I N F S Y S

R E S E A R C H

R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

ANSWERING REGULAR PATH QUERIES IN EXPRESSIVE DESCRIPTION LOGICS: AN AUTOMATA-THEORETIC APPROACH

Magdalena Ortiz Diego Calvanese Thomas Eiter

INFSYS RESEARCH REPORT 1843-08-05

JUNE 2008 (PRELIMINARY VERSION)

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstrassße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TU WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-08-05, JUNE 2008 (PRELIMINARY VERSION)

ANSWERING REGULAR PATH QUERIES IN EXPRESSIVE DESCRIPTION LOGICS: AN AUTOMATA-THEORETIC APPROACH

Diego Calvanese,[1] Thomas Eiter,[2] and Magdalena Ortiz[3]

Abstract. Expressive Description Logics (DLs) have been advocated as formalisms for modeling the domain of interest in various application areas. An important requirement there is the ability to answer complex queries beyond instance retrieval, taking into account constraints expressed in a knowledge base. We consider this task for positive existential path queries (which generalize conjunctive queries and unions thereof), whose atoms are regular expressions over the roles (and concepts) of a knowledge base in the expressive DL $\mathcal{ALCQI}b_{reg}$. Using techniques based on two-way tree-automata, we first provide an elegant characterization of TBox and ABox reasoning, which gives us also a tight EXPTIME bound. We then prove decidability (more precisely, a 2EXPTIME upper bound) of query answering, thus significantly pushing the decidability frontier, both with respect to the query language and the considered DL. We also show that query answering is EXPSPACE-hard already in rather restricted settings.

[1]Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, I-39010 Bolzano, Italy. E-mail: calvanese@inf.unibz.it.

[2]Institute of Information Systems, Knowledge-Based Systems Group, Vienna University of Technology, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: eiter@kr.tuwien.ac.at.

[3]Institute of Information Systems, Knowledge-Based Systems Group, Vienna University of Technology, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: ortiz@kr.tuwien.ac.at.

This report is a preliminary version. Some results in this paper appear in the Proceedings of the 22nd AAAI Conference on Artificial Intelligence *(AAAI '07)* [7].

Contents

1 Introduction

Description Logics (DLs) [1] are a well-established branch of logics for knowledge representation and reasoning, and the premier logic-based formalism for modeling concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). They have gained increasing attention in different areas including the Semantic Web, data and information integration, peer-to-peer data management, and ontology-based data access. In particular, some of the standard Web ontologies from the OWL family are based on DLs [13].

While in DLs, traditionally reasoning tasks had been studied which deal with taxonomic issues like classification and instance checking, the widening range of applications has led in the recent years to extensive studies of answering queries over DL knowledge bases which require, beyond simple instance retrieval, to join pieces of information in finding the answer. Specifically, *conjunctive queries* have been studied in several papers, cf. [2, 11, 12, 16, 17, 21, 22]. As shown therein, answering (classes of) conjunctive queries is decidable for several DLs, including also expressive ones. Recently, the authors of [11] proved this for all conjunctive queries over $\mathcal{SHIQ}$ knowledge bases, while Hustadt *et al.* [16, 17] showed this earlier for conjunctive queries without transitive roles and Ortiz *et al.* [22] for unions of such queries.[1]

At present, (unions of) conjunctive queries over $\mathcal{SHIQ}$ knowledge bases is among the most expressive decidable settings. In this paper, we push the frontier and establish decidability of query answering for the yet more expressive class of *positive (existential) two-way regular path queries* (in short, P2RPQs) over the expressive DL $\mathcal{ALCQIb}_{reg}$, which is close to $\mathcal{SHIQ}$. P2RPQs are queries inductively built using conjunction and disjunction, from atoms that are regular expressions over direct and inverse roles (and allow for testing of concepts). They not only subsume conjunctive queries and unions of conjunctive queries, but also unions of conjunctive regular path queries [6].

More specifically, we make the following contributions.

• Different from previous works, which rely on resolution-based transformations to disjunctive datalog or on tableaux-based algorithms, we use automata techniques for query answering in expressive DLs. While the application of automata techniques in DLs is not novel, cf.[4, 25], previous work was concerned with deciding satisfiability of a knowledge base consisting of a TBox only. Here we address the much more involved task of query answering over a knowledge base, which has data in an ABox; incorporating the query is non-obvious.

