Query Rewriting under Linear $\mathcal{EL}$ Knowledge Bases

Mirko M. Dimartino[1], Andrea Calì[1,2],
Alexandra Poulovassilis[1], and Peter T. Wood[1]

[1] Knowledge Lab, Birkbeck, University of London
[2] Oxford-Man Institute of Quantitative Finance, University of Oxford.
{mirko,andrea,ap,ptw}@dcs.bbk.ac.uk

Abstract. With the adoption of the recent SPARQL 1.1 standard, RDF databases are capable of directly answering more expressive queries than simple conjunctive queries. In this paper we exploit such capabilities to answer conjunctive queries (CQs) under ontologies expressed in the description logic called *linear* $\mathcal{EL}^{\ell in}$, a restricted form of $\mathcal{EL}$. In particular, we show a query answering algorithm that rewrites a given CQ into a conjunctive regular path query (CRPQ) which, evaluated on the given instance, returns the correct answer. Our technique is based on the representation of infinite unions of CQs by non-deterministic finite-state automata. Our results achieve optimal data complexity, as well as producing rewritings straightforwardly implementable in SPARQL 1.1.

1 Introduction

Ontologies have been successfully employed in conceptual modelling of data in several areas, especially Information Integration and the Semantic Web. An ontology is a specification of the domain of interest of an application, and it is usually specified in terms of logical rules which on the one hand restrict the form of the underlying data, and on the other hand allow for *inference* of information that is not explicitly contained in the data. Description Logic (DL) is a common family of knowledge representation formalisms that are able to capture a wide range of ontological constructs [2]; they are based on *concepts* (unary predicates representing classes of individuals) and *roles* (binary predicates representing relations between classes). A DL knowledge base consists of a TBox (*terminological* component) and an ABox (*assertional* component); the former is a conceptual representation of the schema, while the latter is an instance of the schema. It is important to note that a usual assumption in this context is the so-called *open-world* assumption, that is, the information in the ABox is sound but not complete; the TBox, in particular, determines how the ABox is to be completed with additional information so as to answer queries. Answers to a query in this context are called, following database parlance, *certain answers*, as they correspond to the answers that are true in all models of the theory constituted by the knowledge base. This corresponds to *cautious* reasoning as opposed to *bold* reasoning, where in the latter an answer is returned if it is entailed by at least one model. The set of all models (which is not necessarily finite) is represented by the so-called *expansion* (called *chase* in database parlance) of an ABox $\mathcal{A}$ according to a TBox $\mathcal{T}$; this is illustrated in the following example.

Example 1. Consider the TBox $\mathcal{T}$ constituted by the assertions $C \sqsubseteq A$ and $A \sqsubseteq \exists\, S.C$. The concept $\exists\, S.C$ denotes the objects connected via the role S to some object belonging to the concept C; in other words, it contains all x such that $S(x, y)$ and $C(y)$ for some y. The first assertion means that every object in the class C is also in A; the second means that every object in the class A is also in the class represented by $\exists\, S.C$. Now suppose we have the ABox $\mathcal{A} = \{A(a)\}$; we can *expand* $\mathcal{A}$ according to the the TBox $\mathcal{T}$ so as to add to it all atoms entailed by $(\mathcal{T}, \mathcal{A})$; we therefore add $S(a, z_0)$ and $C(z_0)$, where z_0 is a so-called *labelled null*, that is, a placeholder for an unknown value of which we know the existence. Given the query $\boldsymbol{q}$ defined as $q(x) \leftarrow S(x, y)$, the answer to it under $(\mathcal{T}, \mathcal{A})$ is $\{a\}$ because $S(a, z_0)$ is entailed by $(\mathcal{T}, \mathcal{A})$; in fact, the certain answers to $\boldsymbol{q}$ are obtained by evaluating $\boldsymbol{q}$ on the expansion and by considering answers that do not contain nulls. If we consider the query $\boldsymbol{q}_1$ defined as $q_1(x) \leftarrow C(x)$, the answer is empty because z_0, though known to exist, is not known.

Answers to queries over DL knowledge bases can be computed, in certain cases, by a technique called *query rewriting*. In query rewriting, starting from a given query $\boldsymbol{q}$, a new query $\boldsymbol{q}'$ is computed according to a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, such that the answers to $\boldsymbol{q}$ on $\mathcal{K}$ are obtained by evaluating $\boldsymbol{q}'$ on $\mathcal{A}$ only; it is said that $\boldsymbol{q}$ is *rewritten* into $\boldsymbol{q}'$ and that $\boldsymbol{q}'$ is the *perfect rewriting of* $\boldsymbol{q}$ *with respect to* $\mathcal{T}$. The language of $\boldsymbol{q}'$, called the *target language*, can be more expressive than that of $\boldsymbol{q}$. A common rewriting technique for DLs and other knowledge representation formalisms, inspired by resolution in Logic Programming, has *union of conjunctive queries* as target language.

Example 2. Let us consider again the knowledge base of Example 1. The query $\boldsymbol{q}$ is rewritten into the query $\boldsymbol{q}'$ defined as $q(x) \leftarrow A(x) \cup S(x, y)$; intuitively, $\boldsymbol{q}'$ captures the fact that, to search for objects from which some other object is connected via the role S, we need also to consider objects in A, because the TBox might infer the former from the latter objects. The evaluation of $\boldsymbol{q}'$ on $\mathcal{A}$ returns the correct answer.

In this paper we consider a DL which we call $\mathcal{EL}^{\ell in}$ [17]. When executed on $\mathcal{EL}^{\ell in}$ TBoxes, the above rewriting technique does not guarantee termination. We therefore resort to a more expressive target language for the rewriting, namely *conjunctive regular path queries* (CRPQs).

Example 3. Consider the TBox $\mathcal{T} = \{\exists\, R.A \sqsubseteq A\}$ and the query $\boldsymbol{q}$ defined as $q(x) \leftarrow A(x)$. It is easy to see that the above rewriting technique produces an infinite union of conjunctive queries: $q(x) \leftarrow A(x)$, $q(x) \leftarrow R(x, y), A(y)$ and all conjunctive queries of the form $q(x) \leftarrow R(x, y_1), \ldots, R(y_k, y_{k+1}), A(y_{k+1})$, with $k \geqslant 1$. Now, in order to capture this infinite rewriting, we can resort to the CRPQ rewriting $\boldsymbol{q}'$ defined as $q(x) \leftarrow R^*(x, y), A(y)$.

In this paper we propose a novel rewriting technique for the DL $\mathcal{EL}^{\ell in}$, where the query language is that of *conjunctive queries* (CQs) and the target language is that of CRPQs. This allows us to devise a query answering algorithm that has optimum asymptotic complexity and relies on pure rewriting, without ABox expansion. Notice that rewriting is generally considered efficient as the processing operates solely on the query, while the ABox, which is normally considered to be much larger than the query and the TBox, comes into play only at the last step, when the rewriting is evaluated on it.

Our contributions are as follows.

