Conjunctive Queries for $\mathcal{EL}$ with Role Composition*

Markus Krötzsch and Sebastian Rudolph

Institute AIFB, Universität Karlsruhe, Germany
{mak|sru}@aifb.uni-karlsruhe.de

Abstract. $\mathcal{EL}^{++}$ is a rather expressive description logic (DL) that still admits polynomial time inferencing for many reasoning tasks. Conjunctive queries are an important means for expressive querying of DL knowledge bases. In this paper, we address the problem of computing conjunctive query entailment for $\mathcal{EL}^{++}$ knowledge bases. As it turns out, querying unrestricted $\mathcal{EL}^{++}$ is actually undecidable, but we identify restrictions under which query answering becomes decidable and even tractable. To the best of our knowledge, the presented algorithm is the first to answer conjunctive queries in a description logic that admits general role inclusion axioms.

1 Introduction

Conjunctive queries originated from research in relational databases [2], and, more recently, have been considered for expressive description logics (DLs) as well [3–7]. Algorithms for answering (extensions of) conjunctive queries in the expressive DL $\mathcal{SHIQ}$ have been discussed in [4, 5], but the first algorithm that supports queries for transitive roles was presented only very recently [7].

Modern DLs, however, allow for complex role inclusion axioms that encompass role composition and further generalise transitivity. To the best of our knowledge, no algorithms for answering conjunctive queries in those cases have been proposed yet. A relevant logic of this kind is $\mathcal{SROIQ}$ [8], the basic DL considered for OWL 1.1.[1] Another interesting DL that admits complex role inclusions is $\mathcal{EL}^{++}$ [9], which has been proposed as a rather expressive logic for which many inference tasks can be computed in polynomial time. In this paper, we present a novel algorithm for answering conjunctive queries in $\mathcal{EL}^{++}$, which is based on an automata-theoretic formulation of complex role inclusion axioms that was also found useful in reasoning with $\mathcal{SROIQ}$ [10, 8].

Our algorithm in particular allows us to derive a number of complexity results related to conjunctive query answering in $\mathcal{EL}^{++}$. We first show that conjunctive queries in $\mathcal{EL}^{++}$ are undecidable in general, and identify the $\mathcal{EL}^{++}$-fragment of $\mathcal{SROIQ}$ as an appropriate decidable sub-DL. Under some related restrictions of role inclusion axioms, we show that conjunctive query answering in general is PSpace-complete. Query answering for fixed knowledge bases (query complexity) is shown to be NP-complete, whereas for fixed queries (schema complexity) it is merely P-complete.

* Basic results of this work have first been published in [1].

[1] http://webont.org/owl/1.1/

2 Preliminaries

We assume the reader to be familiar with the basic notions of description logics (DLs). The DLs that we will encounter in this paper are $\mathcal{EL}^{++}$ [9] and, marginally, $\mathcal{SROIQ}$ [8]. A *signature* of DL consists of a finite set of *role names* **R**, a finite set of *individual names* **I**, and a finite set of *concept names* **C**, and we will use this notation throughout the paper. $\mathcal{EL}^{++}$ supports *nominals*, which we conveniently represent as follows: for any $a \in \mathbf{I}$, there is a concept $\{a\} \in \mathbf{C}$ such that $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$ (for any interpretation $\mathcal{I}$). As shown in [9], any $\mathcal{EL}^{++}$ knowledge base is equivalent to one in *normal form*, only containing the following axioms:

TBox:	$A \sqsubseteq C$	$A \sqcap B \sqsubseteq C$	$A \sqsubseteq \exists R.C$	$\exists R.A \sqsubseteq C$
RBox:	$R \sqsubseteq T$	$R \circ S \sqsubseteq T$		

where $A, B \in \mathbf{C} \cup \{\top\}$, $C \in \mathbf{C} \cup \{\bot\}$, and $R, S, T \in \mathbf{R}$. Note that ABox statements of the forms $C(a)$ and $R(a, b)$ are internalised into the TBox. The standard model theoretic semantics of $\mathcal{EL}^{++}$ can be found in [9]. Unless otherwise specified, the letters C, D, E in the remainder of this work always denote (arbitrary) concept names, and the letters R, S denote (arbitrary) role names. We do not consider concrete domains in this paper, but are confident that our results can be extended accordingly.

For *conjunctive queries*, we largely adopt the notation of [7] but directly allow for individuals in queries. Let **V** be a countable set of *variable names*. Given elements x, $y \in \mathbf{V} \cup \mathbf{I}$, a *concept atom* (*role atom*) is an expression $C(x)$ with $C \in \mathbf{C}$ ($R(x, y)$ with $R \in \mathbf{R}$). A *conjunctive query* q is a set of concept and role atoms, read as a conjunction of its elements. By $\mathsf{Var}(q)$ we denote the set of variables occurring in q. Consider an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$, and a function $\pi : \mathsf{Var}(q) \cup \mathbf{I} \to \Delta^{\mathcal{I}}$ such that $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathbf{I}$. We define

$$\mathcal{I}, \pi \models C(x) \text{ if } \pi(x) \in C^{\mathcal{I}}, \quad \text{and} \quad \mathcal{I}, \pi \models R(x, y) \text{ if } (\pi(x), \pi(y)) \in R^{\mathcal{I}}.$$

If there is some π such that $\mathcal{I}, \pi \models A$ for all atoms $A \in q$, we write $\mathcal{I} \models q$ and say that $\mathcal{I}$ *entails* q. We say that q is entailed by a knowledge base KB, denoted $KB \models q$, if all models of KB entail q.

3 Conjunctive Queries in $\mathcal{EL}^{++}$

We first investigate the complexity of conjunctive queries in general $\mathcal{EL}^{++}$ as defined in [9]. The following result might be mildly surprising, but is in fact closely related to similar results for logics with complex role expressions (see, e.g., [11]).

Theorem 1. *For an $\mathcal{EL}^{++}$ knowledge base KB and a conjunctive query q, the entailment problem $KB \models q$ is undecidable.*

Proof. The undecidable *Post correspondence problem* is described as follows: given two lists of words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over some alphabet Σ, is there a sequence of numbers $i_1, \ldots, i_k$ ($1 \leq i_j \leq n$) such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$? To reduce this problem to query entailment, we define a knowledge base KB. Consider the set of roles $\mathbf{R} = \{U_j | 1 \leq j \leq n\} \cup \{V_j | 1 \leq j \leq n\} \cup \{M_j | 1 \leq j \leq n\} \cup \{R_\sigma \mid \sigma \in \Sigma\} \cup \{U, V\}$. For each word

$u_j = \sigma_{j1} \dots \sigma_{jm}$ and corresponding role U_j, add an RBox statement $R_{\sigma_{j1}} \circ \dots \circ R_{\sigma_{j1}} \sqsubseteq U_j$, and likewise for words v_j. For each $j = 1, \dots, n$, define RBox statements $V_j \circ V \circ M_j \sqsubseteq V$ and $U_j \circ U \circ M_j \sqsubseteq U$. Moreover, for some concept C, add TBox statements of the form $C \sqsubseteq \exists S.C$ for all roles S of the form R_σ and M_j. Finally, add an ABox statement $C(a)$ for some individual a.