• The technique we apply is simpler and clearer than the existing ones based on tableaux and resolution. Indeed, it is computational in nature, and directly works on the models of the knowledge base. In this way, we are also able to obtain more general results, which seems more difficult using the other approaches.

• As a first result, we present an automata-based algorithm for checking the satisfiability of a knowledge base (consisting of both TBox and ABox) in EXPTIME. This is worst-case optimal.

• Our main result then shows that answering positive existential queries over $\mathcal{ALCQIb}_{reg}$ knowledge bases is feasible in 2EXPTIME. By a reduction of $\mathcal{RIQ}$ and to $\mathcal{ALCQIb}_{reg}$, a similar result follows for $\mathcal{SHIQ}$ and $\mathcal{RIQ}$, and can easily be extended to $\mathcal{SRIQ}$. This compares well to the N3EXPTIME bound for union of conjunctive queries by Ortiz *et al.* [22], and the 2EXPTIME bounds for (classes of) conjunctive queries that emerge from [11, 16]. On the other hand, we establish an EXPSPACE lower bound for positive existential queries.

Our results indicate that automata-techniques have high potential for advancing the decidability frontier of query answering over expressive DLs, and are a useful tool for analyzing its complexity.

[1] Note that the technique in [3] for unions of conjunctive regular path queries is actually incomplete.

2 Preliminaries

2.1 Query Answering in $\mathcal{ALCQIb}_{reg}$

We consider the DL $\mathcal{ALCQIb}_{reg}$, whose concepts and roles obey the following syntax:

$$\begin{array}{lcl} C, C' & \longrightarrow & A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \forall R.C \mid \exists R.C \mid \geqslant n\, Q.C \mid \leqslant n\, Q.C \\ Q, Q' & \longrightarrow & P \mid P^- \mid Q \cap Q' \mid Q \cup Q' \mid Q \setminus Q' \\ R, R' & \longrightarrow & Q \mid R \cup R' \mid R \circ R' \mid R^* \mid id(C) \end{array}$$

where A denotes an *atomic concept*, P an *atomic role*, C an arbitrary *concept*, and R an arbitrary *role*. We use Q to denote *basic roles*, which are those roles which may occur in number restrictions. W.l.o.g., we assume that "$\setminus$" is applied only to atomic roles and their inverses.

An $\mathcal{ALCQIb}_{reg}$*knowledge base* (KB) is a pair $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ where $\mathcal{A}$ (the *ABox*) is a set of *assertions* of the form $A(a)$, $P(a, b)$, and $a \neq b$, with A an atomic concept, P an atomic role, and a, b *individuals*; and $\mathcal{T}$ (the *TBox*) is a set of *concept inclusion axioms* $C \sqsubseteq C'$ for arbitrary concepts C and C'. W.l.o.g. we assume that all concepts occurring in $\mathcal{A}$ also occur in $\mathcal{T}$. We denote by $\mathcal{C}_\mathcal{K}$ the set of atomic concepts occurring in $\mathcal{K}$, by $\mathcal{R}_\mathcal{K}$ the set of atomic roles occurring in $\mathcal{K}$ and their inverses, and by $\mathcal{J}_\mathcal{K}$ the individuals in $\mathcal{K}$. If clear from the context, we may omit the index $\mathcal{K}$.

The semantics is the standard one. *Knowledge base satisfiability* is the task of determining whether there exists an interpretation that satisfies all assertions in $\mathcal{A}$ and all the axioms in $\mathcal{T}$. We do not implement the *unique name assumption* (UNA), i.e., two individuals a and b may be interpreted as the same domain element.[2]

Definition 2.1 [P2RPQs] A *positive 2-way regular path query* (P2RPQ) over a KB $\mathcal{K}$ is a formula $\exists \vec{x}.\varphi(\vec{x})$, where $\varphi(\vec{x})$ is built using $\wedge$ and $\vee$ from atoms of the form $C(z)$ and $R(z, z')$, with z, z' variables from $\vec{x}$ or individuals, C an arbitrary concept, R an arbitrary role, and where all atomic concepts and roles in φ occur in $\mathcal{K}$.

Note that P2RPQs naturally generalize conjunctive regular path queries [6] and unions thereof.