- For illustrative purposes and technical reasons, we show a rewriting algorithm for conjunctive queries on $\mathcal{EL}^{\ell in}$ knowledge bases, which relies on a resolution-like procedure widely adopted in the literature (see e.g. [10]).
- We present a novel rewriting technique, based on non-deterministic finite-state automata, for *atomic* queries on $\mathcal{EL}^{\ell in}$ knowledge bases, with CRPQs as target language. Intuitively, the expressive power of CRPQs is able to finitely capture the infinite rewriting branches of the above algorithm.
- Finally, based on the rewriting technique for atomic queries, we present a technique for rewriting CQs into CRPQs. This is achieved by splitting the problem in two: first we deal with assertions that do not introduce labelled nulls; then, we show that the rest of the assertions are guaranteed to have only tree-like (or, more precisely, forest-like) models; this allows us to capture all paths (including the infinite ones) from roots in the forest by means of finite-state automata. The final rewriting is a CRPQ whose evaluation on the given ABox returns the correct answers to the initial CQ. Since CRPQs can be straightforwardly expressed in SPARQL 1.1, by the means of property paths, our approach is suitable for real-world settings.
- Our technique achieves optimal computational cost in *data complexity*, that is where the TBox and the query are fixed and the ABox alone is a variable input; in fact, our algorithm runs in NLOGSPACE in data complexity, which is the known (tight) bound for CQ answering under $\mathcal{EL}^{\ell in}$ knowledge bases. Notice also that, regarding *combined complexity* (where query, TBox and ABox all constitute the variable input), our rewriting is expressed in the language of CRPQs, which can be evaluated in NP. Moreover, in the case of "simple" queries such as atomic queries, our rewriting is expressed as a regular path query, which can be evaluated in NLOGSPACE.

2 Preliminaries

In this section we present the formal notions which will be used in the rest of the paper.

2.1 Description Logics

We briefly introduce the syntax of the $\mathcal{EL}^{\ell in}$ description logic (DL) [18, 16]. The alphabet contains three pairwise disjoint and countably infinite sets of *concept names* A, *role names* R, and *individual names* I. A *complex concept* C is constructed from a special primitive concept $\top$ ('top'), concept names and role names using the following grammar: $C ::= A \mid \exists R.C \mid \exists R.\top$, where $A \in \mathsf{A}$ and $R \in \mathsf{R}$. The set of complex concepts is denoted by C. A *terminological box* (or TBox) $\mathcal{T}$ is a finite set of *concept* and *role inclusion axioms* of the form $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$, where $C_1, C_2 \in \mathsf{C}$ and $R_1, R_2 \in \mathsf{R}$. An *assertion box* (or ABox) $\mathcal{A}$ is a finite set of *concept* and *role assertions* of the form $C(a)$ and $R(a, b)$, where $C \in \mathsf{C}$, $R \in \mathsf{R}$ and $a, b \in \mathsf{I}$. Given an ABox $\mathcal{A}$, we denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names that occur in $\mathcal{A}$. Taken together, $\mathcal{T}$ and $\mathcal{A}$ comprise a *knowledge base* (or KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

We adopt the semantics of DL defined in terms of interpretations. An *interpretation* $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty *domain of interpretation* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ which assigns *(i)* an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each individual name a, *(ii)* a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name A and *(iii)* a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name P. We adopt the *unique name assumption* (UNA); therefore distinct individuals are assumed to be interpreted by distinct domain elements. The interpretation function $\cdot^{\mathcal{I}}$ is extended inductively for complex concepts by taking:

$$\begin{aligned}(\exists R.\top)^{\mathcal{I}} &= \{u \mid \text{there is a } v \text{ such that } (u,v) \in R^{\mathcal{I}}\}\ ,\\ (\exists R.C)^{\mathcal{I}} &= \{u \mid \text{there is a } v \in C^{\mathcal{I}} \text{ such that } (u,v) \in R^{\mathcal{I}}\}\ .\end{aligned}$$

We now define the satisfaction relation $\models$ for inclusions and assertions:

$$\begin{aligned}\mathcal{I} \models C_1 \sqsubseteq C_2 &\text{ if and only if } C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}\ ,\\ \mathcal{I} \models R_1 \sqsubseteq R_2 &\text{ if and only if } R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}\ ,\\ \mathcal{I} \models C(a) &\text{ if and only if } a^{\mathcal{I}} \in C^{\mathcal{I}}\ ,\\ \mathcal{I} \models R(a,b) &\text{ if and only if } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}\ .\end{aligned}$$

We say that an interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models \mathcal{K}$, if it satisfies all concept and role inclusions of $\mathcal{T}$ and all concept and role assertions of $\mathcal{A}$. A TBox is said to be in *normal form* if each of its concept inclusion axioms is of the forms $A \sqsubseteq B$, $\exists P.C \sqsubseteq A$, or $A \sqsubseteq \exists P.C$, where $A, B \in \mathsf{A}$, $C \in \mathsf{A} \cup \{\top\}$ and $P \in \mathsf{R}$. We recall that every $\mathcal{EL}$ TBox can be transformed into an equivalent TBox in normal form of size that is linear in the size of the original TBox [14].

2.2 Regular Languages and Conjunctive Regular Path Queries

We assume the reader is familiar with regular languages, represented either by regular expressions or nondeterministic finite state automata. A *nondeterministic finite state automaton* (NFA) over a set of symbols Σ is a tuple $\alpha = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, $\delta \subseteq Q \times \Sigma \times Q$ the *transition relation*, $q_0 \in Q$ the *initial state*, and $F \subseteq S$ the set of *final states*. We use $L(\alpha)$ to denote the regular language defined by an NFA α, and $(\Sigma)^*$ to denote the set of all strings over symbols in Σ, including the *empty string* ϵ.

In order to define queries, we also need to assume the existence of a countably infinite set of *variables* V. A *term* t is an individual name in I or a variable in V. An *atom* is of the form $\alpha(t, t')$, where t, t' are terms, and α is an NFA or regular expression defining a regular language over $\mathsf{R} \cup \mathsf{A}$. We say that a string $s \in (\mathsf{R} \cup \mathsf{A})^*$ is a *path*.

A *conjunctive regular path query* (CRPQ) $\boldsymbol{q}$ of arity n has the form $q(\boldsymbol{x}) \leftarrow \gamma(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} = x_1, \ldots, x_n$ and $\boldsymbol{y} = y_1, \ldots, y_m$ are variables, and $\gamma(\boldsymbol{x}, \boldsymbol{y})$ is a set of atoms with variables from $\boldsymbol{x}$ and $\boldsymbol{y}$. $q(\boldsymbol{x})$ is called the *head* of $\boldsymbol{q}$ and is denoted by $head(\boldsymbol{q})$, and $\gamma(\boldsymbol{x}, \boldsymbol{y})$ is the *body* of $\boldsymbol{q}$ and denoted by $body(\boldsymbol{q})$. The variables in $\boldsymbol{x} = x_1, \ldots, x_n$ are the *answer variables* of $\boldsymbol{q}$, while those in $\boldsymbol{y} = y_1, \ldots, y_m$ are the *existentially quantified variables* of $\boldsymbol{q}$. A *Boolean CRPQ* is a CRPQ with no answer variables. A *regular path query* (RPQ) is a CRPQ with a single atom in its body. A *path query* (PQ) is an RPQ $q = head(q) \leftarrow \alpha(x, y)$ such that $\alpha \in (\mathsf{R} \cup \mathsf{A})^*$, where α is the *path of* q denoted by $path(q)$.

A *conjunctive query* (CQ) $\boldsymbol{q}$ is a CRPQ such that, for each atom $\alpha(t, t') \in body(\boldsymbol{q})$, $\alpha \in (\mathsf{R} \cup \mathsf{A})$. Informally, CQs disallow regular expressions in their bodies. Given a CRPQ $\boldsymbol{q}$ with answer variables $\boldsymbol{x} = x_1, \ldots, x_n$ and an n-tuple of individuals $\boldsymbol{a} = (a_1, \ldots, a_n)$, we use $\boldsymbol{q}(\boldsymbol{a})$ to refer to the Boolean query obtained from $\boldsymbol{q}$ by replacing x_i with a_i in $body(\boldsymbol{q})$, for every $1 \leqslant i \leqslant n$.