Now KB entails the query $\{U(a, x), V(a, x)\}$ iff there is a solution to the given Post correspondence problem. Indeed, it is easy to see that any model of KB implies $\{a\} \sqsubseteq \exists R_1 . \dots \exists R_l$ for any possible word $R_1 \dots R_l$ over the alphabet of R_j and M_j. The query is entailed iff some such word implies both U and V, which is the case exactly if a corresponding sequence was found, where the markers M_j ensure that both U and V have been generated from the same sequence. □

Corollary 1. *Checking class subsumptions in $\mathcal{EL}^{++}$ extended with inverse roles or role conjunctions is undecidable, even if those operators occur only in the concepts whose subsumption is checked.*

Proof. The proof of Theorem 1 is easily modified to check for concept subsumptions $\{a\} \sqsubseteq \exists U.\exists V^-.\{a\}$ or $\{a\} \sqsubseteq \exists (U \sqcap V).\{a\}$ instead of query entailment. □

Clearly, arbitrary role compositions are overly expressive when aiming for a decidable (or even tractable) logic that admits conjunctive queries. We thus restrict our attention to the fragment of $\mathcal{EL}^{++}$ that is in the (decidable) description logic $\mathcal{SROIQ}$ [8], and investigate its complexity with respect to conjunctive query answering.

Definition 1. *An $\mathcal{EL}^{++}$ RBox in normal form is* regular *if there is a strict partial order $\prec$ on $\mathbf{R}$ such that, for all role inclusion axioms $R_1 \sqsubseteq S$ and $R_1 \circ R_2 \sqsubseteq S$, we find $R_i \prec S$ or $R_i = S$ ($i = 1, 2$). An $\mathcal{EL}^{++}$ knowledge base is regular if it has a regular RBox.*

The existence of $\prec$ ensures that the role hierarchy does not contain cyclic dependencies other than through direct recursion of a single role.

4 Reasoning Automata for $\mathcal{EL}^{++}$

In this section, we describe the construction of an automaton that encodes certain concept subsumptions entailed by an $\mathcal{EL}^{++}$ knowledge base. The automaton itself is closely related to the reasoning algorithm given in [9], but the representation of entailments via nondeterministic finite automata (NFA) will be essential for the query answering algorithm in the following section. We describe an NFA $\mathcal{A}$ as a tuple $(Q_\mathcal{A}, \Sigma_\mathcal{A}, \delta_\mathcal{A}, i_\mathcal{A}, F_\mathcal{A})$, where $Q_\mathcal{A}$ is a finite set of states, $\Sigma_\mathcal{A}$ is a finite alphabet, $\delta_\mathcal{A} : Q_\mathcal{A} \times Q_\mathcal{A} \to 2^{\Sigma_\mathcal{A}}$ is a transition function that maps pairs of states to sets of alphabet symbols,[2] $i_\mathcal{A}$ is the initial state, and $F_\mathcal{A}$ is a set of final states.

Consider an $\mathcal{EL}^{++}$ knowledge base KB. Given a concept name $A \in \mathbf{C}$, we construct an NFA $\mathcal{A}_{KB}(A) = (Q, \Sigma, \delta, i, F)$ that computes superconcepts of A, where we omit the subscript if KB is clear from the context. Set $Q = F = \mathbf{C} \cup \{\top\}$, $\Sigma = \mathbf{C} \cup \mathbf{R} \cup \{\top, \bot\}$,

[2] A possibly more common definition is to map pairs of states and symbols to sets of states, but the above is more convenient for our purposes.

Table 1. Completion rules for constructing an NFA from an $\mathcal{EL}^{++}$ knowledge base KB.

(CR1) If $C' \in \delta(C,C)$, $C' \sqsubseteq D \in KB$, and $D \notin \delta(C,C)$ then $\delta(C,C) := \delta(C,C) \cup \{D\}$.
(CR2) If $C_1, C_2 \in \delta(C,C)$, $C_1 \sqcap C_2 \sqsubseteq D \in KB$, and $D \notin \delta(C,C)$ then $\delta(C,C) := \delta(C,C) \cup \{D\}$.
(CR3) If $C' \in \delta(C,C)$, $C' \sqsubseteq \exists R.D \in KB$, and $R \notin \delta(C,D)$ then $\delta(C,D) := \delta(C,D) \cup \{R\}$.
(CR4) If $R \in \delta(C,D)$, $D' \in \delta(D,D)$, $\exists R.D' \sqsubseteq E \in KB$, and $E \notin \delta(C,C)$ then $\delta(C,C) := \delta(C,C) \cup \{E\}$.
(CR5) If $R \in \delta(C,D)$, $\bot \in \delta(D,D)$, and $\bot \notin \delta(C,C)$ then $\delta(C,C) := \delta(C,C) \cup \{\bot\}$.
(CR6) If $\{a\} \in \delta(C,C) \cap \delta(D,D)$, and there are states $C_1, \ldots, C_n$ such that
- $C_1 \in \{C, \top, A\} \cup \{\{b\} \mid b \in \mathbf{I}\}$,
- $\delta(C_j, C_{j+1}) \neq \emptyset$ for all $j = 1, \ldots, n-1$,
- $C_n = D$,

and $\delta(D,D) \not\subseteq \delta(C,C)$ then $\delta(C,C) := \delta(C,C) \cup \delta(D,D)$.
(CR7) If $R \in \delta(C,D)$, $R \sqsubseteq S$, and $S \notin \delta(C,D)$ then $\delta(C,D) := \delta(C,D) \cup \{S\}$.
(CR8) If $R_1 \in \delta(C,D)$, $R_2 \in \delta(D,E)$, $R_1 \circ R_2 \sqsubseteq S$, and $S \notin \delta(C,E)$ then $\delta(C,E) := \delta(C,E) \cup \{S\}$.

and $i = A$. The transition function δ is initially defined as $\delta(C,C) := \{C, \top\}$ (for all $C \in Q$), and extended iteratively by applying the rules in Table 1. The rules correspond to completion rules in [9, Table 2], though the conditions for (CR6) are slightly relaxed, fixing a minor glitch in the original algorithm.

It is easy to see that the rules of Table 1 can be applied at most a polynomial number of times. The words accepted by $\mathcal{A}(A)$ are strings of concept and role names. For each such word w we inductively define a concept expression C_w as follows:

- if w is empty, then $C_w = \top$,
- if $w = Rv$ for some $R \in \mathbf{R}$ and word v, then $C_w = \exists R.(C_v)$,
- if $w = Cv$ for some $C \in \mathbf{C}$ and word v, then $C_w = C \sqcap C_v$.

For instance, the word $CRDES$ translates into $C_{CRDES} = C \sqcap \exists R.(D \sqcap E \sqcap \exists S.\top)$. Based on the close correspondence of the above rules to the derivation rules in [9], we can now establish the main correctness result for the automaton $\mathcal{A}(A)$.

Theorem 2. *Consider a knowledge base KB, concept A, and NFA $\mathcal{A}(A)$ as above, and let w be some word over the associated alphabet. Then $KB \models A \sqsubseteq C_w$ iff one of the following holds:*

- *$\mathcal{A}(A)$ accepts the word w, or*
- *there is a transition $\bot \in \delta(C,C)$ where $C = \top$, $C = A$, or $C = \{a\}$ for some individual a.*

In particular, $\mathcal{A}(A)$ can be used to check all subsumptions between A and some atomic concept B.

The second item of the theorem addresses the cases where A is inferred to be empty (i.e. inconsistent) or where the whole knowledge base is inconsistent, from which the subsumption trivially follows. While the above yields an alternative formulation of the $\mathcal{EL}^{++}$ reasoning algorithm presented in [9], it has the advantage that it also encodes all *paths* within the inferred models. This will be essential for our results in the next section. The following definition will be most convenient for this purpose.