Example 2.2 Consider the query q over a genealogy KB $\mathcal{K}$:

$$\exists x, y, z.\ parent^* \cdot parent^{-*}(x, y) \wedge parent^-(x, z) \wedge parent^-(y, z) \wedge\ male(x) \wedge \neg male(y) \wedge (\neg deity(x) \vee \neg deity(y))$$

Informally, q is true if there are relatives x and y that have a common child, z, and if not both of them are deities. ∎

Let q be a P2RPQ, and let $\mathsf{Vl}(q)$ denote the set of variables and individuals in q. Given an interpretation $\mathcal{I}$, let $\pi : \mathsf{Vl}(q) \to \Delta^\mathcal{I}$ be a total function such that $\pi(a) = a^\mathcal{I}$ for each individual a. We write $\mathcal{I}, \pi \models C(z)$ if $\pi(z) \in C^\mathcal{I}$, and $\mathcal{I}, \pi \models R(z, z')$ if $(\pi(z), \pi(z')) \in R^\mathcal{I}$. Let γ be the Boolean expression obtained from φ by replacing each atom α in φ with **true**, if $\mathcal{I}, \pi \models \alpha$, and with **false** otherwise. We say that π is a *match for* $\mathcal{I}$ *and* q, denoted $\mathcal{I}, \pi \models q$, if γ evaluates to **true**. We say that $\mathcal{I}$ *satisfies* q, written $\mathcal{I} \models q$, if there is a match π for $\mathcal{I}$ and q. A KB $\mathcal{K}$ *entails* q, denoted $\mathcal{K} \models q$, if $\mathcal{I} \models q$ for each model $\mathcal{I}$ of $\mathcal{K}$.

[2] The UNA can be easily simulated using assertions of the form $a \neq b$.

Query entailment consists in verifying, given a KB $\mathcal{K}$ and a P2RPQ q, whether $\mathcal{K} \models q$. Note that, w.l.o.g., we consider here query entailment for Boolean queries, i.e., queries without free variables, since query answering for non-Boolean queries is polynomially reducible to query entailment.[3]

2.2 Automata on Infinite Trees.

Infinite trees are represented as prefix-closed sets of words over $\mathbb{N}$ (the set of positive integers). Formally, an *infinite tree* is a set of words $T \subseteq \mathbb{N}^*$, such that if $x{\cdot}c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The elements of T are called *nodes*, the empty word ε is its *root*. For every $x \in T$, the nodes $x{\cdot}c$, with $c \in \mathbb{N}$, are the *successors* of x. If x is a successor of y, then y is a *predecessor* of x and x and y are *neighbors*. By convention, $x{\cdot}0 = x$, and $(x{\cdot}i){\cdot}{-1} = x$. The *branching degree* $d(x)$ of a node x is the number of its successors. If $d(x) \leq k$ for each node x of T, then T has *branching degree* k. An *infinite path* P of T is a prefix-closed set $P \subseteq T$ where for every $i \geq 0$ there exists a unique node $x \in P$ with $|x| = i$. A *labeled tree* over an alphabet Σ is a pair (T, V), where T is a tree and $V : T \to \Sigma$ maps each node of T to an element of Σ.

Alternating automata on infinite trees are a generalization of nondeterministic automata on infinite trees, introduced in [19]. They allow for an elegant reduction of decision problems for temporal and program logics [9].

Let $\mathcal{B}(I)$ be the set of positive Boolean formulas built inductively from **true**, **false**, and atoms from a set I applying $\wedge$ and $\vee$. A set $J \subseteq I$ *satisfies* a formula $\varphi \in \mathcal{B}(I)$, if assigning **true** to the atoms in J and **false** to those in $I \setminus J$ makes φ true. A *two-way alternating tree automaton* (2ATA) running over infinite trees with branching degree k, is a tuple $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$, where Σ is the input alphabet; Q is a finite set of states; $\delta : Q \times \Sigma \to \mathcal{B}([k] \times Q)$, where $[k] = \{-1, 0, 1, \ldots, k\}$, is the transition function; $q_0 \in Q$ is the initial state; and F specifies the acceptance condition.