We now define the semantics of CRPQs [7]. Given the individual names a, b, an interpretation $\mathcal{I}$, and a regular language α over the alphabet $\mathsf{R} \cup \mathsf{A}$, we have that $\mathcal{I} \models a \xrightarrow{\alpha} b$ if and only if there is some $w = u_1 \ldots u_n \in L(\alpha)$ and some sequence $e_0, \ldots, e_n$ with $e_i \in \Delta^{\mathcal{I}}$, $0 \leqslant i \leqslant n$, such that $e_0 = a^{\mathcal{I}}$ and $e_n = b^{\mathcal{I}}$, and for all $1 \leqslant i \leqslant n$: *(i)* if $u_i = A \in \mathsf{A}$, then $e_{i-1} = e_i \in A^{\mathcal{I}}$; *(ii)* if $u_i = R \in \mathsf{R}$, then $(e_{i-1}, e_i) \in R^{\mathcal{I}}$. A *match* for a Boolean CRPQ $\boldsymbol{q}$ in an interpretation $\mathcal{I}$ is a mapping π from the terms in $body(\boldsymbol{q})$ to the elements in I such that: *(1)* $\pi(c) = c$ if $c \in \mathsf{I}$; *(2)* $\mathcal{I} \models \pi(t) \xrightarrow{\alpha} \pi(t')$ for each atom $\alpha(t, t')$ in $\boldsymbol{q}$.

Note that, to avoid notational clutter, we do not allow unary atoms in the body of the query. In fact, each atom of the form $A(t)$, where $A \in \mathsf{A}$ and $t \in \mathsf{V} \cup \mathsf{I}$, can be always replaced by a binary atom $A(t, z)$, where z is a variable. However, for better legibility, we use unary atoms in some examples throughout the paper. We write $\mathcal{I} \models \boldsymbol{q}$ if there is a match for $\boldsymbol{q}$ in $\mathcal{I}$, and $\mathcal{K} \models \boldsymbol{q}$ if $\mathcal{I} \models \boldsymbol{q}$ for every model $\mathcal{I}$ of the KB $\mathcal{K}$. For brevity, given an ABox $\mathcal{A}$ we use $\mathcal{A} \models \boldsymbol{q}$ to refer to $(\varnothing, \mathcal{A}) \models \boldsymbol{q}$, where $(\varnothing, \mathcal{A})$ is a knowledge base with empty TBox. Given a CRPQ $\boldsymbol{q}$ of arity n we say that a tuple of individual names $\boldsymbol{a} = (a_1, \ldots, a_n)$ is a *certain answer* for $\boldsymbol{q}$ with respect to a KB $\mathcal{K}$ if and only if $\mathcal{K} \models \boldsymbol{q}(\boldsymbol{a})$.

3 Rewriting of Conjunctive Queries into First-Order Queries

In this section we show a technique for rewriting CQs into a *union of conjunctive queries* under an $\mathcal{EL}^{\ell in}$ TBox. We base our approach on the rewriting algorithm proposed in [9], which deals with $DL\text{-}Lite_{\mathcal{R}}$ KBs. We do this to establish correctness of the rewriting approach with the NFAs illustrated in Section 4. The technique is based on two steps: a *reduction* step, which eliminates atoms that are more specific than some other atom, and the actual *rewriting* step, which is similar to the resolution step in logic programming. Notice that the algorithm might not terminate; we present it for technical reasons, as in Sections 4 and 5 we will show that our CRPQ rewriting captures all the rewriting branches produced by the algorithm, including infinite ones.

Following the approach of [9], we say that a term of an atom in a query is *bound* if it corresponds to (i) an answer variable, (ii) a shared variable, that is, a variable occurring at least twice in the query body, or (iii) a constant, that is an element in I. Conversely, a term of an atom in a query is *unbound* if it corresponds to a non-shared existentially quantified variable. As usual, we use the symbol '_' to represent an unbound term.

A set of atoms $A = \{a_1, \ldots, a_n\}$, where $n > 2$, *unifies* if there exists a substitution ϕ, called *unifier* for A, such that (i) if $t \in \mathsf{I}$, then $\phi(t) = t$, and (ii) $\phi(a_1) = \cdots = \phi(a_n)$. *Reduce* is a function that takes as input a conjunctive query q and a set of atoms S occurring in the body of q and returns a conjunctive query q obtained by applying to q

the *most general unifier* between the atoms of S. We point out that, in unifying a set of atoms, each occurrence of the _ symbol is considered to be a different unbound variable.

We now define when concept and role inclusion axioms are *applicable* to atoms in a query. An axiom I is *applicable to an atom* $A(x_1, x_2)$ for $A \in \mathsf{A}$ if I is of the form $B \sqsubseteq A$, $\exists R.\top \sqsubseteq A$ or $\exists R.B \sqsubseteq A$. An axiom I is *applicable to an atom* $P(x_1, x_2)$ for $P \in \mathsf{R}$ if (1) $x_2 = \_$ and the right-hand side of I is $\exists P.\top$ or $\exists P.A$; or (2) the right-hand side of I is P. An axiom I is *applicable to a pair of atoms* $P(x_1, x_2), A(x_2, x_3)$ if x_2 does not appear in other atoms of the query body, $x_3 = \_$ and I is of the form $C \sqsubseteq \exists P.A$. Below we define the set of rewriting rules for atoms in the query body. Let I be an inclusion assertion that is applicable to a sequence of query atoms g. The sequence of atoms obtained from g by applying I, denoted by $gr(g, I)$, is defined as follows:

(a) If $g = A(x_1, x_2)$ and $I = B \sqsubseteq A$, then $gr(g, I) = B(x_1, x_2)$;
(b) If $g = A(x_1, x_2)$ and $I = \exists P.\top \sqsubseteq A$, then $gr(g, I) = P(x_1, \_)$;
(c) If $g = A(x_1, x_2)$ and $I = \exists P.B \sqsubseteq A$, then $gr(g, I) = P(x_1, z_1), B(z_1, \_)$, where z_1 is a fresh variable;
(d) If $g = P(x_1, \_)$ and $I = A \sqsubseteq \exists P.\top$ or $I = A \sqsubseteq \exists P.B$, then $gr(g, I) = A(x_1, \_)$;
(e) If $g = P(x_1, x_2)$ and $I = R \sqsubseteq P$, then $gr(g, I) = R(x_1, x_2)$;
(f) If $g = P(x_1, x_2), A(x_2, \_)$ and $I = C \sqsubseteq \exists P.A$, then $gr(g, I) = C(x_1, \_)$;

We denote by $\mathsf{Rewrite}(q, \mathcal{T})$ the rewriting procedure that generates the perfect rewriting of q with respect to $\mathcal{T}$ (see Figure 1).

Algorithm 1: Algorithm $\mathsf{Rewrite}(q, \mathcal{T})$

```
Data: Conjunctive query $q$, TBox $\mathcal{T}$.
Result: Union of conjunctive queries $Q$.
$Q := \{\langle q, 1\rangle\}$;
repeat
    $Q' := Q$ ;
    foreach $\langle qr, x\rangle \in Q'$ do
        /* Reduction step                                    */
        if there exists $I \in \mathcal{T}$ such that I is not applicable to $qr$ then
            foreach set of atoms $S \subseteq body(qr)$ do
                if $S$ unify then
                    $Q := Q \cup \langle \mathsf{Reduce}(qr, S), 0\rangle$

        /* Rewriting step                                    */
        foreach axiom $I \in \mathcal{T}$ do
            if $I$ is applicable to $qr$ then
                $qr'$ := rewrite $qr$ according to $I$ ;
                $Q := Q \cup \langle qr', 1\rangle$
until $Q' = Q$;
$Q_{fin} := \{q \mid \langle q, 1\rangle \in Q\}$ ;
return $Q_{fin}$
```

Example 4. Consider applying the $\mathsf{Rewrite}$ procedure to a query $\boldsymbol{q}$ of the form $q(x) \leftarrow R(x,y), R(\_,y)$ over the TBox $\{A \sqsubseteq \exists R.\top\}$, where $A \in \mathsf{A}$ and $R \in \mathsf{R}$. In this query, the atoms $R(x,y)$ and $R(\_,y)$ unify, and executing $\mathsf{Reduce}(\boldsymbol{q}, \{R(x,y), R(\_,y)\})$ yields the atom $R(x,y)$. The variable y is now unbound, so can be replaced by "_" (a *don't care*). Note that the reduction step produces a query marked with '0' whilst the rewriting step marks queries with '1', and only queries marked with '1' are added to the output set. We adopt this approach to avoid redundancy in the output set, as a query marked with '0' is always contained in a query marked with '1'. Now, the axiom $\{A \sqsubseteq \exists R.\top\}$ can be applied to $R(x,\_)$, whereas, before the reduction process, it could not be applied to any atom of the query. Following this, the rewriting step reformulates the query to $q(x) \leftarrow A(x,\_)$ which is added to the output set. For more details on the rewriting procedure refer to [9, 10].