Definition 2. *Consider a knowledge base KB, concepts* $A, B \in \mathbf{C}$*, and the NFA* $\mathcal{A}(A) = (Q, \Sigma, \delta, i, F)$*. The automaton* $\mathcal{A}_{KB}(A, B)$ *(or just* $\mathcal{A}(A, B)$*) is defined as* $(Q, \mathbf{R}, \delta, i, F')$ *where* $F' = \emptyset$ *whenever* $\bot \in \delta(A, A)$*, and* $F' = \{B\}$ *otherwise.*

The automaton $\mathcal{A}(A, B)$ normally accepts all words of roles $R_1, \ldots, R_n$ such that $A \sqsubseteq \exists R_1(\ldots \exists R_n.B \ldots)$ is a consequence of *KB*, with the border case where $n = 0$ and $KB \models A \sqsubseteq B$. Moreover, the language accepted by the NFA is empty whenever $A \sqsubseteq \bot$ has been inferred.

5 Deciding Conjunctive Queries for $\mathcal{EL}$

In this section, we present a nondeterministic algorithm that decides the entailment of a query q with respect to some regular consistent knowledge base *KB*. Here and in the following, we assume w.l.o.g. that *KB* does not entail $a \approx b$ (i.e. $\{a\} \equiv \{b\}$) for any $a, b \in \mathbf{I}$. Indeed, one can just replace all occurrences of b with a in this case, both within *KB* and within any query we wish to consider later on (and this case can be detected in polynomial time). Moreover, we assume that there is at least one individual in the language, i.e. $\mathbf{I} \neq \emptyset$. The algorithm constructs a so-called *proof graph* which establishes, for all interpretations $\mathcal{I}$ of *KB*, the existence of a suitable function π that shows query entailment.

Formally, a proof graph is a tuple (N, L, E) consisting of a set of nodes N, a labelling function $L : N \to \mathbf{C} \cup \{\top\}$, and a *partial* transition function $E : N \times N \to \mathbf{A}$, where $\mathbf{A}$ is the set of all NFA over the alphabet $\mathbf{C} \cup \{\top, \bot\} \cup \mathbf{R}$. The nodes of the proof graph are abstract representations of elements in the domain of some model of *KB*. The labels assign a concept to each node, and our algorithm ensures that the represented element is necessarily contained in the interpretation of this concept. Intuitively, the label of a node encodes all concept information relevant for the inferences used to show query entailment. A single concept name suffices for this purpose since (1) *KB* is in normal form and thus supplies concept names for all composite concept expressions such as conjunctions, and (2) $\mathcal{EL}^{++}$ does not allow inverse roles or number restrictions that could be used to infer additional information based on the relationship of an element to elements in the model. Finally, the transition function encodes paths in each model, which provide the basis for inferencing about role relationships between elements. It would be possible to adopt a more concrete representation for role paths (e.g. by guessing a single path), but our formulation reduces nondeterminism and eventually simplifies our investigation of algorithmic complexity.

The automaton of Definition 2 encodes concept subsumptions based on TBox and RBox. For deciding query entailment we also require automata that represent the content of the RBox.

Proposition 1. *Given a regular* $\mathcal{EL}^{++}$ *RBox, and some role* $R \in \mathbf{R}$*, there is an NFA* $\mathcal{A}(R)$ *over the alphabet* $\mathbf{R}$ *which accepts a word* $R_1 \ldots R_n$ *iff* $R_1 \circ \ldots \circ R_n \sqsubseteq R$ *is a consequence of every* $\mathcal{EL}^{++}$ *knowledge base with the given RBox.*

Proof. One possible construction for the required automaton is discussed in [8]. Intuitively, the RBox can be understood as a grammar for a regular language, for which an automaton can be constructed in a canonical way. □

The required construction of $\mathcal{A}(R)$ might be exponential for some RBoxes. In [10], restrictions have been discussed that prevent this blow-up, leading to NFA of only polynomial size w.r.t. the RBox. Accordingly, an RBox is *simple* whenever, for all axioms of the form $R_1 \circ S \sqsubseteq S$, $S \circ R_2 \sqsubseteq S$, the RBox does not contain a common subrole R of R_1 and R_2 for which there is an axiom of the form $R \circ S' \sqsubseteq R'$ or $S' \circ R \sqsubseteq R'$. We will usually consider only such simple RBoxes whenever the size of the constructed automata matters.

We are now ready to present the algorithm. It proceeds in various consecutive steps:

Query factorisation. The algorithm nondeterministically selects a variable $x \in \mathsf{Var}(q)$ and some element $e \in \mathsf{Var}(q) \cup \mathbf{I}$, and replaces all occurrences of x in q with e. This step can be executed an arbitrary number of times (including zero).

Proof graph initialisation. The proof graph (N, L, E) is initialised by setting $N := \mathbf{I} \cup \mathsf{Var}(q)$. L is initialised by $L(a) := \{a\}$ for each $a \in \mathbf{I}$. For each $x \in \mathsf{Var}(q)$, the algorithm nondeterministically selects a label $L(x) \in \mathbf{C} \cup \{\top\}$. Finally, E is initialised by setting $E(n, a) := \mathcal{A}(L(n), L(a))$ for each $n \in N$, $a \in \mathbf{I}$. A node $m \in N$ is *reachable* if there is some node $n \in N$ such that $E(n, m)$ is defined, and *unreachable* otherwise (recall that E is a partial function). Thus exactly the nominal nodes are reachable by the initialisation of E. Now as long as there is some unreachable node $x \in \mathsf{Var}(q)$, the algorithm nondeterministically selects one such x and some node $n \in N$ that is reachable, and sets $E(n, x) := \mathcal{A}(L(n), L(x))$. After this procedure, the graph (N, L, E) is such that all nodes are reachable. Finally, the algorithm checks whether any of the automata $E(n, m)$ with $n \in N$ and $m \in \mathsf{Var}(q)$ accepts the empty language, and aborts with failure if this is the case.

Checking concept entailment. For all concept atoms $C(n) \in q$ $(n \in N)$, the algorithm checks whether $L(n) \models C$ with respect to KB.

For the remaining steps of the algorithm, some preliminary definitions and observations are needed. The automata $E(n, m)$ of the proof graph represent chains of existential role restrictions that exist within any model. If $m \in \mathsf{Var}(q)$, then the automaton encodes many possible ways of constructing an element that belongs to the interpretation of $L(m)$ in each model. The role automata $\mathcal{A}(R)$ in turn encode possible chains of roles that suffice to establish role R along some such path. To show that an atom $R(n, m)$ is entailed, one thus has to check whether the automata $E(n, m)$ and $\mathcal{A}(R)$ have a non-empty intersection language. Two issues must be taken into account. First, not every pair of nodes is linked by an edge $E(n, m)$, so one might have to look for a longer path of edges and check non-emptiness of its intersection with $\mathcal{A}(R)$. Second, there might be several role atoms that affect the path between n and m. Since all of them must be taken into account, one either needs to check intersections of many languages concurrently, or to retain the restrictions imposed by one role atom before treating further atoms.

