

On conjunctive query answering in \mathcal{EL}

Riccardo Rosati

Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
Via Salaria 113, 00198 Roma, Italy
rosati@dis.uniroma1.it

1 Introduction

In this paper we study conjunctive query answering in the description logics of the \mathcal{EL} family [2, 3, 7, 6, 5], in particular we consider the DLs \mathcal{EL} , \mathcal{ELH} , \mathcal{EL}^+ , and \mathcal{EL}^{++} . The \mathcal{EL} family has been recently defined in order to identify DLs both having interesting expressive abilities and allowing for tractable reasoning. While the standard reasoning tasks (like concept subsumption and instance checking) have been analyzed in the past for such logics, almost no result is known about answering conjunctive queries in the logics of the \mathcal{EL} family, with the exception of the lower complexity bounds which are immediate consequence of the results in [8] (and of the characterization of instance checking in [7, 3]).

More specifically, we present the following results:

1. we define a query-rewriting-based technique for answering unions of conjunctive queries in \mathcal{EL} . More precisely, we present an algorithm based on the idea of reducing query answering in \mathcal{EL} to answering recursive Datalog queries. We also show that this technique can be easily extended to deal with \mathcal{ELH} KBs;
2. based on the above technique, we prove that answering unions of conjunctive queries in \mathcal{EL} and \mathcal{ELH} is PTIME-complete with respect to both data complexity (i.e., with respect to the size of the ABox) and knowledge base complexity (i.e., with respect to the size of the knowledge base) and is NP-complete with respect to combined complexity (i.e., with respect to the size of both the knowledge base and the query);
3. conversely, we prove that answering conjunctive queries is undecidable in both \mathcal{EL}^+ and \mathcal{EL}^{++} .

As an immediate consequence of the above results, it turns out that if, besides the standard reasoning tasks in DL, also conjunctive query answering is of interest, then \mathcal{EL} and \mathcal{ELH} still exhibit a nice computational behaviour, since they allow for tractable query answering, while the two extensions \mathcal{EL}^+ and \mathcal{EL}^{++} do not show the same behaviour, since conjunctive query answering is undecidable in such DLs. Consequently, \mathcal{EL}^+ and \mathcal{EL}^{++} do not appear well-suited for applications requiring the full power of conjunctive queries.

2 Preliminaries

In this section we briefly recall the logics in the \mathcal{EL} family, in particular the DLs \mathcal{EL} , \mathcal{EL}^+ , and \mathcal{EL}^{++} , and introduce query answering over knowledge bases expressed in such description logics.

\mathcal{EL} and its extensions. \mathcal{EL} [2] is the DL whose abstract syntax for concept expressions is the following: $C ::= A \mid \exists R.C \mid C_1 \sqcap C_2 \mid \top$, where A is a concept name, R is a role name, and the TBox is a set of concept inclusion assertions of the form $C_1 \sqsubseteq C_2$. \mathcal{ELH} [7, 6] extends \mathcal{EL} by also allowing in the TBox simple role inclusion assertions of the form $R_1 \sqsubseteq R_2$, where R_1 and R_2 are role names. \mathcal{EL}^+ [5] extends \mathcal{EL} by also allowing in the TBox role inclusion assertions of the form $R_1 \circ \dots \circ R_n \sqsubseteq R_{n+1}$, where each R_i is a role name. Finally, \mathcal{EL}^{++} [3] extends \mathcal{EL}^+ by allowing the new concept expressions $\perp, \{a\}$ and the concrete domain constructor $p(f_1, \dots, f_n)$.

As usual in DLs, a knowledge base (KB) \mathcal{K} is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where the TBox is a set of concept inclusions and role inclusions, and the ABox \mathcal{A} is a set of instance assertions of the form $A(a), R(a, b)$ where A is a concept name, R is a role name, and a, b are constant (individual) names. Notice that in all the DLs considered, \mathcal{T} may contain cyclic concept inclusions (GCIs) (as well as cyclic role inclusions in \mathcal{ELH} , \mathcal{EL}^+ and \mathcal{EL}^{++}).