Now we show that each disjunct of the perfect rewriting of an atomic concept query with respect to an $\mathcal{EL}^{\ell in}$ TBox is of a special form called a *simple path conjunctive query*, defined below. We then define some technical lemmas which will be used in Section 4 for the rewriting of atomic concepts by means of a finite-state automaton.

Definition 1. *A conjunctive query $\boldsymbol{q}$ is a simple path conjunctive query (SPCQ) if body($\boldsymbol{q}$) is of one of the following forms: (i) $A(x_1,x_2)$;*
(ii) $P_1(x_1,y_1), P_2(y_1,y_2), \ldots, P_{n-1}(y_{n-2},y_{n-1}), P_n(y_{n-1},x_2)$; or
(iii) $P_1(x_1,y_1), P_2(y_1,y_2), \ldots, P_{n-1}(y_{n-2},y_{n-1}), P_n(y_{n-1},y_n), A(y_n,x_2)$, where: x_1, x_2 are terms; for each i, y_i is an existentially quantified variable and $y_i \neq y_{i+1}$; $n \geqslant 1$; $A \in \mathsf{A}$ and $P_1, \ldots, P_n \in \mathsf{R}$.

Note that query $\boldsymbol{q}$ in Example 4 is not an SPCQ. An SPCQ $head(\boldsymbol{q}) \leftarrow Z_1(x_0,x_1), \ldots, Z_n(x_{n-1},x_n)$ is equivalent to an RPQ of the form $head(\boldsymbol{q}) \leftarrow Z_1 \ldots Z_n(x_0,x_n)$; thus, throughout the paper we will use either the RPQ form or the CQ form of a SPCQ, whichever is more natural in the given context. For instance, given a SPCQ $\boldsymbol{q}$, with a little abuse of notation we have that $path(\boldsymbol{q})$ is $Z_1 \ldots Z_n$.

Given two paths p, p', we say that p' *contains* p, written $p \sqsubseteq p'$, if for each ABox $\mathcal{A}$ and for each tuple $\boldsymbol{a} = (a,a')$ it holds that, if $\mathcal{A} \models q() \leftarrow p(\boldsymbol{a})$ then $\mathcal{A} \models q() \leftarrow p'(\boldsymbol{a})$. Given a path p and an NFA $\mathcal{N}$ over $(\mathsf{R} \cup \mathsf{A})^*$ we say that $\mathcal{N}$ *contains* p, written $p \sqsubseteq \mathcal{N}$ if there exists some $\alpha \in L(\mathcal{N})$ such that $p \sqsubseteq \alpha$.

Lemma 1. *Given an SPCQ $\boldsymbol{q}$, an $\mathcal{EL}^{\ell in}$ TBox $\mathcal{T}$ and an axiom $\rho \in \mathcal{T}$ that is not applicable to body($\boldsymbol{q}$), for each set of atoms $S \subseteq$ body($\boldsymbol{q}$) that unify, ρ is also not applicable to body($\mathsf{Reduce}(q,S)$).*

Proof. (Sketch) We consider all the possible cases of ρ.

Case 1: ρ is of the type $B \sqsubseteq A$, $\exists P.\top$, $\exists R.B \sqsubseteq A$, $R \sqsubseteq P$. ρ is not applicable to $body(\boldsymbol{q})$ if an atom $A(x_1,x_2)$ or $P(x_1,x_2)$ are not in $body(\boldsymbol{q})$. If $A(x_1,x_2)$ or $P(x_1,x_2)$ is not in $body(\boldsymbol{q})$, then $A(x_1,x_2)$ or $P(x_1,x_2)$ is clearly not in $body(\mathsf{Reduce}(q,S))$ and the claim follows.

Case 2: ρ is of the type $A \sqsubseteq \exists R.\top$. ρ is not applicable to $body(\boldsymbol{q})$ if an atom $R(x_1,x_2)$ is not in $body(\boldsymbol{q})$, or if $R(x_1,x_2)$ is in $body(\boldsymbol{q})$ and $x_2 \neq \_$. If $R(x_1,x_2)$ is not in $body(\boldsymbol{q})$, then $R(x_1,x_2)$ is clearly not in $body(\mathsf{Reduce}(q,S))$. If $R(x_1,x_2)$ is in

$body(\boldsymbol{q})$ and $x_2 \neq \_$, since $\boldsymbol{q}$ is an SPCQ, this happens only if $R(x_1, x_2)$ is not the last atom of the path. The only way to have $x_2 = \_$ is to unify $R(x_1, x_2)$ with the atom at its right as it is the only atom that can have x_2. If there exists a unification, the reduction produces an atom $R(x_i, x_i)$, with $x_i \neq \_$ and the claim follows.

Case 3: ρ is of the type $A \sqsubseteq \exists R.B$. Since $\boldsymbol{q}$ is an SPCQ, ρ is not applicable to $body(\boldsymbol{q})$ if the atoms $R(x_1, x_2), B(x_2, x_3)$ are not in $body(\boldsymbol{q})$ or if $R(x_1, x_2)$ is in $body(\boldsymbol{q})$ and $x_2 \neq \_$. It is easy to see that this case is similar to the case where ρ is of the type $A \sqsubseteq \exists R.\top$.

Following from Lemma 1 we have that, to rewrite CQs with respect to $\mathcal{EL}^{\ell in}$ TBoxes, the reduction step generates queries that are never processed by the rewriting step. So, in our case, the reduction step is of no use. However, we keep it in the rewriting algorithm in order to handle future extensions to the ontology language.

Lemma 2. *Let $\mathcal{T}$ be an $\mathcal{EL}^{\ell in}$ TBox and $\boldsymbol{q}$ a CQ of the form $q(x) \leftarrow A(x, y)$, with $A \in \mathsf{A}$. If $\boldsymbol{q}_{rew} \in \mathsf{Rewrite}(\boldsymbol{q}, \mathcal{T})$, then $\boldsymbol{q}_{rew}$ is an SPCQ.*

Proof. (Sketch) Following from Lemma 1 we know that queries produced by the reduction step are never processed by the rewriting step, and so they are never marked with '1'. So, queries produced by the reduction step are never in the output set, and thus we can ignore the reduction step. The proof is then by induction on the set of queries that are produced after each rewriting step. We denote by $Q^{[i]}$ the set of the queries produced after the i-th iteration of the repeat loop in Algorithm 1.

BASE STEP. $Q^{[1]} = \{q(x) \leftarrow A(x, y)\}$ plus the queries obtained by the 1st rewriting step. The possible cases are the rewriting rules (a), (b) and (c) which generate SPCQs.

INDUCTIVE STEP. If $q \in Q^{[i+1]}$, then q is computed by applying a rewriting rule to a query in $Q^{[i]}$. The claim follows by induction if for each rewriting q to q', we have that q' is a SPCQ. If q is a SPCQ then we can identify a fixed set of possible rewriting cases, according to the rewriting rules. For each possible rewriting case, when q is rewritten to q', it is easy to see that if q is a SPCQ, then q' is a SPCQ.

