaped ABox and $a'$ the answer whose existence is guaranteed by Lemma 15. Moreover, let $\mathcal{A}''$ be the result of removing from $\mathcal{A}'$ all facts that contain at least one constant whose distance from the core is larger than $|q_r|$ and adding for all constants $a$ of distance exactly $|q_r|$ from the core both $\mathcal{A}(a)$ for all concept names $A$ in $\text{sch}(\Gamma)$ and $r(a,a)$ for all role names $r$ in $\text{sch}(\Gamma)$.

We show in the appendix that $a' \in \text{cert}_Q(A'')$ and that there is a $\text{CQ}$ $p$ in $M^-(\mathcal{A}'',a')$ such that for any disjunct $q$ of $q_r$, $q \not\rightarrow p$ (this depends on $q_r$ being rooted). The former implies that $(\mathcal{A}'',a')$, seen as $\text{CQ}$, is a disjunct in $q_r$ and hence Step 2 of the algorithm succeeds. The latter guarantees that Step 4 succeeds. Moreover, $\mathcal{A}''$ satisfies the size bound given in Step 1 of the algorithm.

**Expressibility in $\mathcal{ELHI}$: Upper Bound for Unrestricted Queries**

We consider the UCQ-to-UCQ expressibility problem in $\mathcal{ELHI}$, $\text{GAV}$, that is, we drop the assumption from the previous section that the source query is rooted. This increases the complexity from CO$\text{NEXPTIME}$ to 2EXPTIME. Note that similar effects have been observed in the context of different reasoning problems in (Lutz 2008, Bienvenu et al. 2016). In this section, we show the upper bound.

**Theorem 16.** In $\mathcal{ELHI}$, $\text{GAV}$, the UCQ-to-UCQ expressibility problem is in 2EXPTIME.

As in the rooted case, we again prove a slightly more general result that can be reused when studying the verification problem. We can obtain Theorem 16 from the following by setting $q_r = M(q_r)$ and applying Theorem 6.

**Theorem 17.** Given an OBDA setting $\mathcal{S} = (\mathcal{O}, \mathcal{M}, S)$ from $\mathcal{ELHI}$, $\text{GAV}$ a UCQ $q_r$ over $\mathcal{S}$, and a UCQ $q_t$ over $\text{sch}(\Gamma)$, it is in 2EXPTIME to decide whether $M^-(q_r) \subseteq S q_t$, where $q_t$ is an infinitary UCQ-rewriting of the OMQ $Q = (\mathcal{O}, \text{sch}(\Gamma), q_t)$.

To prove Theorem 17, we start by choosing a suitable UCQ-rewriting $q_r$. Instead of working with the canonical infinitary UCQ-rewriting, here we prefer to use the UCQ that consists of all pairs $(\mathcal{A},a)$ viewed as a UCQ and where $\mathcal{A}$ is a pseudo tree-shaped $\text{sch}(\Gamma)$-ABox that satisfies $a \in \text{cert}_Q(A)$ and is of the dimensions stated in Lemma 15, that is, the core of $\mathcal{A}$ is not larger than $|q|$ and the outdegree of $\mathcal{A}$ is not larger than $|Q|$. Due to that lemma, $q_t$ clearly is an infinitary UCQ-rewriting of $Q$.

We give a decision procedure for the complement of the problem in Theorem 17. We thus have to decide whether there is a pseudo-tree shaped $\text{sch}(\Gamma)$-ABox $\mathcal{A}$ of the mentioned dimensions, an $a \in \text{cert}_Q(A)$, and a $\text{CQ}$ $p$ in the UCQ $M^-(\mathcal{A},a)$ such that $q \not\rightarrow p$ for all UCQ $q$ in $q_t$. This can be done by constructing a two-way alternating parity tree automaton (TW APA) $\mathcal{A}$ on finite trees that accepts exactly those trees that represent a triple $(\mathcal{A},a,p)$ with the components as described above, and then testing whether the language accepted by $\mathcal{A}$ is empty.

In the following, we detail this construction. We reuse some encodings and notation from a TW APA construction that is employed in (Bienvenu et al. 2016) to decide OMQ containment as this saves us from redoing certain routine work. A tree is a non-empty (and potentially infinite) set $T \subseteq \mathbb{N}^*$ closed under prefixes. We say that $T$ is $m$-ary if $T \subseteq \{1, \ldots, m\}^*$ and call the elements of $T$ the nodes of the tree and $\varepsilon$ its root. For an alphabet $\Gamma$, a $\Gamma$-labeled tree is a pair $(T,L)$ with $T$ a tree and $L : T \rightarrow \Gamma$ a node labeling function.