The semantics of concept and role constructs is well-known [5]. The semantics of a KB is defined as usual, based on the interpretation of concept and role expressions [4]. We point out that we do not impose the unique name assumption (UNA) on constant names: however, our results also hold under the UNA.

As shown in [3], \mathcal{EL}^+ TBoxes admit a normal form, i.e., we can assume without loss of generality that every concept inclusion in the TBox is in one of the following forms: $A_1 \sqsubseteq A_2$, $A_1 \sqcap A_2 \sqsubseteq A_3$, $A_1 \sqsubseteq \exists R.A_2$, $\exists R.A_1 \sqsubseteq A_2$, where each A_1, A_2, A_3 is either a concept name or the concept \top , R is a role name, and every role inclusion is of the form $R_1 \sqsubseteq R_2$ or $R_1 \circ R_2 \sqsubseteq R_3$, where R_1, R_2, R_3 are role names.

Unions of conjunctive queries. We now briefly recall conjunctive queries and unions of conjunctive queries. To simplify the notation in the next sections, we use a Datalog-like notation for such queries.

A Datalog rule is an expression of the form $\alpha :- body$, in which the head α is an atom (i.e., an expression of the form $p(t_1, \dots, t_n)$ in which each t_i is either a constant or a variable) and $body$ is a set of atoms, such that each variable occurring in α also occurs in some atom in $body$.

A conjunctive query (CQ) over a DL-KB \mathcal{K} is a Datalog rule using a special predicate name p_q (i.e., p_q does not belong to the set of concept and role names occurring in \mathcal{K}) in the head of the rule, and whose body is a set of atoms whose predicates are concept and role names occurring in \mathcal{K} (notice that the predicate p_q cannot occur in the body of the rule). The arity of q is defined as the arity of p_q . A Boolean CQ is a CQ whose arity is zero. For a CQ q , we denote by $body(q)$ the body of the Datalog rule corresponding to q . A union of conjunctive queries (UCQ) Q over \mathcal{K} is a set of CQs of the same arity which use the same predicate p_Q in the head of every rule.

For ease of exposition, and without loss of generality, from now on we only consider Boolean queries, and the *query entailment* problem. It is well-known that *query answering* of an arbitrary (non-Boolean) query can be reduced to query entailment.

The semantics of (Boolean) UCQs over DL-KBs is the usual one (see, e.g., [8]).

In the following, we study complexity of query entailment over DL-KBs. In particular, we consider *data complexity*, i.e., the complexity with respect to the size of the ABox, *KB complexity*, i.e., the complexity with respect to the size of both the ABox and the TBox, and *combined complexity*, i.e., the complexity with respect to the size of both the KB and the query.

3 Answering unions of conjunctive queries in \mathcal{EL} and \mathcal{ELH}

We now present an algorithm for answering unions of conjunctive queries posed to \mathcal{EL} -KBs. We start by introducing the auxiliary procedures *Unify*, *Roll-up*, *Normalize*, *Rename*, and *O-Rules*.

The procedure *Unify*. Given a UCQ Q , *Unify*(Q) returns a UCQ obtained by adding to Q all the possible unifications of terms for every conjunctive query q in Q .