It is important to note that this algorithm is not guaranteed to terminate on $\mathcal{EL}^{\ell in}$ knowledge bases; see e.g. the TBox of Example 3. This is because the algorithm essentially enumerates all rewritings produced by single applications of the reduction and the rewriting steps; when the algorithm is forced by the TBox to cycle on a set of assertions, it produces infinite branches and does not terminate. However, it is possible to capture such cyclic applications of the rewriting steps if we adopt a more expressive target language for the rewriting; this is the subject of the next two sections: in Section 4 we show how to encode the rewritings for an atomic query by means of a finite-state automaton; in Section 5 we present a technique, similar to that shown in [14] for OWL QL, that allows us to combine rewritings for atomic queries so as to obtain a CRPQ rewriting for CQs.

4 Rewriting for Atomic Concept Queries

In this section we show how to encode rewritings for atomic queries under $\mathcal{EL}^{\ell in}$ by means of a finite-state automaton; intuitively, the automaton is able to encode infinite sequences of rewriting steps executed according to the algorithm of Section 3. We concentrate on atomic queries having a concept atom in the body, since in the case of role atoms, this is done by a simple check on sequences of role inclusions in the TBox — see [13].

Definition 2. *Let $\mathcal{T}$ be an $\mathcal{EL}^{\ell in}$ TBox in normal form, Σ the alphabet $\mathsf{R} \cup \mathsf{A}$ and A a concept name appearing in $\mathcal{T}$. The NFA-rewriting of A with respect to $\mathcal{T}$, denoted by $\mathsf{NFA}_{A,\mathcal{T}}$, is the NFA over Σ of the form $(Q, \Sigma, \delta, q_0, F)$ defined as follows:*

(1) states S_A and SF_A are in Q, SF_A is in F, and transition (S_A, A, SF_A) is in δ;
(2) S_A is the initial state q_0, $S_\top$ is a final state;
(3) for each $B \in \mathsf{A}$ that appears in at least one concept or role inclusion axiom of $\mathcal{T}$, states S_B and SF_B are in Q,SF_B is in F, and transition (S_B, B, SF_B) is in δ;
(4) for each concept inclusion axiom $\rho \in \mathcal{T}$: (4.1) if ρ is of the form $B \sqsubseteq C$, where $B, C \in \mathsf{A}$, the transition (S_C, ϵ, S_B) is in δ; (4.2) if ρ is of the form $B \sqsubseteq \exists R.\top$, where $B \in \mathsf{A}$ and $R \in \mathsf{R}$, for each transition $(S_X, R, S_\top) \in \delta$, the transition (S_X, ϵ, S_B) is in δ; (4.3) if ρ is of the form $\exists R.\top \sqsubseteq B$, where $B \in \mathsf{A}$ and $R \in \mathsf{R}$, the transition $(S_B, R, S_\top)$ is in δ; (4.4) if ρ is the form $\exists R.D \sqsubseteq C$, where $C, D \in \mathsf{A}$ and $R \in \mathsf{R}$, the transition (S_C, R, S_D) is in δ; (4.5) if ρ is the form $C \sqsubseteq \exists R.D$, where $C, D \in \mathsf{A}$ and $R \in \mathsf{R}$, for any sequence of transitions starting from S_X that accepts the strings RD or R, the transition (S_X, ϵ, S_C) is in δ;
(5) for each role inclusion axiom $T \sqsubseteq S \in \mathcal{T}$ and each transition of the form $(S_C, S, S_B) \in \delta$, the transition (S_C, T, S_B) is in δ.

Example 5. Consider the TBox $\mathcal{T}$ defined by the following inclusion assertions: $\exists R.C \sqsubseteq \exists P.\top$, $\exists P.\top \sqsubseteq A$, $\exists P.\top \sqsubseteq B$, $\exists T.B \sqsubseteq C$ and $\exists S.A \sqsubseteq A$, where P, R, S, T are role names and A, B, C are concept names. Consider now the query $\boldsymbol{q} = q(x) \leftarrow A(x, y)$. First, we transform $\mathcal{T}$ into normal form, say $\mathcal{T}'$, by adding a fresh concept name X and by replacing $\exists R.C \sqsubseteq \exists P.\top$ by $\exists R.C \sqsubseteq X$ and $X \sqsubseteq \exists P.\top$. It is easy to see that $\mathsf{Rewrite}(\boldsymbol{q}, \mathcal{T}')$ runs indefinitely (for instance, we have an infinite loop when rule *(c)* is applied to the atom $A(x, y)$). Let us consider the NFA rewriting of A with respect to $\mathcal{T}'$. We construct $\mathsf{NFA}_{A,\mathcal{T}'}$ as follows: by *(3)* we have the transitions $(S_A, A, SF_A), (S_B, B, SF_B), (S_C, C, SF_C)$ and (S_X, X, SF_X); by *(4.3)* and the inclusion assertions $\exists P.\top \sqsubseteq A$ and $\exists P.\top \sqsubseteq B$, we have the transitions $(S_A, P, S_\top)$ and $(S_B, P, S_\top)$; by *(4.2)* and the inclusion assertion $X \sqsubseteq \exists P.\top$, we have the transitions (S_A, ϵ, S_X) and (S_B, ϵ, S_X); finally, by *(4.4)* and the inclusion assertions $\exists R.C \sqsubseteq X$, $\exists T.B \sqsubseteq C$ and $\exists S.A \sqsubseteq A$, we have the transitions (S_X, R, S_C), (S_C, T, S_B) and (S_A, S, S_A). The NFA $\mathsf{NFA}_{A,\mathcal{T}'}$ is illustrated in Figure 1. The language accepted by $\mathsf{NFA}_{A,\mathcal{T}'}$ can be described by the following regular expression: $(\exists S.)^*(A|X|(RT)^*(P|RC|RT(B|X)))$. It is easy to see that all the infinite outputs of $\mathsf{Rewrite}(\boldsymbol{q}, \mathcal{T}')$ are of the form $q(x) \leftarrow \mathsf{NFA}_{A,\mathcal{T}'}(x, y)$. For instance, some possible rewritings of $\boldsymbol{q}$ are:

$$\begin{array}{l} q(x) \leftarrow S(x, z_1), S(z_1, z_2), P(z_2, y) \\ q(x) \leftarrow S(x, z_1), S(z_1, z_2), A(z_2, y) \\ q(x) \leftarrow R(x, z_1), T(z_1, z_2), R(z_2, z_3), C(z_3, y) \end{array}$$

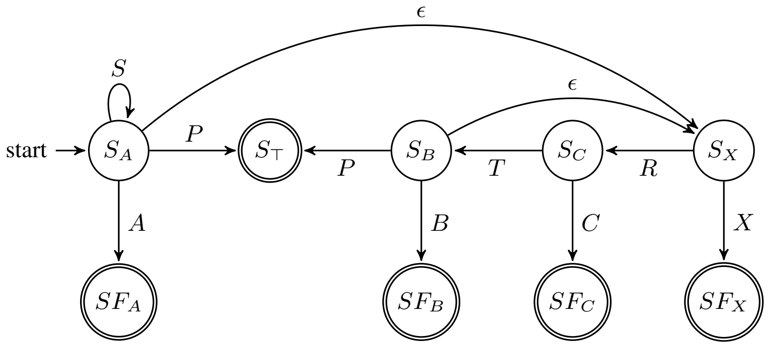

Fig. 1. NFA for Example 5.

It is easy to verify that each of these output queries is a SPCQ and each path is in $L(\mathsf{NFA}_{A,\mathcal{T}'})$.