We encode triples $(\mathcal{A},a,p)$ as finite $(|\mathcal{O}| \cdot |q_t|)$-ary $\Sigma_{\mathcal{L}} \cup \Sigma_{\mathcal{N}}$-labeled trees, where $\Sigma_{\mathcal{L}}$ is the alphabet used for labeling the root node and $\Sigma_{\mathcal{N}}$ is for non-root nodes. These alphabets are different because the root of a tree represents the core part of a pseudo tree-shaped ABox whereas each non-root node represents a single constant of the ABox that is outside the core. Let $C_{\text{core}}$ be a fixed set of $|q_t|$ constants. Formally, the alphabet $\Sigma_{\mathcal{L}}$ is the set of all triples $(B,a,\mu)$ where $B$ is a $\text{sch}(\Gamma)$-ABox of size at most $|q_t|$ that uses only constants from $C_{\text{core}}$, $a$ is a tuple over $C_{\text{core}}$ whose length matches the arity of $q_t$, and $\mu$ associates every fact $\alpha$ in $B$ with a mapping $\mu(\alpha) \in \mathcal{M}$ that is suitable for $\alpha$. The alphabet $\Sigma_{\mathcal{N}}$ consists of all triples $(\Theta,\mathcal{M},\mu)$ where $\Theta \subseteq (\mathcal{N} \cap \text{sch}(\Gamma)) \cup \{r,r^{-}\}$, $r \in \mathcal{N}$, and $C_{\text{core}}$ contains exactly one (potentially inverse) role and at most one element of $C_{\text{core}}$. $\mathcal{M} \in \mathcal{M}$ is a mapping suitable for the fact $r(a,b)$ with $r$ the unique role name in $\Theta$, and $\mu$ assigns to each $A \in \Theta$ a mapping $\mu(A) \in \mathcal{M}$ suitable for the fact $A(a)$.

In the following, a labeled tree generally means a $(|\mathcal{O}| \cdot |q_t|)$-ary $\Sigma_{\mathcal{L}} \cup \Sigma_{\mathcal{N}}$-labeled tree. A labeled tree is proper if (i) the root node is labeled with a symbol from $\Sigma_{\mathcal{L}}$, (ii) each child of the root is labeled with a symbol from $\Sigma_{\mathcal{N}}$ that contains an element of $C_{\text{core}}$, (iii) every other non-root node is labeled with a symbol from $\Sigma_{\mathcal{N}}$ that contains no constant name, and (iv) every non-root node has at most $|\mathcal{O}|$ successors and (v) for every $a \in C_{\text{core}}$, the root node has at most $|\mathcal{O}|$ successors whose label includes $a$. A proper labeled tree $(T,L)$ with $L(\varepsilon) = (B,a,\mu)$ encodes the triple $(\mathcal{A},a,p)$ where $\mathcal{A}$ is the ABox

\begin{align*}
\mathcal{B} &\cup \{A(x) \mid A \in \Theta(x)\} \\
&\cup \{r(b,x) \mid \{b,r\} \subseteq \Theta(x)\} \cup \{r(x,b) \mid \{b,r^{-}\} \subseteq \Theta(x)\} \\
&\cup \{r(x,y) \mid r \in \Theta(y), y \text{ is a child of } x, \Theta(x) \subseteq \Sigma_{\mathcal{N}}\} \\
&\cup \{r(y,x) \mid r^{-} \in \Theta(y), y \text{ is a child of } x, \Theta(x) \subseteq \Sigma_{\mathcal{N}}\}. \\
\end{align*}

$\Theta(x)$ denoting $\Theta$ when $L(x) = (\Theta,\mathcal{M},\mu)$ (and undefined otherwise), and where $p$ is the $\text{CQ}$ from $M^-(\mathcal{A})$ that can be obtained by choosing for every fact in $\mathcal{A}$ the suitable mapping from $\mathcal{M}$ assigned to it by $L$.