Proposition 2. *For every pair of nodes $n, m \in N$, there is a unique* shortest *connecting path $n_0 = n, n_1, \ldots, n_k = m$ with $n_i \in N$ and $E(n_i, n_{i+1})$ defined. This path can be computed by a deterministic algorithm in polynomial time.*

Proof. By construction of (N, L, E), there is a (necessarily shortest) path of length $k = 1$ whenever $m \in \mathbf{I}$. Likewise, if $m \in \mathsf{Var}(q)$, there is a shortest ("generating") path $m_0, \ldots, m_l = m$ from some element $m_0 \in \mathbf{I}$. The shortest path from n to m is found immediately if $n = m_i$ for some $i < l$. Otherwise, the shortest path has an additional initial segment $E(n, m_0)$. Clearly, all of this can be determined in polynomial time. □

Now any role atom in the query should span over some existing path, and we need to check whether this path suffices to establish the required role. To do this, we nondeterministically split the role automaton into parts that are distributed along the path.

Definition 3. *Consider an NFA $\mathcal{A} = (Q, \Sigma, \delta, i, \{f\})$. A* split *of $\mathcal{A}$ into k parts is given by NFA $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with $\mathcal{A}_j$ of the form $(Q, \Sigma, \delta, q_{j-1}, \{q_j\})$ such that $q_0 = i$, $q_k = f$, and $q_j \in Q$ for all $j = 1, \ldots, k-1$.*

It is easy to see that, if each split automaton $\mathcal{A}_j$ accepts some word w_j, we find that $w_1 \ldots w_k$ is accepted by $\mathcal{A}$. Likewise, any word accepted by $\mathcal{A}$ is also accepted in this sense by some split of $\mathcal{A}$. Since the combination of any split in general accepts less words than $\mathcal{A}$, splitting an NFA usually involves some don't-know nondeterminism. We can now proceed with the final steps of the algorithm.

Splitting of role automata. For each role atom $R(n, m)$ within the query, the algorithm computes the shortest path $n = n_0, \ldots, n_k = m$ from n to m. Next, it splits the NFA $\mathcal{A}(R)$ into k automata $\mathcal{A}(R(n, m), n_0, n_1), \ldots, \mathcal{A}(R(n, m), n_{k-1}, n_k)$, and aborts with failure if the language accepted by any of the split automata is empty.

Check role entailment. Finally, for each $n, m \in N$ with $E(n, m)$ defined, the algorithm executes the following checks:

(a) If $m \in \mathbf{I}$, it checks for each split automaton of the form $\mathcal{A}(F, n, m)$ whether there is a word accepted by $\mathcal{A}(F, n, m)$ and by the edge automaton $E(n, m)$.
(b) If $m \in \mathsf{Var}(q)$, it checks whether there is a (single) word that is accepted by all split automata of the form $\mathcal{A}(F, n, m)$ and by the edge automaton $E(n, m)$.

If all those checks succeed (i.e. if the required words exist), the algorithm confirms the entailment of the query (we say that it *accepts* the query). Else it terminates with failure.

Intuitively, the above checks show the existence of suitable role paths in any model, represented by accepted words. In case (a), only pairwise comparisons are needed, since different role paths may still lead to the same element represented by the individual $m \in \mathbf{I}$. But in case (b), the identity of the domain element represented by $m \in \mathsf{Var}(q)$ depends on the chosen role path, and it must be ensured that all conditions refer to the same path (and thus to the same element).

The above conditions could also be stated as emptiness problems for the automata's intersection languages, but this tends to make the verbal description more ambiguous.

6 Correctness of the Algorithm

We now prove soundness and completeness of the algorithm presented in Section 5.

Proposition 3. *Consider a regular consistent $\mathcal{EL}^{++}$ knowledge base KB and a conjunctive query q. If the algorithm of Section 5 accepts q, then indeed KB $\models$ q.*

Proof. We use the notation from Section 5 to denote structures computed by the algorithm. When terminating successfully, the algorithm has computed the following:

- A proof graph (N, L, E),
- For each role atom $R(m, n) \in q$, a k-split $\mathcal{A}(R(n, m), n_0, n_1), \ldots, \mathcal{A}(R(n, m), n_{k-1}, n_k)$ of the NFA $\mathcal{A}(R)$, where k is the length of the shortest path from n to m in (N, L, E).

In the following, let $\mathcal{I}$ be some model of KB. To show $KB \models q$, we need to provide a mapping π as in Section 2 for $\mathcal{I}$. Since $\mathcal{I}$ is arbitrary, this shows the entailment of q. We can derive π from the proof graph, and then show its correctness based on the conditions checked by the algorithm.

When factorising the query, the algorithm replaces variables by individual names or by other variables. This is no problem: whenever a query q' is obtained from q by uniformly replacing a variable $x \in \mathsf{Var}(q)$ by an individual $a \in \mathbf{I}$ (or variable $y \in \mathsf{Var}(q)$), we have that $KB \models q'$ implies $KB \models q$. Indeed, any mapping π' for q' can be extended to a suitable mapping π for q by setting $\pi(x) := a^{\mathcal{I}}$ ($\pi(x) := y^{\mathcal{I}}$). Thus we can assume w.l.o.g. that all variables $x \in \mathsf{Var}(q)$ also occur as nodes in the proof graph, i.e. $x \in N$.

When checking role entailment, the algorithm checks non-emptiness of the intersection languages of the automata $E(n, m)$, and one/all split automata $\mathcal{A}(F, n, m)$, for each $n, m \in N$ with $E(n, m)$ defined. Thus for any pair $n \in N$, $m \in \mathsf{Var}(q)$, there is some word w accepted by *all* of the given automata. Choose one such word $w(n, m)$. By the definition of $\mathcal{A}(R)$ and the split automata, $w(n, m)$ is a word over $\mathbf{R}$, and we can assume this to be the case even when no split automata (but just the single edge automaton) are considered for the given edge from n to m. $E(n, m)$ in turn is of the form $\mathcal{A}(L(n), L(m))$ (Definition 2) for the selected class names $L(n)$ and $L(m)$ of the proof graph.

Now by Theorem 2, the construction of Definition 2, and the fact that KB is consistent, it is easy to see that $E(n, m)$ accepts the word $w(n, m) = R_1 \ldots R_l$ iff $KB \models L(n) \sqsubseteq \exists R_1. \ldots \exists R_l.L(m)$. We employ this fact to inductively construct a mapping π.

When constructing the transition function of the proof graph, the algorithm has defined labels $L(x)$ for all $x \in \mathsf{Var}(q)$, and we will retrace this process to construct π. We claim that the following construction ensures that, whenever a node $n \in N$ is reachable, $\pi(n)$ has been assigned a unique value such that $\pi(n) \in L(n)^{\mathcal{I}}$. For starting the induction, set $\pi(a) := a^{\mathcal{I}}$ for each $a \in \mathbf{I}$ (which is necessarily reachable and clearly satisfies $\pi(a) \in L(a)^{\mathcal{I}} = \{a\}^{\mathcal{I}}$). Now assume that in one step the algorithm selected some $x \in \mathsf{Var}(q)$ that was not reachable yet, and node $n \in N$ which is reachable. As noted above, $KB \models L(n) \sqsubseteq \exists R_1. \ldots \exists R_l.L(x)$ where $w(n, x) = R_1 \ldots R_l$, and hence there is an element $e \in L(x)^{\mathcal{I}}$ such that $(\pi(n), e) \in R_1^{\mathcal{I}} \circ \ldots \circ R_l^{\mathcal{I}}$ (where $\circ$ denotes forward composition of binary relations). Pick one such e and set $\pi(x) := e$. It is easy to see that the claim of the induction is satisfied.