The transition function δ maps a state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a positive Boolean formula φ. over atoms $[k] \times Q$. Intuitively, if $\delta(q, \sigma) = \varphi$, then each atom (c, q') in φ corresponds to a new copy of the automaton going in the direction given by c and starting in state q'. E.g., let $k = 2$ and $\delta(q_1, \sigma) = (1, q_2) \wedge (1, q_3) \vee (-1, q_1) \wedge (0, q_3)$. If $\mathbf{A}$ is in the state q_1 and reads the node x labeled with σ, it proceeds by sending off either two copies, in the states q_2 and q_3 respectively, to the first successor of x (i.e., $x{\cdot}1$), or one copy in the state q_1 to the predecessor of x (i.e., $x{\cdot}{-1}$) and one copy in the state q_3 to x itself (i.e., $x{\cdot}0$).

2ATAs generalize standard nondeterministic automata on infinite trees in two ways. First, in (standard) one-way automata input trees are always navigated in a strictly top-down manner, moving always to the successors of the current node. In contrast, in two-way automata transitions can be defined that move up on the input tree to the predecessor of the current node, or that stay at the current position. Comparing with the above definition, the transition function of a one way automaton is of the form $\delta : Q \times \Sigma \to \mathcal{B}(\{1, \ldots, k\} \times Q)$, i.e., the directions -1 and 0 are not allowed. The second difference is *alternation*, which generalizes non-determinism. While in alternating automata the transition function can have any positive Boolean formula in $\mathcal{B}([k] \times Q)$ on the right hand side, the transition function of a nondeterministic automaton only allows for disjunctions of the form $d_0, \cdots, d_n$, where each d_i is in turn a conjunction of pairs (i, q) containing exactly one such pair for each $i \in [k]$. We denote *one-way non-deterministic automata* by 1NTA.

[3] Here we refer to the decision problem associated to query answering, i.e., to decide whether a given tuple is in the answer of the query.

Informally, a run of a 2ATA $\mathbf{A}$ over a labeled tree (T, V) is a labeled tree (T_r, r) in which each node n is labeled by an element $r(n) = (x, q) \in T \times Q$ and describes a copy of $\mathbf{A}$ that is in the state q and reads the node x of T; the labels of adjacent nodes must satisfy the transition function of $\mathbf{A}$. Formally, a *run* (T_r, r) is a $T \times Q$-labeled tree satisfying:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$.
2. Let $y \in T_r$, with $r(y) = (x, q)$ and $\delta(q, V(x)) = \varphi$. Then there is a (possibly empty) set $S = \{(c_1, q_1), \ldots, (c_n, q_n)\} \subseteq [k] \times Q$ such that:
   - S satisfies φ and
   - for all $1 \leq i \leq n$, we have that $y{\cdot}i \in T_r$, $x{\cdot}c_i$ is defined, and $r(y{\cdot}i) = (x{\cdot}c_i, q_i)$.

   In this case, we say that $r(y) = (x, q)$ *evaluates to true in the run* (T_r, r).

For a (possibly infinite) path π in a run of the automaton, we denote by $\mathsf{Inf}(\pi)$ the states of Q that occur infinitely often in π (as second components of node labels). Different kinds of acceptance conditions can be considered for π:

1. A *Büchi* acceptance condition is given by a set of states $F \subseteq Q$. The path π is accepting if $\mathsf{Inf}(\pi) \cap F \neq \emptyset$.
2. A *parity acceptance condition (of length n)* is defined by a finite sequence of sets of states $F = (G_1, \ldots, G_n)$ with $G_1 \subseteq G_2 \subseteq \cdots \subseteq G_n = Q$. The path π is accepting if there is an *even* i such that $\mathsf{Inf}(\pi) \cap G_i \neq \emptyset$ and $\mathsf{Inf}(\pi) \cap G_{i-1} = \emptyset$.
3. A *coparity* acceptance condition (of length n) is defined as a parity one, being the only difference that π is accepting if there is an *odd* i such that $\mathsf{Inf}(\pi) \cap G_i \neq \emptyset$ and $\mathsf{Inf}(\pi) \cap G_{i-1} = \emptyset$.

A run (T_r, r) is *accepting*, if all its paths are accepting. An automaton with a parity acceptance condition can be easily converted into one with coparity acceptance that accepts exactly the same input trees, and vice versa [20]. Furthermore, it is easy to see that a *Büchi* acceptance condition can be easily converted into a parity or coparity one. The converse is not true, however: parity/coparity automata are strictly more expressive than Büchi automata [24].

The *nonemptiness problem* for 2ATAs is deciding whether the set $\mathcal{L}(\mathbf{A})$ of trees accepted by a given 2ATA $\mathbf{A}$ is nonempty. The following result is well-known.