The procedure *Roll-up*. Given a UCQ Q' , *Roll-up*(Q') returns a rewriting of the query Q' obtained by expressing subtrees in the query through \mathcal{EL} concept expressions. Formally, we define $\text{Roll-up}(Q') = \bigcup_{q \in Q'} \text{Roll-up}(q)$ where *Roll-up*(q) returns the CQ obtained from the CQ q by exhaustively applying the following rewriting rules to the atoms in $\text{body}(q)$:

1. if variable y only occurs in a binary atom of the form $R(t, y)$, then replace $R(t, y)$ with the unary atom $\exists R.\top(t)$
2. if variable y only occurs in unary atoms of the form $C_1(y), \dots, C_n(y)$, then replace the above atoms with the 0-ary atom $(C_1 \sqcap \dots \sqcap C_n)^0$;
3. if variable y only occurs in unary atoms of the form $C_1(y), \dots, C_n(y)$ and in a single binary atom $R(t, y)$, then replace all the above atoms in which y occurs with the unary atom $(\exists R.C_1 \sqcap \dots \sqcap C_n)(t)$;
4. if y is a variable which only occurs in an atom of the form $R(y, z)$ where z is a variable different from y , and there is another atom of the form $R(t, z)$ in $\text{body}(q)$, then delete $R(y, z)$.

Notice that the query returned by *Roll-up* is not exactly a UCQ according to the definition given in Section 2, since arbitrary concept expressions C may occur as (both unary and 0-ary) predicate symbols in the body of the CQs of the returned query. So we call such a query an *extended UCQ*.

The procedure *Normalize*. Given an \mathcal{EL} TBox \mathcal{T} and an extended UCQ Q' , *Normalize*(\mathcal{T}, Q') returns an \mathcal{EL} TBox \mathcal{T}' in normal form which: (i) is a conservative extension of \mathcal{T} ; (ii) defines all concept expressions occurring in Q' and \mathcal{T}' ; (iii) is closed with respect to the entailed concept inclusions. More precisely, \mathcal{T}' is such that:

- for every concept expression C such that \mathcal{T}' contains a concept inclusion of the form $C \sqsubseteq D$ or $D \sqsubseteq C$, there exists a concept name C' such that $\mathcal{T}' \models C' \equiv C$;
- for every concept expression C such that Q' contains either a unary atom of the form $C(t)$ or a 0-ary atom of the form $(C)^0$, there exists a concept name C' such that $\mathcal{T}' \models C' \equiv C$.
- \mathcal{T}' is closed with respect to the entailment of simple concept inclusions, i.e., for every pair of distinct concept names A_1, A_2 occurring in \mathcal{T}' , if $\mathcal{T}' \models A_1 \sqsubseteq A_2$ then $A_1 \sqsubseteq A_2 \in \mathcal{T}'$.

From the existence of a linear normalization procedure for \mathcal{EL} KBs and from the results on entailment of concept inclusions in \mathcal{EL} shown in [7], it follows that it is possible to compute a TBox \mathcal{T}' satisfying the above conditions in polynomial time with respect to the size of \mathcal{T} and Q' .

The procedure *Rename*. Given an extended UCQ Q' and a normalized \mathcal{EL} TBox \mathcal{T}' , *Rename*(Q', \mathcal{T}') returns the UCQ obtained from Q' by replacing each complex concept

expression C (i.e., such that C is not a concept name) occurring in Q' with the corresponding concept name C' in \mathcal{T}' (i.e., the concept name C' such that $\mathcal{T}' \models C' \equiv C$). Since the presence of complex concept expressions is eliminated from the query returned by $\text{Rename}(Q', \mathcal{T}')$, such a query corresponds to a set of ordinary Datalog rules.

The procedure Rules . Given a normalized \mathcal{EL} TBox \mathcal{T}' , $\text{Rules}(\mathcal{T}')$ returns the set of Datalog rules corresponding to \mathcal{T}' . More precisely, $\text{Rules}(\mathcal{T}')$ is the following set of Datalog rules:

- the rule $A_2(x) :- A_1(x)$ for each concept inclusion $A_1 \sqsubseteq A_2$ in \mathcal{T}' , where A_1, A_2 are concept names;
- the rule $A_3(x) :- A_1(x), A_2(x)$ for each $A_1 \sqcap A_2 \sqsubseteq A_3$ in \mathcal{T}' , where A_1, A_2, A_3 are concept names;
- the rule $A_2(x) :- R(x, y), A_1(y)$ for each $\exists R.A_1 \sqsubseteq A_2$ in \mathcal{T}' , where A_1, A_2 are concept names.