Theorem 1. *Let $\mathcal{T}$ be an $\mathcal{EL}^{\ell in}$ TBox, and a concept A. We have that $\boldsymbol{q} \in$ $\mathsf{Rewrite}(q(x) \leftarrow A(x,y), \mathcal{T})$ if and only if $path(\boldsymbol{q}) \in L(\mathsf{NFA}_{A,\mathcal{T}})$.*

Proof. (Sketch) ($\Rightarrow$) The proof is by induction on the set of queries that are marked with '1' after each rewriting step, as the queries marked with '0' are not returned by the algorithm. We denote by $Q^{[i]}$ the set of queries marked with '1' after the i-th application of the rewriting step.

BASE STEP. $Q^{[0]} = \{\boldsymbol{q}\}$. By *(2)* we have that S_A is the initial state q_0 and by *(3)* we have the transition (S_A, A, SF_A) therefore $A \in L(\mathsf{NFA}_{A,\mathcal{T}})$ and the claim follows trivially.

INDUCTIVE STEP. From Lemma 1 we have that, if an axiom $\rho \in \mathcal{T}$ is not applicable to $body(\boldsymbol{q})$, for each set of atoms $S \subseteq body(\boldsymbol{q})$ that unify, ρ is also not applicable to $body(\mathsf{Reduce}(q, S))$. It follows that if a query q' is marked with '0' then there is no axiom in $\mathcal{T}$ that is applicable to q'. Thus, if $q \in Q^{[i+1]}$, then q is computed by applying a rewriting rule to a query that is marked with '1' at i-th application of the rewriting step, which is a query in $Q^{[i]}$. Suppose that for each $q \in Q^{[i]}$ we have that $\mathsf{Path}(q) \in L(\mathsf{NFA}_{A,\mathcal{T}})$, the claim follows by induction if for each rewriting q to q', we have that $\mathsf{Path}(q') \in L(\mathsf{NFA}_{A,\mathcal{T}})$. From Lemma 2 it follows that the body of each query marked with '1' is a simple path, thus we can identify a fixed set of possible rewriting cases. For each of possible rewriting case, when q is rewritten to q', there is a rule in the definition of $L(\mathsf{NFA}_{A,\mathcal{T}})$ such that $\mathsf{Path}(q) \in L(\mathsf{NFA}_{A,\mathcal{T}}) \rightarrow \mathsf{Path}(q') \in L(\mathsf{NFA}_{A,\mathcal{T}})$ is true.

($\Leftarrow$) The claim follows by induction on the construction rules of the $\mathsf{NFA}_{A,\mathcal{T}}$ starting from A, which correspond to all the possible rewriting steps of $\mathsf{Rewrite}(q(x) \leftarrow A(x,y), \mathcal{T})$.

Theorem 2. *Given an $\mathcal{EL}^{\ell in}$ TBox $\mathcal{T}$, concept A and a complex concept B, we have that $\mathcal{T} \models B \sqsubseteq A$ if and only if $B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}$.*

Proof. From Theorem 1 we have that $\mathsf{NFA}_{A,\mathcal{T}}$ is a perfect rewriting of A with respect to $\mathcal{T}$ and the claim follows.

5 CRPQ rewriting for $\mathcal{EL}^{\ell in}$

In this section, following the approach of [13, 14], we split the problem of rewriting CQs under $\mathcal{EL}^{\ell in}$ in two: we deal separately with the part of the TBox that does not have existential quantification on the right-hand side of assertions (that is, the part that when expanded does not produce any labelled null) and with the rest of the TBox. We make use of the algorithm for atomic queries presented in the previous section. Then, we put together the solutions devised for the two parts to produce a rewriting algorithm for CQs under $\mathcal{EL}^{\ell in}$.

Given an $\mathcal{EL}^{\ell in}$ knowledge base $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ in normal form, we can find all answers to a CQ q over this KB by evaluating q over the (possibly infinite) canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ which can be constructed using the chase procedure. We begin by defining the *standard model* $\mathcal{I}_{\mathcal{A}}$ of the ABox $\mathcal{A}$ as follows: *(1)* $\Delta^{\mathcal{I}_\mathcal{A}} = \mathsf{ind}(\mathcal{A})$; *(2)* $a^{\mathcal{I}_\mathcal{A}} = a$, for $a \in \mathsf{ind}(\mathcal{A})$; *(3)* $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$, for concept name A; *(4)* $P^{\mathcal{I}_\mathcal{A}} = \{(a, b) \mid P(a, b) \in \mathcal{A}\}$, for role name P. Then we take the standard model $\mathcal{I}_{\mathcal{A}}$ as $\mathcal{I}_0$ and apply inductively the following rules to obtain $\mathcal{I}_{k+1}$ from $\mathcal{I}_k$: *(a')* if $d \in A_1^{\mathcal{I}_k}$ and $A_1 \sqsubseteq A_2 \in \mathcal{T}$, then we add d to $A_2^{\mathcal{I}_{k+1}}$; *(b')* if $(d, d') \in R_1^{\mathcal{I}_k}$ and $R_1 \sqsubseteq R_2 \in \mathcal{T}$, then we add (d, d') to $R_2^{\mathcal{I}_{k+1}}$; *(c')* if $d \in (R.D)^{\mathcal{I}_k}$ and $\exists R.D \sqsubseteq A \in \mathcal{T}$, where D is a concept name or $\top$, then we add d to $A^{\mathcal{I}_{k+1}}$; *(d')* if $d \in A^{\mathcal{I}_k}$ and $A \sqsubseteq \exists R.D \in \mathcal{T}$, where D is a concept name or $\top$, then we take a *fresh* labelled null, d', and add d' to $D^{\mathcal{I}_{k+1}}$ and (d, d') to $R^{\mathcal{I}_{k+1}}$. The canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ constructed using rules (a'), (b'), (c') and (d') in a bottom-up fashion can alternatively be defined with the top-down approach illustrated in this section; this will be required for query rewriting in Section 5.2. There are two key observations that lead us to the alternative definition: first, fresh labelled nulls can only be added by applying (d'), and, second, if two labelled nulls, d_1 and d_2, are introduced by applying (d') with the same concept inclusion $A \sqsubseteq \exists R.D$, then the same rules will be applicable to d_1 and d_2 in the continuation of the chase procedure. So, each labelled null d' resulting from applying (d') to some $A \sqsubseteq \exists R.D$ on a domain element d can be identified with a pair of the form $(d, \exists R.D)$. Following from Theorem 2, for each concept $\exists R.D$ that appears at the RHS of a concept inclusion axiom in $\mathcal{T}$, we introduce a fresh symbol $w_{\exists R.D}$ that is a *witness* for $\exists R.D$ and define a generating relation $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ on the set of these witnesses together with $\mathsf{ind}(\mathcal{A})$ by taking:

- $a \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D}$, if $a \in \mathsf{ind}(\mathcal{A})$, $\mathcal{I}_{\mathcal{A}} \models B(a)$ and $B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}$,
- $w_{\exists S.B} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R.D}$ if $B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}$ and $A \sqsubseteq \exists R.D \in \mathcal{T}$,

where S is a role name. We point out that we are able to define a finite generating relation $\rightsquigarrow_{\mathcal{T},\mathcal{A}}$ for an $\mathcal{EL}^{\ell in}$ knowledge base $(\mathcal{T}, \mathcal{A})$ with the definition of $\mathsf{NFA}_{A,\mathcal{T}}$. In fact, $\mathsf{NFA}_{A,\mathcal{T}}$ captures all (possibly infinite) expressions B such that $\mathcal{T} \models B \sqsubseteq A$. This allows us to exploit the Tree-Witness rewriting technique in [14] (see Section 5.2).