Theorem 2.3 [27] *For any 2ATA $\mathbf{A}$ (with parity acceptance condition) with n states and an input alphabet with m elements, nonemptiness of $\mathbf{A}$ is decidable in time single exponential in n and polynomial in m. Also, there is a one-way nondeterministic Parity tree automaton (1NTA) $\mathbf{A}_1$ with $O(2^n)$ states and parity condition of size $O(n)$ such that $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A}_1)$.*

3 Deciding KB satisfiability via automata

For many DLs including $\mathcal{ALCQI}b_{reg}$, the standard reasoning tasks are naturally solvable by tree-automata, thanks to the *tree model property* of such DLs: every satisfiable TBox $\mathcal{T}$ (or similarly, every satisfiable concept C) has a tree-shaped model. In the presence of an ABox $\mathcal{A}$ this property fails, since the assertions in $\mathcal{A}$ may arbitrarily connect the individuals. While a satisfiable $\mathcal{ALCQI}b_{reg}$ KB $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ may lack a

tree-shaped model, it always has a forest-shaped *canonical model*, in which each individual is the root of a tree-shaped model of $\mathcal{T}$. This property is usually sufficient to adapt algorithms for concept satisfiability to decide KB satisfiability. In particular, automata-based algorithms have been adapted using the *precompletion* technique [25], in which after a reasoning step on the ABox, automata are used to verify the existence of a tree-shaped model rooted at each ABox individual.

Our approach is different. We represent a forest-shaped interpretation as a tree $\mathbf{T}$, and encode $\mathcal{K}$ into an automaton $\mathbf{A}_\mathcal{K}$ that accepts $\mathbf{T}$ iff $\mathbf{T}$ corresponds to a canonical model of $\mathcal{K}$. To the best of our knowledge, this is the first algorithm that deals with ABox assertions and individuals directly in the automaton. This enables us to extend the automata-based algorithm also to query answering (see next section).

First we define the *(syntactic) closure* of an $\mathcal{ALCQIb}_{reg}$concept, which extends the standard Fischer-Ladner closure for *converse*-PDL [10]. Intuitively, it contains all the concepts and roles that may occur when a concept is decomposed during a run of an automaton on a tree representing a model of $\mathcal{K}$. Usually the closure is defined as a set of concepts. We will use a different definition, where the closure is a set of *expressions* that may contain concepts and basic roles, as well as some additional elements. As we will see, the automata algorithm treats similarly boolean constructors on concepts and roles. Individual names and auxiliary labels are handled analogously as the atomic concepts and roles. Putting all these elements into the closure will simplify the construction of the automata.

Here and in the following, we use $\gtrless$ to denote either $\geqslant$ or $\leqslant$ and we use Q^- to denote P when $Q = P^-$. For a non-atomic basic role Q, Q^- denotes the role obtained from Q by replacing each atomic or inverse of atomic role occurring in Q by its inverse. In the rest of this subsection we assume that $\sqcup$ and $\forall$ are expressed by means of $\sqcap$ and $\exists$ using $\neg$. (We remind that C and C' stand for arbitrary concepts, P and P' for atomic or inverse of atomic roles, Q and Q' for basic roles and R and R' for arbitrary roles).

The closure $Cl(D)$ of an $\mathcal{ALCQIb}_{reg}$concept D is defined as the smallest set of expressions such that $D \in Cl(D)$ and:

if	$C \in Cl(D)$	then	$\neg C \in Cl(D)$	(if C is not of the form $\neg C'$)
if	$\neg C \in Cl(D)$	then	$C \in Cl(D)$	
if	$C \sqcap C' \in Cl(D)$	then	$C,\, C' \in Cl(D)$	
if	$\exists R.C \in Cl(D)$	then	$C \in Cl(D)$	
if	$\exists (R \cup R').C \in Cl(D)$	then	$\exists R.C,\, \exists R'.C \in Cl(D)$	
if	$\exists (R \circ R').C \in Cl(D)$	then	$\exists R.\exists R'.C \in Cl(D)$	
if	$\exists R^*.C \in Cl(D)$	then	$\exists R.\exists R^*.C \in Cl(D)$	
if	$\exists id(C).C' \in Cl(D)$	then	$C,\, C' \in Cl(D)$	
if	$\exists Q.C \in Cl(D)$	then	$Q \in Cl(D)$	
if	$\gtrless n\, Q.C \in Cl(D)$	then	$Q,\, C \in Cl(D)$	
if	$Q \cap Q' \in Cl(D)$	then	$Q,\, Q' \in Cl(D)$	
if	$Q \cup Q' \in Cl(D)$	then	$Q,\, Q' \in Cl(D)$	
if	$P \setminus P' \in Cl(D)$	then	$P,\, P' \in Cl(D)$	
if	$Q \in Cl(D)$	then	$\neg Q,\, Q^-,\, \neg Q^- \in Cl(D)$	
if	$Q^- \in Cl(D)$	then	$\neg Q,\, Q,\, \neg Q^- \in Cl(D)$	