The desired TW APA $\mathcal{A}$ is obtained as the intersection of two TWAPAs $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A}_1$ accepts exactly the proper labeled trees $(T,L)$ that encode a pair $(\mathcal{A},a,p)$ with

\footnote{Here, $a$ and $b$ are arbitrary but fixed constants.}
\( a \in \text{cert}_Q(A) \) and \( \mathfrak{A}_2 \) is obtained as the complement of an automaton \( \mathfrak{A}_1 \) that accepts a proper labeled tree \( (T, L) \) encoding a pair \((A, a, p)\) iff \( q \rightarrow p \) for some CQ \( q \in q_s \). In fact, the automaton \( \mathfrak{A}_1 \) is what we can reuse from (Bienvenu et al. 2016), see Point 1 in Proposition 13 there. The only difference is that our trees are decorated in a richer way, so in our case the TWAPA ignores the part of the labeling that is concerned with mappings from \( M \). The number of states of \( \mathfrak{A}_1 \) is single exponential in \(|q|\) and \(|O|\).

We now sketch the construction of the automaton \( \mathfrak{A}_2 \) for a single CQ \( q \) of \( q_s \) (the general case can be dealt with using union). Let \( q_1, \ldots, q_k \) be the maximal connected components of \( q \). We define automata \( \mathfrak{A}_{2,1}, \ldots, \mathfrak{A}_{2,k} \) where \( \mathfrak{A}_{2,i} \) accepts \( (T, L) \) encoding \( (A, a, p) \) iff \( q_i \rightarrow p \), and then intersect to obtain \( \mathfrak{A}_2 \). Let \( (T, L) \) be a proper labeled tree. A set \( T' \subseteq T \) is a subtree of \( T \) if for any \( s, t \in T' \), all nodes from \( T \) that are on the shortest (undirected) path from \( s \) to \( t \) are in \( T' \). We use \( (T', L) \) to denote the restriction of \( (T, L) \) to \( T' \) and \( M^{-1}(T', L) \) to denote the subquery of \( p \) which contains only the atoms in \( p \) that can be derived from the part of \( A \) generated by the subtree \( (T', L) \) of \( (T, L) \).

To define \( \mathfrak{A}_{2,i} \), let \( C \) denote the set of all labeled trees \( (T', L) \) of size at most \(|q_i|\) such that there is a proper labeled tree \( (T, L) \) and \( q_i \rightarrow M^{-1}(T', L) \) with a homomorphism that only needs to respect the answer variables from \( q \) that actually occur in \( q_i \) (if there are any, then \( T' \) must thus contain the root of \( T \)). The automaton \( \mathfrak{A}_{2,i} \) is then constructed such that it accepts a proper labeled tree \( (T', L) \) iff it contains a subtree from \( C \). It should be clear that such an automaton can be constructed using only single exponentially many states. Moreover, it can be verified that \( \mathfrak{A}_{2,i} \) accepts exactly the desired trees. We obtain an overall automaton with single exponentially many states which together with the EXPTime-complete emptiness problem of TWAPAs gives Theorem 17.

**Verification in \( \mathcal{ELHI} \): Upper Bounds**

We show that in \( [\mathcal{ELHI}, \mathsf{GAV}] \), the complexity of the verification problem is not higher than the complexity of the expressivity problem both in the rooted and in the general case.

**Theorem 18.** In \( [\mathcal{ELHI}, \mathsf{GAV}] \),

1. the rUCQ-to-UCQ verification problem can be decided in \text{ConEXPTime}.

2. the UCQ-to-UCQ verification problem can be decided in \text{2EXPTime}.

Recall the characterization of realizations from Theorem 5: a UCQ \( q_t \) is a realization of \( q_s \) iff \( q_s \equiv M^{-1}(q_t) \), where \( q_s \) is a rewriting of the OMQ \((O, \mathsf{sch}(M), q_s)\). The inclusion \( q_s \subseteq M^{-1}(q_t) \) is already treated by Theorems 13 and 17 and thus it remains to show that the converse inclusion can be decided in the relevant complexity class. We show that it actually is in \text{EXPTime} even in the unrooted case. We thus aim to prove the following.

**Theorem 19.** Given an OBDA setting \( S = (O, M, S) \) from \([\mathcal{ELHI}, \mathsf{GAV}]\), a UCQ \( q_s \) over \( S \), and a UCQ \( q_t \) over \( \mathsf{sch}(M) \), it is in \text{EXPTime} to decide whether \( q_s \subseteq M^{-1}(q_t) \), where \( q_t \) is an infinitary UCQ-rewriting of the OMQ \( Q = (O, \mathsf{sch}(M), q_s) \).