A $path_{\rightsquigarrow_{\mathcal{T},\mathcal{A}}}$ σ is a finite sequence $aw_{\exists R_1.D_1} \dots w_{\exists R_n.D_n}$, $n \geqslant 0$, such that $a \in \mathsf{ind}(\mathcal{A})$ and, if $n > 0$, then $a \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R_1.D_1}$ and $w_{\exists R_i.D_i} \rightsquigarrow_{\mathcal{T},\mathcal{A}} w_{\exists R_{i+1}.D_{i+1}}$, for $i < n$. Thus, a path of the form $\sigma w_{\exists R.D}$ is also the fresh labelled null introduced by applying (d') to some $A \sqsubseteq \exists R.D$ on the domain element σ (and which corresponds to the pair $(\sigma, \exists R.D)$ mentioned above). Let us denote by $\mathsf{tail}(\sigma)$ the last element in σ; as we noted above, the last element in σ uniquely determines all the subsequent rule applications. The *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is defined by taking $\Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ to be the set of all $path_{\leadsto_{\mathcal{T},\mathcal{A}}}$ and taking: *(1)* $a^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} = a$, for $a \in \mathsf{ind}(\mathcal{A})$; *(2)* $A^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} = \{a \in \mathsf{ind}(A) \mid \mathcal{I}_{\mathcal{A}} \models B(a)$ and $B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}\} \cup \{\sigma w_{\exists R.D} \mid D \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}\}$, for each concept name A; *(3)* $P^{\mathcal{C}_{\mathcal{T},\mathcal{A}}} = \{(a,b) \mid \mathcal{I}_{\mathcal{A}} \models R(a,b)$ and $\mathcal{T} \models R \sqsubseteq P\} \cup \{(\sigma, \sigma w_{\exists R.D}) \mid \mathsf{tail}(\sigma) \leadsto_{\mathcal{T},\mathcal{A}} w_{\exists R.D}, \mathcal{T} \models R \sqsubseteq P\}$, for a role name P. We point out that, by the definition of rule (b'), we have that $\mathcal{T} \models R \sqsubseteq P$ only if there is a sequence of roles $R_0, \ldots, R_n$ such that $R_{i-1} \sqsubseteq R_i$ are in $\mathcal{T}$, for $1 \leqslant i \leqslant n$, and $R_n = P$. For proof refer to [13].

Given a CQ $\boldsymbol{q}$, we use the assertions of the TBox $\mathcal{T}$ to rewrite $\boldsymbol{q}$ into another query $\boldsymbol{q}'$ that returns, when evaluated over the data instance (ABox) $\mathcal{A}$, all the certain answers of $\boldsymbol{q}$ with respect to $(\mathcal{T}, \mathcal{A})$. Notice that the rewriting $\boldsymbol{q}'$ only depends on the TBox $\mathcal{T}$ and the given query $\boldsymbol{q}$; it is independent of the ABox $\mathcal{A}$. In query processing, therefore, we use $\mathcal{A}$ only in the final step, when the rewriting is evaluated on it.

We call a CQ $\boldsymbol{q}$ and a TBox $\mathcal{T}$ *CRPQ-rewritable* if there exists a CRPQ $\boldsymbol{q}'$ such that, for any ABox $\mathcal{A}$ and any tuple $\mathbf{a}$ of individuals in $\mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models q(\boldsymbol{a})$ if and only if $\mathcal{A} \models q'(\boldsymbol{a})$.

5.1 Rewriting for Flat $\mathcal{EL}^{\ell in}$

We first consider an important special case of *flat* $\mathcal{EL}^{\ell in}$ TBoxes that do not contain existential quantifiers on the right-hand side of concept inclusions. In other words, flat $\mathcal{EL}^{\ell in}$ in normal form can only contain concept and role inclusions of the form $A_1 \sqsubseteq A_2$, $\exists R.D \sqsubseteq A$ and $R_1 \sqsubseteq R_2$, for concept names A, A_1, A_2, role names R_1, R_2, and D a concept name or $\top$. Now let $\mathcal{T}$ be a flat $\mathcal{EL}^{\ell in}$ TBox, q a conjunctive query and a a tuple of individuals. Since $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is the canonical model for $(\mathcal{T}, \mathcal{A})$, we have that $(\mathcal{T}, \mathcal{A}) \models q(a)$ if and only if $q(a)$ is true in the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. The TBox is flat, the generating relation $\leadsto_{\mathcal{T},\mathcal{A}}$ is empty, the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ contains no labelled nulls, and so, by the definition of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, we have that:

- $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models A(a)$ if and only if $\mathcal{I}_{\mathcal{A}} \models B(a)$ and $\mathcal{T} \models B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}$, for some B,
- $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models P(a,b)$ if and only if $\mathcal{I}_{\mathcal{A}} \models R(a,b)$ and $T \models R \sqsubseteq P$, for some R.

For a CQ $\boldsymbol{q}$, we define now rewriting $\boldsymbol{q}_{ext}$ as a union of CRPQs which is the result of replacing every atom $A(z_1, z_2)$ in $\boldsymbol{q}$ with $A_{ext}(z_1, z_2)$ and every atom $P(z_1, z_2)$ in $\boldsymbol{q}$ with $P_{ext}(z_1, z_2)$, where $A_{ext}(u_1, u_2) = \mathsf{NFA}_{A,\mathcal{T}}(u_1, u_2)$ and $P_{ext}(u_1, u_2) = \bigcup\limits_{\mathcal{T} \models R \sqsubseteq P} R(u_1, u_2)$. This leads to the following results.

Proposition 1. *For all concept names A, role names P and individual names a and b we have: (1) $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models A(a)$ if and only if $\mathcal{I}_{\mathcal{A}} \models q() \leftarrow A_{ext}(a,a)$, (2) $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models P(a,b)$ if and only if $\mathcal{I}_{\mathcal{A}} \models q() \leftarrow P_{ext}(a,b)$.*

Proof. Follows immediately from the definitions of the formulas A_{ext} and P_{ext}.

Proposition 2. *For any CQ $\boldsymbol{q}$ and any flat $\mathcal{EL}^{\ell in}$ TBox $\mathcal{T}$, $\boldsymbol{q}_{ext}$ is the CRPQ rewriting of $\boldsymbol{q}$ with respect to $\mathcal{T}$.*

Proof. Follows immediately from the previous proposition and from the fact that each formula of the form $R_1(u_1, u_2) \cup \cdots \cup R_n(u_1, u_2)$ can be expressed as a regular path formula of the form $R_1|\cdots|R_n(u_1, u_2)$.

5.2 Tree-Witness Rewriting for Full $\mathcal{EL}^{\ell in}$

Following a divide and conquer strategy, we show how the process of constructing FO-rewritings can be split into two steps: the first step considers only the flat part of the TBox and uses the formulas $A_{ext}(u_1, u_2)$ and $P_{ext}(u_1, u_2)$ defined in Section 5.1; the second step (to be described below) takes account of the remaining part of the TBox, that is, inclusions of the form $A \sqsubseteq \exists R.D$. We first need some preliminary definitions.

Definition 3. *(H-completeness) Let T be a (not necessarily flat) $\mathcal{EL}^{\ell in}$ TBox. A simple ABox $\mathcal{A}$ is said to be* H-complete with respect to $\mathcal{T}$ *if, for all concept names $\mathcal{A}$ and role names P, we have:*

- $A(a) \in A$ *if* $\mathcal{I}_\mathcal{A} \models B(a)$ *and* $B \sqsubseteq \mathsf{NFA}_{A,\mathcal{T}}$, *for some* B,
- $P(a, b) \in A$ *if* $\mathcal{I}_\mathcal{A} \models R(a, b)$ *and* $T \models R \sqsubseteq P$, *for some* R.

Observe that, if an ABox A is H-complete with respect to $\mathcal{T}$, then the ABox part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ coincides with $\mathcal{I}_\mathcal{A}$. Thus, if $\mathcal{T}$ is flat then q itself is clearly the perfect rewriting of q and T over H-complete ABoxes. This leads to the following proposition.