The algorithm has verified that $L(n) \sqsubseteq C$ holds for each $C(n) \in q$ (using standard polynomial time reasoning for $\mathcal{EL}^{++}$), so we find $\pi(n) \in C^{\mathcal{I}}$. It remains to show that a similar claim holds for all binary query atoms. Thus consider some role atom $R(n, m) \in q$, and let $n = n_0, \ldots, n_k = m$ denote the shortest path in the proof graph used to split the role automaton. So far, we have defined $w(n_i, n_{i+1})$ only for cases where

$n_{i+1} \in \mathsf{Var}(q)$. By a slight overloading of notation, we now let $w(n_i, n_{i+1})$ for $n_{i+1} \in \mathbf{I}$ denote some word accepted by the intersection of $E(n_i, n_{i+1})$ and the specific split automaton $\mathcal{A}(R(n,m), n_i, n_{i+1})$, which must exist as the algorithms must have verified non-emptiness of the intersection language. Assuming that $w(n_i, n_{i+1}) = S_1 \dots S_l$, we note that this still entails $KB \models L(n_1) \sqsubseteq \exists S_1 . \dots \exists S_l . L(n_{i+1})$. Since $n_{i+1} \in \mathbf{I}$, this actually shows that $(\pi(n_i), \pi(n_{i+1})) \in S_1^{\mathcal{I}} \circ \dots S_l^{\mathcal{I}}$.

The word $w = w(n_0, n_1) \dots w(n_{k-1}, n_k)$ is accepted by $\mathcal{A}(R)$, which is clear from the construction in Definition 3 as the parts $w(n_i, n_{i+1})$ are accepted by the respective split automata. Assume that $w = R_1 \dots R_k$. We conclude $(\pi(n), \pi(m)) \in R_1^{\mathcal{I}} \circ \dots \circ R_k^{\mathcal{I}}$ from the construction of π and the above observations for the case of edges connecting to individual elements. Thus by Proposition 1 we have $(\pi(n), \pi(m)) \in R^{\mathcal{I}}$ as required. □

It remains to show that the algorithm is also complete. This is done by demonstrating that there are suitable nondeterministic choices that enable the algorithm to accept a query whenever it is entailed. To guide those choices, we first construct a canonical model for some knowledge base.

Consider a regular consistent $\mathcal{EL}^{++}$ knowledge base KB as before. We now provide an iterative construction of a model $\mathcal{I}$ of KB. Our goal is to obtain a concise definition of a suitable *canonical model*, so it is no matter of concern that the given construction does not terminate after finitely many steps.

Table 2. Closure rules for an interpretation $\mathcal{I}$ w.r.t. some knowledge base KB. In general, we assume that $C, D \in \mathbf{C} \cup \{\top, \bot\}$ and $R_1, R_2, S \in \mathbf{R}$.

$$(1)\quad \frac{\delta \in C^{\mathcal{I}} \qquad KB \models C \sqsubseteq D}{D^{\mathcal{I}} := D^{\mathcal{I}} \cup \{\delta\}}$$

$$(2)\quad \frac{\delta \in C^{\mathcal{I}} \qquad KB \models C \sqsubseteq \exists R.D \qquad KB \not\models D \sqsubseteq \{a\} \text{ for any } a \in \mathbf{I}}{\Delta_{\mathcal{I}} := \Delta_{\mathcal{I}} \cup \{\epsilon\} \qquad R^{\mathcal{I}} := R^{\mathcal{I}} \cup \{(\delta, \epsilon)\} \qquad D^{\mathcal{I}} := D^{\mathcal{I}} \cup \{\epsilon\}} \text{ where } \epsilon = \epsilon_{\delta, C \sqsubseteq \exists R.D}$$

$$(3)\quad \frac{\delta \in C^{\mathcal{I}} \qquad KB \models C \sqsubseteq \exists R.D \qquad KB \models D \sqsubseteq \{a\} \text{ for some } a \in \mathbf{I}}{R^{\mathcal{I}} := R^{\mathcal{I}} \cup \{(\delta, a)\}}$$

$$(4)\quad \frac{(\delta, \epsilon) \in R^{\mathcal{I}} \qquad R \sqsubseteq S \in KB}{S^{\mathcal{I}} := S^{\mathcal{I}} \cup \{(\delta, \epsilon)\}} \qquad (5)\quad \frac{(\delta, \epsilon) \in R_1^{\mathcal{I}} \qquad (\epsilon, \gamma) \in R_2^{\mathcal{I}} \qquad R_1 \circ R_2 \sqsubseteq S \in KB}{S^{\mathcal{I}} := S^{\mathcal{I}} \cup \{(\delta, \gamma)\}}$$

To simplify our arguments, we adopt a naming scheme for potential elements of the domain of $\mathcal{I}$. Let Δ be the smallest set such that $\mathbf{I} \subseteq \Delta$ and, for any $\delta \in \Delta$, $C, D \in \mathbf{C}$, and $R \in \mathbf{R}$, we find that $\epsilon_{\delta, C \sqsubseteq \exists R.D} \in \Delta$. We will define $\mathcal{I}$ such that $\Delta_{\mathcal{I}} \subseteq \Delta$.

For any two interpretations $\mathcal{J}_1$ and $\mathcal{J}_2$ of KB, we say that $\mathcal{J}_1$ is *smaller* than $\mathcal{J}_2$ if, for any $F \in \mathbf{C} \cup \mathbf{R} \cup \{\top\}$, $F^{\mathcal{J}_1} \subseteq F^{\mathcal{J}_2}$. The interpretation $\mathcal{I}$ is defined to be the smallest interpretation that satisfies the following:

(i) $\Delta_{\mathcal{I}} \subseteq \Delta$,
(ii) $\{a\}^{\mathcal{I}} := a$ for all $a \in \mathbf{I}$, and
(iii) $\mathcal{I}$ is closed under the rules of Table 2.

It is easy to see that this smallest interpretation exists: just consider all interpretations satisfying conditions (i) and (ii), ordered by the "smaller than" relation defined above, which clearly yields a *complete lattice* with least upper bounds given by taking the (pointwise) unions of interpretation domains and extensions. Given an element $\mathcal{J}$ of this set, an interpretation $f(\mathcal{J})$ is defined as the result of exhaustively applying all rules of Table 2 whose premisses are satisfied by $\mathcal{J}$. The construction is easily seen to be monotonic, and hence indeed has a least fixed point $\mathcal{I}$ [12, Theorem 8.22].

The rules of Table 2 have the special property that each individual is "initialised" with at most one concept name. Formally, we define for each element $\delta \in \Delta_{\mathcal{I}}$ a concept name $\iota(\delta)$ as follows:

- if $\delta \in \mathbf{I}$, $\iota(\delta) \coloneqq \{\delta\}$,
- if $\delta = \epsilon_{\delta',C\sqsubseteq\exists R.D}$ for some $\delta' \in \Delta_{\mathcal{I}}$, $C, D \in \mathbf{C}$, $R \in \mathbf{R}$, then $\iota(\delta) \coloneqq D$.

Note that the above cases are indeed exhaustive and mutually exclusive.