Notice that concept inclusions of the form $A_1 \sqsubseteq \exists R.A_2$ are not actually considered in the computation of $\text{Rules}(\mathcal{T}')$.

The procedure 0-Rules . Finally, to correctly handle 0-ary atoms in the query, we have to define entailment of inclusions between 0-ary predicates with respect to the TBox \mathcal{T}' . In particular, for every pair of concept names A_1, A_2 occurring in \mathcal{T}' , we want to decide whether the first-order existential sentence $\exists x.A_1(x) \rightarrow \exists y.A_2(y)$ is satisfied by every model of \mathcal{T}' . Actually, entailment of such sentences can be decided in a way very similar to entailment of concept inclusions. More precisely, we define inductively the following relation $\vdash_{\mathcal{T}'}^{\exists}$, between concept names occurring in \mathcal{T}' :

- $A \vdash_{\mathcal{T}'}^{\exists}, A$ for every concept name A occurring in \mathcal{T}' ;
- if $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_2$ and $A_2 \sqsubseteq A_3 \in \mathcal{T}'$ then $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_3$;
- if $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_2$ and $A_2 \sqsubseteq \exists R.A_3 \in \mathcal{T}'$, then $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_3$.

Based on the fact that \mathcal{T}' is closed with respect to entailment of inclusions between atomic concepts, it can be shown that, for every pair of concept names A_1, A_2 occurring in \mathcal{T}' , the sentence $\exists x.A_1(x) \rightarrow \exists y.A_2(y)$ is satisfied by every model of \mathcal{T}' iff $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_2$.

Then, based on the relation $\vdash_{\mathcal{T}'}^{\exists}$, we define the procedure $0\text{-Rules}(\mathcal{T}')$, which returns the following set of Datalog rules:

- $A_2^0 :- A_1^0$ for each pair of concept names A_1, A_2 such that $A_1 \vdash_{\mathcal{T}'}^{\exists}, A_2$;
- $A^0 :- A(x)$ for each concept name A .

The algorithm computeRewriting . We are now ready to define the algorithm computeRewriting which, given an \mathcal{EL} TBox \mathcal{T} and a Boolean UCQ Q , computes a Datalog program \mathcal{P} by making use of the procedures previously defined.

Algorithm $\text{computeRewriting}(Q, \mathcal{T})$

Input: Boolean union of conjunctive queries Q , \mathcal{EL} TBox \mathcal{T}

Output: Datalog program \mathcal{P}

```

 $Q' := \text{Unify}(Q);$ 
 $Q' := \text{Roll-up}(Q');$ 
 $\mathcal{T}' := \text{Normalize}(\mathcal{T}, Q');$ 
 $Q' := \text{Rename}(Q', \mathcal{T}');$ 
 $\mathcal{P} := Q' \cup \text{Rules}(\mathcal{T}') \cup 0\text{-Rules}(\mathcal{T}');$ 
return  $\mathcal{P}$ 

```

The algorithm `computeQueryEntailment`. The Datalog program \mathcal{P} computed by `computeRewriting(Q, \mathcal{T})` can be used to decide entailment of the query Q with respect to every \mathcal{EL} -KB, as shown by the algorithm `computeQueryEntailment(Q, \mathcal{K})` defined below. In the following, $\top(\mathcal{A})$ denotes the set of facts $\{\top(a) \mid a \text{ is a constant occurring in } \mathcal{A}\}$, while $MM(\mathcal{P})$ denotes the minimal model of a Datalog program \mathcal{P} .