Proposition 3. *If q' is the perfect rewriting of q and $\mathcal{T}$ over H-complete ABoxes, then q'_{ext} is the perfect rewriting of q with respect to T.*

So, to generate a CRPQ rewriting we can now focus on constructing rewritings over H-complete ABoxes. To achieve this, we reuse a technique adopted in [14] called *Tree Witness*. Suppose $\mathcal{T}$ is a $\mathcal{EL}^{\ell in}$ TBox in normal form. To compute certain answers to q over $(\mathcal{T}, \mathcal{A})$, for some $\mathcal{A}$, it is enough to find answers to q in the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. To do this, we have to check, for every tuple a of elements in $\mathsf{ind}(\mathcal{A})$, whether there exists a homomorphism from $q(a)$ to $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. Thus, as in the case of flat TBoxes, the answer variables take values from $\mathsf{ind}(A)$. However, the existentially quantified variables in q can be mapped both to $\mathsf{ind}(\mathcal{A})$ and to the labelled nulls in $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. In order to identify how the existential variables can be mapped to the anonymous part, it is sufficient to take a look at the tree-like structure of the generating relation. This technique allows us to rewrite a CQs with respect to $\mathcal{EL}^{\ell in}$ TBoxes over H-complete ABoxes; for details on the Tree Witness rewriting technique, consult [13, 14].

Theorem 3. *Let $\mathcal{T}$ be an $\mathcal{EL}^{\ell in}$ TBox and q a CQ. Let q' be the Tree Witness rewriting for q with respect to $\mathcal{T}$ over H-complete ABoxes. For any ABox $\mathcal{A}$ and any tuple a in $\mathsf{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models q(a)$ if and only if $\mathcal{I}_\mathcal{A} \models q'_{ext}(a)$.*

Corollary 1. *Let T be an $\mathcal{EL}^{\ell in}$ TBox and* **q** *a CQ. $\mathcal{T}$ is CRPQ-rewritable with respect to* **q**.

6 Discussion

In this paper we presented a rewriting algorithm for answering conjunctive queries under $\mathcal{EL}^{\ell in}$ knowledge bases. We showed how to encode rewritings of atomic queries with finite-state automata, and finally how to combine such automata in order to produce rewritings for the full language of CQs. We believe that our contribution sheds light on the possibilities of efficient query rewriting under DLs. Our rewriting technique achieves optimal data complexity (see below), and produces a "compact" rewriting without having to take the ABox into account; then the rewriting is evaluated on the ABox, which does not need to be expanded. We therefore argue that this pure rewriting approach is likely to be suitable for real-world cases, especially considering that the final CRPQ evaluation step can be performed by expressing the CRPQ in SPARQL 1.1.

Complexity. When considering query answering under ontologies, the most important asymptotic complexity measure is the so-called *data complexity*, i.e. the complexity w.r.t. the ABox $\mathcal{A}$. Our rewriting is evaluated on $\mathcal{A}$ in NLOGSPACE in data complexity [4], which coincides with the lower bound for CQ answering in $\mathcal{EL}^{\ell in}$ (see [17], where $\mathcal{EL}^{\ell in}$ is called DL-lite$^+$). In terms of *combined complexity*, i.e. the complexity w.r.t. $\mathcal{A}$, $\mathcal{T}$ and $\boldsymbol{q}$, we limit ourselves to a few consideration, leaving the issue to another work. Our rewriting, which is exponential in the query and the TBox, similarly to other approaches, has the advantage of being expressed in CRPQs (which implies the possibility of being easily translated in SPARQL, as above noted) whose evaluation is in NP [8]. Moreover our technique behaves as "pay-as-you-go" because in the case of atomic queries it produces a rewriting as an RPQ which can be evaluated in NLOGSPACE.

Related work. Query rewriting has been extensively employed in query answering under ontologies [10, 16, 15]. In particular, [17] presents a resolution-based query rewriting algorithm for DL-Lite+ ontologies (which is $\mathcal{EL}^{\ell in}$), with Linear Datalog as target language. In [3] the authors introduce a backward chaining mechanism to identify decidable classes of tuple-generating dependencies. Tractable query rewriting (in NLOGSPACE) for the DL-Lite family was presented in [9]; similarly, Rosati [18] used a rewriting algorithm for DL TBoxes expressed in the $\mathcal{EL}$ family of languages [1] to show that query answering in $\mathcal{EL}$ is PTIME-complete in data complexity. Other works [5, 6] study FO-rewritability of conjunctive queries in the presence of ontologies formulated in a description logic between $\mathcal{EL}$ and Horn-$\mathcal{SHIF}$, along with related query containment problems. In [12, 11] the authors propose an algorithm for computing FO rewritings of concept queries under $\mathcal{EL}$ TBoxes that is tailored towards efficient implementation. The tree-witness technique adopted in this paper is derived from that of [13, 14], which address query rewriting over $\mathcal{EL}$, $\mathcal{QL}$ and $\mathcal{RL}$, and propose the tree-witness approach to rewrite $\mathcal{QL}$. The complexity of answering CRPQs under DL-Lite and $\mathcal{EL}$ families is studied in [7].

Future work. We are extending our work in several directions. The most immediate ones are the following: *(1)* we plan to consider CRPQs as the language for queries, and devise a suitable rewriting algotithm; *(2)* we plan to consider inverse roles, and identify syntactic properties that would still guarantee rewritability of CQs into CRPQs; note that the ontology language so defined subsumes $\mathcal{QL}$; *(3)* we intend to include complex role chains and unions in the language, as in [15]. We already have results on *(1)* and *(3)*.

Acknowledgments. We thank Michael Zakharyaschev and Roman Kontchakov for precious discussions about this material.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI (2005)
2. Baader, F., Nutt, W.: Basic description logics. In: Description logic handbook. pp. 43–95 (2003)
3. Baget, J.F., Leclre, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9), 1620–1654 (2011)
4. Barcelo, P., Libkin, L., Lin, A.W., Wood, P.T.: Expressive languages for path queries over graph-structured data. TODS 37(4), 31 (2012)
5. Bienvenu, M., Hansen, P., Lutz, C., Wolter, F.: First order-rewritability and containment of conjunctive queries in horn description logics. In: DLOG (2016)
6. Bienvenu, M., Lutz, C., Wolter, F.: First-order rewritability of atomic queries in horn description logics. In: IJCAI (2013)
7. Bienvenu, M., Ortiz, M., Simkus, M.: Conjunctive regular path queries in lightweight description logics. In: IJCAI (2013)
8. Bourhis, P., Krötzsch, M., Rudolph, S.: Reasonable highly expressive query languages. In: IJCAI (2015)
9. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
10. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE (2011)
11. Hansen, P., Lutz, C., Seylan, I., Wolter, F.: Query rewriting under EL TBoxes: Efficient algorithms. In: Description Logics (2014)
12. Hansen, P., Lutz, C., Seylan, I., Wolter, F.: Efficient query rewriting in the description logic EL and beyond. In: IJCAI (2015)
13. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive Query Answering with OWL 2 QL. In: KR (2012)
14. Kontchakov, R., Zakharyaschev, M.: An introduction to description logics and query rewriting. Reasoning Web. Reasoning on the Web in the Big Data Era (2014)
15. Mosurovic, M., Krdzavac, N., Graves, H., Zakharyaschev, M.: A decidable extension of SROIQ with complex role chains and unions. JAIR 47, 809–851 (2013)
16. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient query answering for OWL 2. In: ISWC (2009)
17. Pérez-Urbina, H., Motik, B., Horrocks, I.: Rewriting conjunctive queries over description logic knowledge bases. In: Semantics in Data and Knowledge Bases (2008)
18. Rosati, R.: On conjunctive query answering in EL. In: DL (2007)