Lemma 1. *The interpretation $\mathcal{I}$ as constructed above is a model of KB.*

Proof. First note that the domain of $\mathcal{I}$ is non-empty since we assume the existence of at least one individual. We have to check that all axioms of *KB* are indeed satisfied. For axioms of the form $C \sqsubseteq \exists R.D$ this is obvious by rules (2) and (3) of Table 2. Similarly, all role inclusion axioms are directly accounted for by rules (4) and (5).

So it remains to show that axioms Φ of the forms $C \sqsubseteq D$, $\exists R.C \sqsubseteq D$, and $C_1 \sqcap C_2 \sqsubseteq D$ are satisfied. Obviously, whenever $\delta \in C^{\mathcal{I}}$ ($\delta \in \exists R.C^{\mathcal{I}}$) for some $C \in \mathbf{C}$ (and $R \in \mathbf{R}$), we find $KB \models \iota(\delta) \sqsubseteq C$ ($KB \models \iota(\delta) \sqsubseteq \exists R.C$). We conclude that, whenever the premise of some axiom Φ as above is satisfied for δ, then it is entailed by $\iota(\delta)$, and so its conclusion D is a direct consequence of $\iota(\delta)$ under *KB*. Thus Φ is satisfied by rule (1). □

Proposition 4. *Consider a regular consistent $\mathcal{EL}^{++}$ knowledge base KB and a conjunctive query q. If $KB \models q$, then there is a sequence of nondeterministic choices for the algorithm of Section 5 such that it accepts q.*

Proof. Consider the canonical model $\mathcal{I}$ as constructed above. Since $KB \models q$ and $\mathcal{I} \models KB$, there is some mapping π such that $\mathcal{I}, \pi \models q$. We will use π to guide the algorithm.

In the query factorisation step, a variable $x \in \mathsf{Var}(q)$ is replaced by $n \in \mathsf{Var}(q) \cup \mathbf{I}$ whenever $\pi(x) = \pi(n)$. For the proof graph initialisation, we choose the labelling L of the proof graph by setting $L(e) \coloneqq \iota(\pi(e))$. As we have argued above, $\delta \in C^{\mathcal{I}}$ iff $KB \models \iota(\delta) \sqsubseteq C$, and hence we conclude that $\pi(e) \in C^{\mathcal{I}}$ implies that $KB \models L(e) \sqsubseteq C$ for all $e \in \mathbf{I} \cup \mathsf{Var}(q)$. Thus all unary atoms of q are accepted by the algorithm.

Continuing with the construction of edges in the proof graph, we first observe some important basic properties of the canonical model.

Property 1. For any element $\delta \in \Delta_{\mathcal{I}}$ that is not an individual $\delta \notin \mathbf{I}$, there is a unique chain of elements $\delta_0 \ldots \delta_k = \delta$ and role names $R_0, \ldots, R_{k-1} \in \mathbf{R}$, such that $\delta_0 \in \mathbf{I}$ and, for all $i = 1, \ldots, k$, $\delta_i \in \Delta_{\mathcal{I}}$ is of the form $\delta_{\epsilon,C\sqsubseteq R.D}$ with $\epsilon = \delta_{i-1}$ and $R = R_{i-1}$. This is easily verified by observing that any δ of the given form must have been entailed by rule (2), and by applying a simple induction on the depth of this entailment. In this case, we say that δ_i *generates* δ via the roles $R_i \ldots R_k$ ($i = 0, \ldots, k$).

Property 2. Consider elements $\delta, \epsilon \in \Delta_{\mathcal{I}}$ such that δ generates ϵ via the roles $R_0 \dots R_k$. Then $\iota(\delta) \sqsubseteq \exists R_0.(\dots \exists R_k.\iota(\epsilon) \dots)$. This is obvious by another simple inductive argument that utilises the preconditions of the applications of rule (3).

Property 3. For any $(\delta, \epsilon) \in R^{\mathcal{I}}$, there is a chain of elements $\delta = \delta_0 \dots \delta_k = \epsilon$ and role names R_i ($i = 0, \dots, k-1$), such that

- $(\delta_i, \delta_{i+1}) \in R_i^{\mathcal{I}}$ is directly entailed by one of rules (2) and (3), and
- $R_0 \circ \dots \circ R_{k-1} \sqsubseteq R$ is a consequence of *KB*.

We show this by an inductive argument as follows: for the base case, assume that $(\delta, \epsilon) \in R^{\mathcal{I}}$ follows from rule (2) or (3). Then the above condition clearly holds. For the induction step, assume that $(\delta, \epsilon) \in R^{\mathcal{I}}$ follows by applying rule (5) to $R_1 \circ R_2 \sqsubseteq R$, and that the claim holds for the statements $(\delta, \delta_j) \in R_1^{\mathcal{I}}$ and $(\delta_j, \epsilon) \in R_2^{\mathcal{I}}$. We easily can construct from these assumptions a suitable chain of elements from the chains postulated for R_1 and R_2. Similarly, the second condition of the claim follows from the assumption that $R_1 \circ R_2 \sqsubseteq R$ and the induction hypothesis. Rule (4) is treated analogously.

Now in each step of the generation of the edges E of the proof graph, the algorithm needs to pick some (unreachable) $x \in \mathsf{Var}(q)$ and some reachable node n. By Property 1 above, there is a unique generating chain for each $\pi(x)$ where x is not reachable within the proof graph yet. Moreover, since the chain of Property 1 is unique and shortest, it is also acyclic. Hence there is some unreachable x such that $\pi(x)$ is not generated by any element of the form $\pi(y)$ with y unreachable. Pick one such element x. Finally select one element $n \in \mathbf{I} \cup \mathsf{Var}(q)$ such that $\pi(n)$ generates $\pi(x)$, and such that there is no element m for which $\pi(m)$ generates $\pi(x)$ and $\pi(n)$ generates $\pi(m)$. Construct an edge $E(m, x)$.

Now for any elements n and m of the query, with $m \in \mathsf{Var}(q)$ and $E(n, m)$ defined, the automaton $E(n, m)$ accepts a non-empty language. This is seen by combining Property 2 with Theorem 2, where the second case of the theorem is excluded since *KB* is consistent. The algorithm's checks for non-emptiness of these languages thus succeed.

The algorithm now has completed the proof graph construction, and the selection of split automata is required next. For all query atoms $R(n, m)$, we find that $(\pi(n), \pi(m)) \in R^{\mathcal{I}}$, and thus we can apply Property 3 to obtain a respective chain of elements and role names, which we denote as $\delta_0 \dots \delta_k$ and $R_0 \dots R_{k-1}$ in the remainder of this proof.

Let $j > 0$ denote the largest index of $\delta_0 \dots \delta_k$, such that δ_j is of the form $\pi(e_1)$ for some $e_1 \in \mathbf{I}$, if any such element exists. Otherwise, let $j > 0$ denote the smallest index such that δ_i is of the form $\pi(e_1)$ for any $e_1 \in \mathsf{Var}(q)$. We claim that there is a connection between n and e_1 in the proof graph. Clearly, this is true if $e_1 \in \mathbf{I}$ since these edges were constructed explicitly. Otherwise, Property 1 and our choice of e_1 imply that an edge from n to e_1 was constructed by the algorithm. Starting by δ_{j+1}, find all elements δ_i of the form $\pi(e)$, $e \in \mathsf{Var}(q)$, and label them consecutively as $e_2, \dots, e_l$. Note that this sequence can be empty, in which case we define $l := 1$. Obviously, $e_l = m$. We claim that $n = e_0 \dots e_l = m$ is the shortest path from n to m within the proof graph. We already showed the connection between $n = e_0$ and e_1. The connections between e_i and e_{i+1} are also obvious, since each e_1 generates e_{i+1} by definition. Since the latter path is also the only path from e_1 to e_l, the overall path is clearly the shortest connection.