Algorithm `computeQueryEntailment(Q, \mathcal{K})`

Input: Boolean UCQ Q (with head predicate p_Q), \mathcal{EL} -KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

Output: *true* if $\mathcal{K} \models Q$, *false* otherwise

```

 $\mathcal{P} := \text{computeRewriting}(Q, \mathcal{T});$ 
if  $MM(\mathcal{P} \cup \mathcal{A} \cup \top(\mathcal{A})) \models p_Q$ 
then return true else return false

```

In practice, the above algorithm simply evaluates the Datalog program \mathcal{P} over the ABox \mathcal{A} (remember that \mathcal{A} is a set of ground atoms, hence $\mathcal{P} \cup \mathcal{A}$ is a Datalog program) in order to decide whether Q is entailed by \mathcal{K} . The addition of the facts $\top(\mathcal{A})$ is necessary in order to correctly handle the presence of the concept \top in the query (more precisely, in the evaluation of the Datalog program we consider \top as a concept name, i.e., an EDB predicate).

Correctness. We now show correctness of the algorithm `computeQueryEntailment`.

Theorem 1. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an \mathcal{EL} -KB and let Q be a UCQ. Then, $\mathcal{K} \models Q$ iff `computeQueryEntailment(Q, \mathcal{K})` returns *true*.

Proof (sketch). The proof of soundness of the technique is immediate. The proof of completeness is based on the construction of a canonical model for a normalized \mathcal{EL} -KB \mathcal{K} through the definition of the *chase* of \mathcal{K} . The chase of \mathcal{K} (denoted by $chase(\mathcal{K})$) is a function which returns a generally infinite ABox and is inductively defined starting from the initial ABox \mathcal{A} and adding facts to \mathcal{A} based on the following *chase rules*:

- *chase-rule-1*: if $A(a) \in chase(\mathcal{K})$ and $A \sqsubseteq B \in \mathcal{T}$ and $B(a) \notin chase(\mathcal{K})$ then add $B(a)$ to $chase(\mathcal{K})$;
- *chase-rule-2*: if $A_1(a) \in chase(\mathcal{K})$ and $A_2(a) \in chase(\mathcal{K})$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B(a) \notin chase(\mathcal{K})$ then $B(a) \in chase(\mathcal{K})$;
- *chase-rule-3*: if $R(a, b) \in chase(\mathcal{K})$ and $A(a) \in chase(\mathcal{K})$ and $\exists R.A \sqsubseteq B \in \mathcal{T}$ and $B(a) \notin chase(\mathcal{K})$ then $B(a) \in chase(\mathcal{K})$;
- *chase-rule-4*: if $A(a) \in chase(\mathcal{K})$ and $A \sqsubseteq \exists R.B \in \mathcal{T}$ and there is no b such that both $R(a, b) \in chase(\mathcal{K})$ and $B(b) \in chase(\mathcal{K})$ then add $R(a, n)$ and $B(n)$ to $chase(\mathcal{K})$, where n is a constant that does not occur already in $chase(\mathcal{K})$;
- *chase-rule-5*: if a is a constant occurring in $chase(\mathcal{K})$ and $\top(a) \notin chase(\mathcal{K})$ then add $\top(a)$ to $chase(\mathcal{K})$.

The chase of \mathcal{K} is a generally infinite ABox which is isomorphic to a *canonical model* of \mathcal{K} , denoted by $\mathcal{I}_{chase(\mathcal{K})}$. Such a model can be used to compute entailment of UCQs in \mathcal{K} , which is formally stated by the following property:

Lemma 1. For every Boolean UCQ Q , $\mathcal{K} \models Q$ iff $\mathcal{I}_{chase(\mathcal{K})} \models Q$.

Then, we use the chase to prove completeness of our algorithm. Let us consider the first part of the algorithm `computeRewriting`, which ends with the execution of `Rename(Q', \mathcal{T}')`. With respect to this part of the rewriting, we prove the following:

Lemma 2. Let Q'' be the UCQ returned by $\text{Rename}(Q', \mathcal{T}')$ in the algorithm $\text{computeRewriting}(Q, \mathcal{T})$. If $\mathcal{I}_{\text{chase}(\mathcal{K})} \models Q''$ then there exists a CQ $q \in Q''$ and a homomorphism H of $\text{body}(q)$ in $\text{chase}(\mathcal{K})$ such that, for every variable x occurring in q , $H(x)$ is a constant occurring in \mathcal{A} .