For what follows, it is convenient to assume that the ontology \( O \) is in \textit{normal form}, that is, all CIs in it are of one of the forms \( T \sqsubseteq A, A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B, A \sqsubseteq R.B, \) and \( \exists R.A \sqsubseteq B \) where \( A, B \) and all \( A_i \) range over concept names and \( r \) ranges over roles. It is well-known that every \( \mathcal{ELHI} \)-ontology \( O \) can be converted into an \( \mathcal{ELHI} \)-ontology \( O' \) in normal form in linear time such that \( O' \) is a conservative extension of \( O \) in the model-theoretic sense (Baader et al. 2017). It is easy to verify that for the verification problem, we can w.l.o.g. assume the involved ontology to be in normal form.

We again start by choosing a suitable concrete infinitary UCQ-rewriting to use for \( q_s \). As in the previous section, we would like to use CQs derived from pseudo tree-shaped ABoxes of certain dimension that entail an answer to the OMQ \( Q \), as sanctioned by Lemma 15. Here, however, we use a slight strengthening of that lemma where Condition 2 is replaced with the following strictly stronger Condition 2', where \( C' \) is the core of the pseudo tree-shaped ABox \( A' \):

\[ a' \in \text{cert}_O(C \cup \{ A(a) \mid A', O \models A(a), a \in \text{dom}(C') \}) \]

This condition essentially says that, in the universal model of \( A' \) and \( O \) (defined in the appendix), there is a homomorphism \( h \) from a CQ in the UCQ \( q \) in \( Q \) that only involves constants from the core and ‘anonymous subtrees’ (generated by existential quantifiers) below them. This is true only since \( O \) is assumed to be in normal form and it is a consequence of the proof of Lemma 15 where one chooses a homomorphism \( h \) from a CQ in \( q \) to the universal model of \( A \) and \( O \), selecting as the core \( C' \) of the pseudo-tree ABox \( A' \) to be constructed all constants \( a \) from \( A \) that are in the range of \( h \) or which root an anonymous subtree that contains an element in the range of \( h \), and then unraveling the rest of \( A \). Summing up, we thus use for \( q_s \) the set of all pairs \((A, a)\) viewed as a CQ where \( A \) is a pseudo tree-shaped \( \mathsf{sch}(M) \)-ABox with core \( C \) that satisfies

\[ a \in \text{cert}_Q(C \cup \{ A(a) \mid A, O \models A(a), a \in \text{dom}(C) \}) \]

and is of the dimensions stated in Lemma 15.

For deciding \( q_s \subseteq M^{-1}(q_t) \), we need to show that for every disjunct \( q \) in \( q_s \), there is a disjunct \( p \) in \( M^{-1}(q_t) \) such that \( p \rightarrow q \), which we can do this for every disjunct \( q \) in \( q_s \) separately. Hence let \( q \) be such a disjunct. To find a CQ \( p \) in \( M^{-1}(q_t) \) with \( p \rightarrow q \), we again aim to utilize TWAPAs. As in the previous section, let \( C_{\text{core}} \) be a fixed set of \(|q_s|\) constants. A \textit{homomorphism pattern} for \( q_t \) is a function \( \lambda \) that maps every variable \( y \) in \( q_t \) to a pair \((a, o) \in C_{\text{core}} \times \{ \text{core}, \text{subtree} \} \).

Informally, \( \lambda \) is an abstract description of a homomorphism \( h \) from \( q_t \) to the universal model of a pseudo-tree ABox \( A \) and \( O \) (assume that the core of \( A \) uses only constants from \( C_{\text{core}} \)) such that \( h(x) = a \) when \( \lambda(x) = (a, \text{core}) \) and \( h(x) \) is an element in the anonymous subtree below \( a \) when \( \lambda(x) = (a, \text{subtree}) \).

We build one TWAPA \( \mathfrak{A}^\lambda \) for every homomorphism pattern \( \lambda \) (there are single exponentially many). These TWAPAs again run on \((|O| \cdot |q_s|)\)-ary \( \Sigma_3 \cup \Sigma_N \)-labeled trees that encode a triple \((A, a, p)\), defined exactly as in the previous section and from now on are only referred to as labeled trees. We also use the same notion of properness as in the previous
there is a homomorphism \( h \) from \( q_1(x) \) to the universal model of \( \mathcal{A} \) and \( \mathcal{O} \) that satisfies \( h(x) = a \) and follows the homomorphism pattern \( \lambda \) and

2. \( p \rightarrow q \).

Note that it would be sufficient to demand in Point 1 that \( a \in \text{cert}_{\mathcal{O}}(\mathcal{A}) \) and recall that, in the previous section, we have reused an automaton from (Bienvenu et al. 2016) which checks exactly this condition. That automaton, however, has exponentially many states because it is built using a construction known under various names such as query splitting, forest decomposition, and squid decomposition. To attain an \( \text{XP} \)-upper bound, though, the automaton \( \mathfrak{A}^\lambda \) can only have polynomially many states. This is in fact the reason why we have the strengthened Condition 2 of Lemma 15.