The algorithm now splits $\mathcal{A}(R)$ along the path $n = e_0 \dots e_l = m$. For each e_i, there is an index $j(i)$ such that $\delta_{j(i)} = \pi(e_i)$. Hence, for each pair (e_i, e_{i+1}), there is a corresponding sequence of roles $R_{j(i)+1} \dots R_{j(i+1)}$ which we denote by r_i ($i = 0, \dots, l-1$), and the concatenation of those sequences yields the original $R_0 \dots R_{k-1}$. By Proposition 1 and Property 3, the automaton $\mathcal{A}(R)$ accepts the word $R_0 \dots R_{k-1}$. To split the automaton, we consider one accepting run and define q_i to be the state of the automaton after reading the partial sequence r_i, for each $i = 0, \dots, l-1$. The states q_i are now used to construct the split automata $\mathcal{A}_i$, and it is easy to see that those automata accept the sequences r_i.

Now assume that all required split automata have been constructed in this way. Consider any pair of query elements $e, e' \in \mathbf{I} \cup \mathsf{Var}(q)$ for which a split automaton $\mathcal{A}(F, e, e')$ was constructed using a partial sequence of roles r. We claim that the edge automaton $E(e, e')$ accepts r. Indeed, this follows from Property 2 and Theorem 2. This shows non-emptiness of intersections between any single split automaton and the corresponding edge automaton in the proof graph, and thus suffices for the case where $e' \in \mathbf{I}$.

Finally, consider the case that $e' \in \mathsf{Var}(q)$, and assume that two split automata $\mathcal{A}(F, e, e')$ and $\mathcal{A}(F', e, e')$ have been constructed for the given pair, based on two partial role sequences r and r'. We claim that $r = r'$. Indeed, this is obvious from the fact that r and r' both correspond to the unique generating sequence of roles for the elements e and e', which is part of the sequence constructed for Property 1. This shows that r is accepted both by $\mathcal{A}(F, e, e')$ and by $\mathcal{A}(F', e, e')$. We conclude that the intersection of all split automata and the edge automaton $E(e, e')$ is again non-empty.

The algorithm thus has completed all checks successfully and accepts the query. □

7 Complexity of Query Answering for $\mathcal{EL}^{++}$

Finally, we harvest a number of complexity results from the algorithm of Section 5.

Lemma 2. *Given a regular $\mathcal{EL}^{++}$ knowledge base KB and a conjunctive query q, the entailment problem KB $\models$ q is hard for* NP *w.r.t. the size of q, hard for* P *w.r.t. the size of the ABox of KB, and hard for* PSpace *w.r.t. to the combined problem size, even when restricting to simple RBoxes.*

Proof. It is well-known that the evaluation of a single function-free Horn-clause is NP-complete, even for a fixed set of ground facts [13]. This can easily be reduced to conjunctive query answering over some ABox.

Likewise, mere instance retrieval is known to be P-complete already, even with respect to an empty RBox and a fixed TBox that uses only a subset of the description logic $\mathcal{EL}$ [14].

Hardness of the combined problem is shown by reducing the problem of deciding non-emptiness of the intersection of languages accepted by a set $\mathcal{A}_1, \dots, \mathcal{A}_n$ of deterministic finite automata (DFA) to query entailment. This intersection problem is indeed known to be hard for PSpace w.r.t. the size and number of intersected automata [15]. Obviously, asking for the existence of a *non-empty* word accepted by all those automata is of the same complexity since checking for acceptance of the empty word can be done in *P*.

Assume w.l.o.g. that the intersected automata use a common alphabet Σ represented by role names R_σ for each $\sigma \in \Sigma$, and consider some class C and individual a. As in the proof of Theorem 1, we force models to represent all possible words over Σ by adding an axiom $\{a\} \sqsubseteq C$, and axioms $C \sqsubseteq \exists R_\sigma.C$ for every $\sigma \in \Sigma$.

Now we employ a construction very similar to the one used in the proof of Kleene's Theorem (equivalence of regular expressions and finite automata [16]) displayed e.g. in [17]: considering a specific DFA $\mathcal{A}_l = (\{q_1, \ldots, q_m\}, \Sigma, \delta, i, F)$, introduce roles R^k_{gh}, S^k_{gh}, and R^{loop}_k for $0 \leq k \leq m$ and $k, g, h \in 1, \ldots, m$ and define the following role inclusion axioms:

- $R_\sigma \sqsubseteq R^0_{gh}$ whenever σ causes a transition from q_g to q_h
- $R^{k-1}_{kk} \sqsubseteq R^{\text{loop}}_k$
- $R^{\text{loop}}_k \circ R^{\text{loop}}_k \sqsubseteq R^{\text{loop}}_k$
- $R^{k-1}_{gh} \sqsubseteq R^k_{gh}$
- $R^{k-1}_{gk} \circ R^{k-1}_{kh} \sqsubseteq R^k_{gh}$
- $R^{k-1}_{gk} \circ R^{\text{loop}}_k \sqsubseteq S^{k-1}_{gk}$
- $S^{k-1}_{gk} \circ R^{k-1}_{kh} \sqsubseteq R^k_{gh}$
- $R^m_{ih} \sqsubseteq R^{\text{accept}}_l$ whenever $h \in F$

W.l.o.g., we assume the sets of role names introduced for the different automata to be disjoint. Syntactically, the RBox defined this way is both regular (according to Definition 1) *and* polynomial (namely in $O(n^3)$) in the cumulated size of the automata.

Semantically, the RBox ensures the following: Assume a non-empty word $\sigma_1 \ldots \sigma_{j-1}$ causes a transition from q_g to q_h in the automaton $\mathcal{A}_l$. Then, in any model $\mathcal{I}$ with elements $e_1, \ldots, e_j$ such that $(e_o, e_{o+1}) \in R^{\mathcal{I}}_{\sigma_o}$ we also have $(e_1, e_{j-1}) \in R^{m\,\mathcal{I}}_{gh}$. This can be shown in analogy to the proof of Kleene by induction on k.

Likewise, every two elements connected by a role chain $R_{\sigma_1} \ldots R_{\sigma_j}$ for a non-empty word $\sigma_1 \ldots \sigma_j$ accepted by $\mathcal{A}_l$ are forced to additionally be directly connected by the role R^{accept}_l.

Moreover, for all words that are not accepted by $\mathcal{A}_l$, there clearly is a model that violates this property for the corresponding start and end elements (as can be easily shown by the construction of a tree-shaped minimal free model).

As mentioned above, the choice of the TBox enforces for every model of the KB and any word on Σ that there is a corresponding role sequence starting from a.

Hence the intersection problem (while excluding the empty word) for $\mathcal{A}_1, \ldots, \mathcal{A}_n$ can be reduced to the conjunctive query $\{R^{\text{accept}}_1(a, x), \ldots, R^{\text{accept}}_n(a, x)\}$. □

We remark that the above results are quite generic, and can be established for many other DLs. Especially, NP-hardness w.r.t. knowledge base size can be shown for any logic that admits an ABox, whereas PSpace hardness of the combined problem follows whenever the DL additionally admits role composition and existential role restrictions.