Although the above lemma might seem rather obscure, it implies the following crucial property: answering the query computed by $\text{Rename}(Q', \mathcal{T}')$ can actually be done by first “grounding” the query (considering all the instantiations of the variables with constants occurring in \mathcal{A}) and then considering each atom in a separate way. So, the above lemma shows that the first part of the rewriting reduces entailment of a UCQ to entailment of single atoms.

Then, we consider the second part of the rewriting, i.e., the set of Datalog rules generated by $\text{Rules}(\mathcal{T}')$ and $0\text{-Rules}(\mathcal{T}')$. Here, we use the chase of \mathcal{K} to prove that the Datalog program \mathcal{P}' returned by $\text{Rules}(\mathcal{T}')$ constitutes a correct encoding of the entailment of unary and binary atoms (which correspond to standard instance checking problems), in the sense that the minimal model of $\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})$ contains all ground unary atoms $A(a)$ such that $\mathcal{K} \models A(a)$ and all ground binary atoms $R(a, b)$ such that $\mathcal{K} \models R(a, b)$; similarly, we prove that the Datalog program returned by $0\text{-Rules}(\mathcal{T}')$ constitutes a correct encoding of the entailment of 0-ary atoms. Formally:

Lemma 3. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a normalized \mathcal{EL} -KB, let \mathcal{P}' be the Datalog program returned by $\text{Rules}(\mathcal{T})$, and let α be either an atom of the form $A(a)$ where A is a concept name or an atom of the form $R(a, b)$ where R is a role name (a, b are constants occurring in \mathcal{A}). If $\mathcal{K} \models A(a)$ then $\text{MM}(\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})) \models \alpha$.

Lemma 4. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a normalized \mathcal{EL} -KB, let \mathcal{P}' be the Datalog program returned by $0\text{-Rules}(\mathcal{T})$, and let A be a concept name occurring in \mathcal{T} . If the sentence $\exists x.A(x)$ is satisfied by every model for \mathcal{K} , then $\text{MM}(\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})) \models A^0$.

From the above properties of the two parts of the rewriting, the thesis follows. \square

Complexity results. Based on the above algorithm, we can characterize the computational properties of entailment of UCQs in \mathcal{EL} .

Theorem 2. Entailment of UCQs in \mathcal{EL} is: (i) PTIME-complete with respect to data complexity; (ii) PTIME-complete with respect to KB complexity; (iii) NP-complete with respect to combined complexity.

Proof (sketch). PTIME-hardness with respect to data complexity has been proved in [8], while NP-hardness with respect to combined complexity follows from NP-hardness of simple database evaluation of a CQ [1]. Membership in PTIME with respect to KB complexity follows from the fact that the procedures *Normalize*, *Rename*, *Rules*, and *0-Rules* run in time polynomial with respect to their input, which implies that the algorithm *computeRewriting* runs in time polynomial with respect to the size of \mathcal{T} . Now, since the Datalog program \mathcal{P} returned by *computeRewriting*(Q, \mathcal{T}) has size polynomial with respect to \mathcal{T} , and the number of variables used in each rule of \mathcal{P} does not depend on \mathcal{K} , it follows that the minimal model of $\mathcal{P} \cup \mathcal{A} \cup \top(\mathcal{A})$ can be computed in time polynomial in the size of \mathcal{K} , thus the algorithm *computeQueryEntailment*(Q, \mathcal{K}) also runs in time polynomial in the size of \mathcal{K} . Finally, membership in NP with respect to combined complexity follows from the fact that all the procedures executed by