We construct the automaton \( \mathfrak{A}^\lambda \) as the intersection of three automata \( \mathfrak{A}^\text{proper} \), \( \mathfrak{A}^1 \), and \( \mathfrak{A}^2 \), where \( \mathfrak{A}^\text{proper} \) accepts if the input tree is proper, \( \mathfrak{A}^1 \) accepts trees that encode a triple \( (\mathcal{A}, a, p) \) that satisfy Condition 1 for \( \lambda \) and \( \mathfrak{A}^2 \) accepts trees that encode a triple \( (\mathcal{A}, a, p) \) that satisfy Condition 2. It is easy to build the automaton \( \mathfrak{A}^\text{proper} \) and we leave out the details. The precise construction of \( \mathfrak{A}^1 \) and \( \mathfrak{A}^2 \) can be found in the appendix, only give a brief description here. Automaton \( \mathfrak{A}^\lambda \) works as follows: it accepts labeled trees \( (T, L) \) whose root node label \( (B, a, \mu) \) is such that the extension of the ABox \( B \) with certain facts of the form \( A(a) \) results in a universal model which admits a homomorphism following pattern \( \lambda \), and it then verifies that the facts \( A(a) \) used in the extension can be derived from the ABox encoded by \( (T, L) \). The automaton \( \mathfrak{A}^2 \) checks Condition 2 by traversing the input tree once from the root to the leaves and guessing the homomorphism from \( p \) to \( q \) along the way. Some care is required since \( p \) is represented only implicitly in the input.

Overall, we obtain single exponentially many automata with polynomially many states each and we answer ‘yes’ if any of the automata recognizes a non-empty language. This gives Theorem 19.

**Expressibility and Verification in \( \mathcal{EL} \): Lower Bounds**

We establish lower bounds that match the upper bounds obtained in the previous three sections and show that they even apply to \( [\mathcal{EL}, \mathcal{GAV}] \), that is, inverse roles are not required.

**Theorem 20.**

1. In \( [\mathcal{EL}, \mathcal{GAV}] \), the \( \text{rUCQ-to-UCQ} \) expressibility and verification problems are \( \text{XP} \)-hard.
2. In \( [\mathcal{EL}, \mathcal{GAV}] \), the \( \text{UCQ-to-UCQ} \) expressibility and verification problems are \( \text{2XP} \)-hard.

By Theorem 6, it suffices to establish the lower bounds for the expressibility problem. We prove both points of Theorem 20 by a reduction from certain OMQ containment problems. For Point 1, we reduce from the following problem.

**Theorem 21.** (Bienvenu et al. 2016) Containment between OMQs \( Q_1 = (O, \Sigma, q_1) \) and \( Q_2 = (O, \Sigma, q_2) \) with \( O \) an \( \mathcal{ELI} \)-ontology. \( q_1 \) an AQ, and \( q_2 \) a rooted UCQ is \( \text{EXP} \)-hard even when

1. \( q_2(x) \) uses only symbols from \( \Sigma \) and
2. no symbol from \( \Sigma \) occurs on the right-hand side of a CI in \( O \).

We first establish Point 1 of Theorem 20 for \( [\mathcal{ELI}, \mathcal{GAV}] \) instead of for \( [\mathcal{EL}, \mathcal{GAV}] \) and in a second step show how to get rid of inverse roles. To reduce the containment problem in Theorem 21 to \( \text{rUCQ-to-UCQ} \) expressibility in \( [\mathcal{ELI}, \mathcal{GAV}] \), let \( Q_1 = (O, \Sigma, A_0(x)) \) and \( Q_2 = (O, \Sigma, q) \) be as in that theorem. We define an \( \text{OBDA} \)-specification \( S = (O', M, S) \) and a query \( q_s \) over \( S \) as follows. Let \( B \) be a concept name that does not occur in \( Q_1 \) and \( Q_2 \). Set

\[
\begin{align*}
O' &= O \cup \{A_0 \subseteq B\} \\
S &= \Sigma \cup \{B\} \\
qu_s(x) &= B(x) \lor q(x)
\end{align*}
\]

Note that \( q_s \) is a rooted UCQ, as required. Moreover, the set \( M \) of mappings contains \( A(x) \rightarrow A(x) \) for all concept names \( A \in S \) and \( r(x, y) \rightarrow r(x, y) \) for all role names \( r \in S \). In that case, the CI \( A_0 \subseteq B \) ‘pollutes’ \( B \), potentially preventing the disjunction \( B(x) \lor q_s(x) \) to be expressible, but this is not a problem if (and only if) \( Q_1 \not\subseteq Q_2 \).