Lemma 3. *Given a regular $\mathcal{EL}^{++}$ knowledge base KB and a conjunctive query q, the entailment problem KB $\models$ q can be decided in* P *w.r.t. the size of the knowledge base, in* NP *w.r.t. the size of the query, and in* PSpace *w.r.t. the combined problem size, given that RBoxes are simple whenever KB is not fixed.*

Proof. First consider the step of query factorisation of the algorithm in Section 5. It clearly can be performed nondeterministically in polynomial time. If the query is fixed, the number of choices is polynomially bounded, and so the whole step is executable in polynomial time.

Similar observations hold for the proof graph initialisation. Concept names and automata for edges clearly can be assigned in polynomial time by a nondeterministic algorithm (and thus in polynomial space). If the query is of fixed size, the nondeterministic choices are again polynomial in the size of *KB*: the assignment of labels L admits at most $|\mathbf{C}|^{|\mathsf{Var}(q)|}$ different choices, and for each such choice, there are at most n^2 many possible proof graphs, where n is the number of nodes in the graph. Since n and $|\mathsf{Var}(q)|$ are considered fixed, this yields a polynomial bound.

Further nondeterminism occurs in the splitting of role automata. However, if the query is fixed, each of the polynomially many proof graphs clearly dictates a number of splits that is bounded by the size of the query m. Since splitting an automaton into k parts corresponds to selecting k (not necessarily distinct) states from the respective role automaton, there are $|Q_{\mathcal{A}}|^k$ different ways of splitting $\mathcal{A}$. Since k is again bounded by the size of the query m, we obtain an upper bound $|Q|^{m^m}$ that is still polynomial in the size of *KB* (which, by our assumptions on simplicity of the RBox, determines the maximum number of states $|Q|$ of some role automaton). If the query is not fixed, splitting again can be done nondeterministically in polynomial time.

Now for the final check of role entailment, the algorithm essentially has to check the emptiness of intersection languages of various automata. Given NFA $\mathcal{A}_1, \ldots, \mathcal{A}_l$, this check can be done in two ways, each being worst-case optimal for different side conditions of the algorithm:

(1) Initialise state variables $q_1, \ldots, q_l$ as being the initial states of the involved NFA. Then nondeterministically select one input symbol and one transition for this symbol in each of the considered NFA, and update the states q_j accordingly. The algorithm is successful if at some stage each q_j is a final state of the automaton $\mathcal{A}_j$. The algorithm runs in NPSpace w.r.t. the accumulated size of the input automata.

(2) Iteratively compute the intersection NFA for $\mathcal{A}_j = (Q_j, \Sigma, \delta_j, i_j, F_j)$ and $\mathcal{A}_{j+1} = (Q_{j+1}, \Sigma, \delta_{j+1}, i_{j+1}, F_{j+1})$. This intersection is the NFA $(Q_j \times Q_{j+1}, \Sigma, \delta, (i_j, i_{j+1}), F_j \times F_{j+1})$, with $\delta((a_1, b_1), (a_2, b_2)) = \delta(a_1, a_2) \cap \delta(b_1, b_2)$. The algorithm is successful if the intersection is non-empty. This construction is polynomial if the number of the input NFA is known to be bounded.

Method (1) establishes a general (nondeterministic) polynomial space procedure, which by Savitch's Theorem is also in PSpace. Method (2) can be used to establish tighter bounds in special cases: each intersection might cause a quadratic increase of the size of the automaton, but the number of required intersections is bounded if *KB* or q are fixed. Indeed, if the query is fixed, the number of required intersections is bounded by the overall number of role statements in the query. If the knowledge base is fixed, the possible number of interesting intersections is bounded by the number of split automata that can be produced from role automata constructed from the RBox, which is clearly bounded by a fixed value. In both cases, checking intersections can be done deterministically in polynomial time. □

We summarise the contents of Lemmas 2 and 3 in Table 3.

Table 3. Complexities of conjunctive query answering in regular $\mathcal{EL}^{++}$ knowledge bases. Whenever the RBox is variable, we assume that it is simple.

	Variable parts: Query	RBox	TBox	ABox	Complexity
Combined complexity	×	×	×	×	PSPACE-complete
Query complexity	×				NP-complete
Schema complexity		×	×	×	P-complete
Data complexity				×	P-complete

8 Conclusion

We have proposed a novel algorithm for answering conjunctive queries in $\mathcal{EL}^{++}$ knowledge bases, which is worst-case optimal under various assumptions. To the best of our knowledge, this also constitutes the first inference procedure for conjunctive queries in a DL that supports complex role inclusions (including composition of roles) in the sense of OWL 1.1. Showing undecidability of conjunctive queries for unrestricted $\mathcal{EL}^{++}$, we illustrated that the combination of role atoms in queries and complex role inclusion axioms can indeed make reasoning significantly more difficult.

A compact automata-based representation of role chains *and* (parts of) models allowed us to establish polynomial bounds for inferencing in various cases, thus identifying querying scenarios that are still tractable for $\mathcal{EL}^{++}$. Conjunctive queries inherently introduce some nondeterministism, but automata can conveniently represent sets of possible solutions instead of considering each of them separately. We therefore believe that the presented algorithm can be a basis for actual implementations that introduce additional heuristics to ameliorate nondeterminism.

Acknowledgements. This work was substantially improved through the comments of Pascal Hitzler and various anonymous reviewers. This research has been supported by the EU in the IST project NeOn (IST-2006-027595).

References

1. Krötzsch, M., Rudolph, S.: Conjunctive queries for $\mathcal{EL}$ with role composition. In: Proc. 2007 Description Logic Workshop (DL 2007), CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/` (2007)
2. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In Hopcroft, J.E., Friedman, E.P., Harrison, M.A., eds.: Proc. 9th annual ACM Symposium on Theory of Computing (STOC'77), ACM Press (1977) 77–90
3. Horrocks, I., Sattler, U., Tessaris, S., Tobies, S.: How to decide query containment under constraints using a description logic. In Parigot, M., Voronkov, A., eds.: Proc. 7th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2000). Volume 1955 of LNAI., Springer (2000) 326–343
4. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: Proc. 11th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2004). (2005) 21–35

5. Ortiz, M.M., Calvanese, D., Eiter, T.: Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. In: Proc. 2006 Description Logic Workshop (DL 2006), CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/` (2006)
6. Ortiz, M.M., Calvanese, D., Eiter, T.: Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. 21st Nat. Conf. on Artificial Intelligence (AAAI'06). (2006)
7. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In: Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-07), Hyderabad, India (2007) Available at `http://www.ijcai.org/papers07/contents.php`.
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006), AAAI Press (2006) 57–67
9. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05), Edinburgh, UK, Morgan-Kaufmann Publishers (2005)
10. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In Gottlob, G., Walsh, T., eds.: Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03), Acapulco, Mexico, Morgan-Kaufmann Publishers (2003) 343–348
11. Wessel, M.: Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In: Proc. 2001 Description Logic Workshop (DL 2001). (2001)
12. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. second edn. Cambridge University Press (2002)
13. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys **33** (2001) 374–425
14. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). (2006) 260–270
15. Kozen, D.: Lower bounds for natural proof systems. In: Proc. 18th Symp. on the Foundations of Computer Science. (1977) 254–266
16. Kleene, S. In: Representation of Events in Nerve Nets and Finite Automata. Princeton University Press, Princeton, N.J. (1956) 3–42
17. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts (1979)