`computeRewriting` run in time polynomial with respect to their input, with the only exception of the procedure *Unify*, which runs in exponential time with respect to the size of Q . However, if we consider a nondeterministic version of such a procedure, which returns just one CQ q' obtained by choosing one CQ q in Q and one substitution which is applied to q , then the whole algorithm `computeRewriting` runs in time polynomial with respect to the size of both Q and \mathcal{T} . Then, in a way analogous to the above proof of PTIME-membership for KB complexity, it follows that this nondeterministic version of the algorithm `computeQueryEntailment`(Q, \mathcal{K}) also runs in time polynomial in the size of Q and \mathcal{K} , which implies the thesis. \square

We remark that the above characterization with respect to data complexity was already stated in [12].

Extension to \mathcal{ELH} . Finally, the above technique for deciding entailment of UCQs can be easily extended in order to deal with \mathcal{ELH} -KBs. The algorithms `computeRewriting` and `computeQueryEntailment` are actually the same as before, the only differences concern the procedures *Roll-up*, *Normalize*, and *Rules*. Specifically: (i) the procedure *Roll-up* must take into account the presence of role assertions, since such inclusions allow for additional eliminations of redundant binary atoms. More precisely, we add to the previous definition of *Roll-up* the following rule: if R_1 and R_2 are distinct role names, $R_1(t_1, t_2)$ and $R_2(t_1, t_2)$ occur in *body*(q), and $\mathcal{T} \models R_1 \sqsubseteq R_2$, then delete $R_2(t_1, t_2)$; (ii) the procedure *Normalize*(\mathcal{T}, Q') must be modified in order to account for the presence of simple role inclusions in the TBox. A procedure for deciding entailment of concept and role inclusions in \mathcal{ELH} TBoxes has been presented in [7]: such a procedure shows that entailment of such inclusions can still be computed in polynomial time; (iii) the procedure *Rules* also adds a Datalog rule for each role inclusion in the \mathcal{ELH} TBox. More precisely, in the case of an \mathcal{ELH} TBox, the previous definition of the set of rules returned by *Rules*(\mathcal{T}') is modified by adding the following condition: add the rule $R_2(x, y) :- R_1(x, y)$ for each role inclusion $R_1 \sqsubseteq R_2$ in \mathcal{T}' .

The above extension demonstrates that the computational characterization of entailment of UCQs provided by Theorem 2 extends to the case of \mathcal{ELH} -KB.

4 Undecidability of conjunctive query answering in \mathcal{EL}^+

We now show that the nice computational properties of answering conjunctive queries in \mathcal{EL} , shown in the previous section, do not extend to \mathcal{EL}^+ and \mathcal{EL}^{++} , since answering conjunctive queries in such DLs is undecidable.

Theorem 3. *Entailment of conjunctive queries in \mathcal{EL}^+ is undecidable.*

Proof (sketch). We reduce the emptiness problem for intersection of context-free languages, which is known to be undecidable [9], to conjunctive query entailment in \mathcal{EL}^+ . Consider two context-free grammars $G_1 = \langle NT_1, Term, S_1, P_1 \rangle$, $G_2 = \langle NT_2, Term, S_2, P_2 \rangle$, where NT_1 is the alphabet of nonterminal symbols of G_1 , NT_2 is the alphabet of nonterminal symbols of G_2 (which is disjoint from NT_1), $Term$ is the alphabet of terminal symbols (which is the same for G_1 and G_2 and is disjoint from $NT_1 \cup NT_2$), S_1 is the axiom of G_1 , S_2 is the axiom of G_2 , P_1 is the set of production rules of G_1 and P_2 is the set of production rules of G_2 . W.l.o.g., we assume that no production rule has an empty right-hand side. Now consider the \mathcal{EL}^+ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, which uses $Term \cup NT_1 \cup NT_2$ as the set of role names plus the concept name C . The TBox \mathcal{T} is composed of: (i) the role inclusion assertions encoding the production rules