**Lemma 22.** \( Q_1 \not\subseteq Q_2 \) iff \( q_s \) is UCQ-expressible in \( S \).

In short, \( Q_1 \not\subseteq Q_2 \) iff there is a tree-shaped \( \Sigma \)-ABox witnessing this iff such an ABox, viewed as a CQ, is a disjunction of an infinitary UCQ-rewriting \( q_r \) of the OMQ \( Q = (O', S, q_s) \) iff \( q_r \not\subseteq q_s \). The latter is the case iff \( q_s \) is not UCQ-expressible in \( S \) by Lemma 6 and since \( M(q_s) = q_s \) and \( M^{-1}(q_r) = q_r \).

In the appendix, we describe how to replace the \( \mathcal{ELI} \)-ontology \( O \) with an \( \mathcal{EL} \)-ontology. The crucial observation is that the hardness proof from (Bienvenu et al. 2016) uses only a single symmetric role \( S \) implemented as a composition \( r_0 : r_0 \) with \( r_0 \) a normal role name, and that it is possible to replace this composition with a normal role name \( r \) in \( O \) when reintroducing it in \( M^{-1}(q_r) \) via mappings \( r_0(x, y) \land r_0(y, z) \rightarrow r(x, z) \) where \( q_r \) is an infinitary UCQ-rewriting of the OMQ \( Q \) mentioned above.

The \( \text{2XP} \)-lower bound in Point 2 of Theorem 20 is proved similarly, using \( \text{2XP} \)-hardness of a different containment problem also studied in (Bienvenu et al. 2016).

**Conclusion**

We believe that several interesting questions remain. For example, our lower bounds only apply when the source query is a UCQ and it would be interesting to see whether the complexity drops when source queries are CQs. It would also be interesting to consider ontologies formulated in more expressive DLs such as \( \mathcal{ALC} \). As a first observation in this direction, we note the following undecidability result, where \( \mathcal{ALCF} \) is \( \mathcal{ALC} \) extended with (globally) functional roles. It is proved by a reduction from the emptiness of AQs w.r.t. \( \mathcal{ALCF} \)-ontologies (Baader et al. 2016).
Theorem 23. In $[\text{ALCF}, \text{GAV}]$, the AQ-to-Q expressibility and verification problems are undecidable for any $\mathcal{Q} \in \{\text{AQ, CQ, UCQ}\}$.

Regarding the expressibility problem, we note that the realization $\mathcal{M}(q_s)$ identified by Theorem 6 does not use any symbols introduced by the ontology and, in fact, is also a realization regarding the empty ontology. It would be interesting to understand how to obtain realizations that make better use of the ontology and to study setups where it can be unavoidable to exploit the ontology in realizations. This is the case, for example, when source queries are formulated in Datalog, the ontology is formulated in (some extension of) $\mathcal{EL}$, and target queries are UCQs. Finally, we note that it would be natural to study maximally contained realizations instead of exact ones and to take into account constraints over the source databases.

Acknowledgments

We thank the anonymous reviewers for their helpful remarks. This research was supported by ERC consolidator grant 647289 CODA.

References

Afrati, F. N. 2011. Determinacy and query rewriting for conjunctive queries and views. Theor. Comput. Sci. 412(11):1005–1021.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zhakharyaschev, M. 2009. The dl-lite family and relations. J. Artif. Intell. Res. 36:1–69.

Baader, F.; Bienvenu, M.; Lutz, C.; and Wolter, F. 2016. Query and predicate emptiness in ontology-based data access. J. Artif. Intell. Res. 56:1–59.

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

Beeri, C.; Levy, A. Y.; and Rouset, M. 1997. Rewriting queries using views in description logics. In Proc. of PODS, 99–108. ACM Press.

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.

Bienvenu, M.; Hansen, P.; Lutz, C.; and Wolter, F. 2016. First order-rewritability and containment of conjunctive queries in Horn description logics. In Proc. of IJCAI2016, 965–971.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2002. Lossless regular views. In Proc. of PODS, 247–258. ACM.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2012. View-based query answering in description logics: Semantics and complexity. J. Comput. Syst. Sci. 78(1):26–46.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2000. Answering queries using views over description logics.