in P_1 : e.g., if P_1 contains the production rule $X \rightarrow UVW$, we add to \mathcal{T} the role inclusion $U \circ V \circ W \sqsubseteq X$; (ii) the role inclusion assertions encoding the production rules in P_2 ; (iii) the role inclusion assertion $C \sqsubseteq \exists T_i.C$ for every $T_i \in \text{Term}$. Moreover, the ABox \mathcal{A} contains the only assertion $C(a)$. Finally, consider the Boolean conjunctive query q of the form $p_q :- S_1(a, y), S_2(a, y)$, where S_1 is the axiom of grammar G_1 and S_2 is the axiom of grammar G_2 .

We prove that the language $\mathcal{L}(G_1) \cap \mathcal{L}(G_2)$ is non-empty iff $\langle \mathcal{T}, \mathcal{A} \rangle \models q$. To this aim, we make use of three auxiliary properties. Such properties make use of the notion of chase of an \mathcal{EL}^+ -KB, which extends in a straightforward way the chase for \mathcal{EL} , introduced in the proof of Theorem 1, by adding a chase rule for role inclusions.

Lemma 5. *Let x and y be two terms in $\text{chase}(\mathcal{K})$. There is at most one path of terminal symbols between x and y in $\text{chase}(\mathcal{K})$, i.e., a sequence $T_1(z_1, z_2), \dots, T_k(z_k, z_{k+1})$ with $z_1 = x$, $z_{k+1} = y$, and s.t. each $T_i(z_i, z_{i+1}) \in \text{chase}(\mathcal{K})$ and each $T_i \in \text{Term}$.*

Lemma 6. *Let x and y be two terms in $\text{chase}(\mathcal{K})$. Let π be the path of terminal symbols between x and y in $\text{chase}(\mathcal{K})$. Then, for every nonterminal symbol $N \in NT_1$ (resp., for every $N \in NT_2$), $N(x, y) \in \text{chase}(\mathcal{K})$ iff $N \Rightarrow_{G_1}^* \pi$ (resp., iff $N \Rightarrow_{G_2}^* \pi$).*

Lemma 7. *For every word $T_1 \dots T_k$ in Term^* , there exists a pair x, y such that there exists the path of terminal symbols $T_1 \dots T_k$ between x and y in $\text{chase}(\mathcal{K})$.*

From the above properties, the thesis easily follows. \square

Obviously, the above theorem also implies undecidability of conjunctive query entailment (and thus of conjunctive query answering) in \mathcal{EL}^{++} .

We point out that the above theorem has been independently proved by other authors [11, 10].

Acknowledgments. This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU, by project HYPER, funded by IBM through a SUR Award grant, and by MIUR FIRB project TOCAI.IT.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
2. F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of IJCAI 2003*, pages 325–330, 2003.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the \mathcal{EL} envelope. In *Proc. of IJCAI 2005*, pages 364–369, 2005.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*, 2003.
5. F. Baader, C. Lutz, and B. Suntisrivaraporn. Efficient reasoning in \mathcal{EL}^+ . In *Proc. of DL 2006*. <http://ceur-ws.org/Vol-189/>, 2006.
6. S. Brandt. On subsumption and instance problem in ELH w.r.t. general TBoxes. In *Proc. of DL 2004*. <http://ceur-ws.org/Vol-104/>, 2004.
7. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In *Proc. of ECAI 2004*, pages 298–302, 2004.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
9. M. A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, 1978.
10. A. Krisnadhi and C. Lutz. Data complexity in the EL family of DLs. In *Proc. DL 2007*.
11. M. Krötzsch and S. Rudolph. Conjunctive queries for EL with role composition. In *Proc. DL 2007*.
12. R. Rosati. The limits of querying ontologies. In *Proc. ICDT 2007, LNCS 4353*, pages 164–178. Springer, 2007.