Note that $|Cl(D)|$ is linear in the length of D. Sometimes we consider concepts C in *negation normal form*, denoted $nnf(C)$, in which negations are pushed inside as much as possible, i.e., negation only occurs in the form $\neg A$ for A a concept name. Analogously, in the *negation normal form* of a basic role Q, which is denoted $nnf(Q)$, only atomic or inverse of atomic roles may occur negated. Clearly, any concept or basic role E can be transformed into its negation normal form in time linear in the size of E. We let $Cl^{nnf}(D) =$

$\{nnf(E) \mid E \in Cl(D)\}$. For a concept D and a given set S, we denote the set $Cl^{nnf}(D) \cup \{\neg s,\ s \mid s \in S\}$ by $Cl^{nnf}(D, S)$. and call it the *S-extended closure* of D.

3.1 Representing Canonical Models as Trees

Let C be an $\mathcal{ALCQIb}_{reg}$concept and let $\mathbf{n}$ be the maximal n occurring in a concept of the form $\geqslant n\, Q.C'$ in $Cl^{nnf}(C)$. Consider an arbitrary interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for C, and consider the directed graph whose nodes are the elements of $\Delta^{\mathcal{I}}$, and where any two nodes $a, b \in \Delta^{\mathcal{I}}$ are connected by an arc $a \rightarrow b$ iff $\langle a, b\rangle \in P^{\mathcal{I}}$ for some P. Then $\mathcal{I}$ is a tree interpretation if this graph forms a tree. The *branching degree* of the tree is n iff every node $a \in \Delta^{\mathcal{I}}$ has at most n outgoing arcs. It is well known that $\mathcal{ALCQIb}_{reg}$has the following *tree model property*:

Theorem 3.1 *[4] Every satisfiable $\mathcal{ALCQIb}_{reg}$concept C has a tree-model with branching degree $k_C = |Cl(C)| \times \mathbf{n}$.*

Consider an $\mathcal{ALCQIb}_{reg}$knowledge base $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}\rangle$, and let

$$C_{\mathcal{T}} = \forall(\textstyle\bigcup_{R \in \mathcal{R}_{\mathcal{K}}} R)^*.\ \bigsqcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2)$$

By internalization [23], $C_{\mathcal{T}}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that satisfies all the concept inclusion axioms in $\mathcal{T}$ [5]. This implies that, by Theorem 3.1, the existence of a tree-shaped model for the TBox $\mathcal{T}$ can be ensured. The assertions in $\mathcal{A}$ may impose arbitrary relations between the individuals, thus $\mathcal{K}$ does not necessarily have tree shaped models. However, it has a slightly weaker *canonical model* property.

Let $\mathcal{I}$ be an interpretation for $\mathcal{K}$ and let $\{o_1, \ldots, o_n\} \subseteq \Delta^{\mathcal{I}}$ be the elements interpreting the individuals in $\mathcal{A}$. We say that $\mathcal{I}$ is canonical if the associated graph is such that, when all arcs of the form $o_i \rightarrow o_j$ with $1 \leq i \leq j \leq n$ are removed, we obtain a set of trees, each of which is rooted at some o_i. The domain of any such canonical model $\mathcal{I}$ can be represented as a set of trees. Let $\mathcal{J} = \{a_1, \ldots, a_m\}$ be the set of ABox individuals interpreted in $\mathcal{I}$, and let $S = \{t_1, \ldots, t_n\}$ be the set of tree-shaped models of $C_{\mathcal{T}}$ in $\mathcal{I}$, each of which is rooted at the interpretation of some a_i. The objects interpreting the individuals in $\mathcal{J}$ are denoted $1, \ldots, n$ (note that, since any two individuals may be interpreted as the same object, we have that $n \leq m$). Each tree t_i in S is rooted at some $i \in \{1, \ldots, n\}$ and has a branching degree of $k_{C_{\mathcal{T}}}$. Thus, we can define canonical models as follows:

Definition 3.2 Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ be an $\mathcal{ALCQIb}_{reg}$ KB and let $\mathcal{J} = \{a_1, \ldots, a_m\}$ be the individuals in $\mathcal{A}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for $\mathcal{K}$ is called a *canonical interpretation* if:

(1) $\Delta^{\mathcal{I}} \subseteq \{1, \ldots, n\}\{1, \ldots, k_{C_{\mathcal{T}}}\}^*$ where $n \leq m$ and $\{\varepsilon\} \cup \Delta^{\mathcal{I}}$ is prefix closed (i.e., it is a tree).

(2) For each $a_i \in \mathcal{J}$ there is exactly one $j \in \Delta^{\mathcal{I}}$, such that $j \in \{1, \ldots, n\}$ and $a_i^{\mathcal{I}} = j$.

(3) If $(i{\cdot}w, j{\cdot}w') \in P^{\mathcal{I}}$ for some atomic role P, $i, j \in \{1, \ldots, n\}$, then either (*i*) $w = w' = \varepsilon$ or (*ii*) $i = j$ and w, w' are neighbors (in the tree $\{\varepsilon\} \cup \Delta^{\mathcal{I}}$).

The canonical model property of $\mathcal{ALCQIb}_{reg}$ is also well known:

Theorem 3.3 *Every satisfiable $\mathcal{ALCQIb}_{reg}$KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ has a canonical model $\mathcal{I}$ such that $\mathcal{I}$ is a canonical interpretation for $\mathcal{K}$, $\mathcal{I} \models \mathcal{A}$ and $a_i^{\mathcal{I}} \in C_{\mathcal{T}}^{\mathcal{I}}$ holds for all $a_i \in \mathcal{J}$.*

We want to decide the satisfiability of a KB $\mathcal{K}$. Due to Theorem 3.3, it is sufficient to decide the existence of a canonical model for $\mathcal{K}$. In order to do the latter by means of tree automata, we represent a canonical interpretation $\mathcal{I}$ for $\mathcal{K}$ as a labeled tree $\mathbf{T}_\mathcal{I}$. We define an automaton $\mathbf{A}_\mathcal{K}$ that accepts a labeled tree $\mathbf{T}_\mathcal{I}$ iff $\mathbf{T}_\mathcal{I}$ represents a canonical model of $\mathcal{K}$, and check this automaton for emptiness.

First, we define the way in which a canonical interpretation is represented as a labeled tree. The domain of a canonical interpretation is almost a tree; we only need to add a root ε to it and to represent the extensions of the interpretations of individuals, concepts and roles as labels in the tree. This can be done in a straightforward manner. Roughly, each element of $x \in \Delta^\mathcal{I}$, which is a node of the tree, is labeled with a set $V(x)$ that contains the atomic concepts A such that $x \in A^\mathcal{I}$ and the basic roles that connect the predecessor of x to x. Only the root ε and the nodes $\{1, \ldots, n\}$ (which are at the first level of the tree) are treated differently. The label of each node in $x \in \{1, \ldots, n\}$, apart from the atomic concepts to which x belongs, also contains the name of the individuals in $\mathcal{J}$ which it interprets. Furthermore, the label of x contains no basic roles. Instead, the relations between level one nodes are stored in the label of the root ε. The root ε does not represent any object in $\Delta^\mathcal{I}$ and is marked with a special symbol r, to identify it as the root, and symbols of the form Pij indicating that the pair of individuals (a_i, a_j) belongs to the extension of the basic role P. For simplicity, if $m > n$, to ensure that the root has exactly m children, $m - n$ dummy children labeled $\{d\}$ are added to the root ε.

Definition 3.4 Let $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ be an $\mathcal{ALCQI}b_{reg}$ knowledge base and let $\mathcal{J} = \{a_1, \ldots, a_m\}$ be the set of individuals occurring in $\mathcal{A}$. Let $\mathcal{I}$ be a canonical interpretation for $\mathcal{K}$ in which the individuals in $\mathcal{J}$ are interpreted as $\{1, \ldots, n\}$. The *tree representation of* $\mathcal{I}$ is the labeled tree $\mathbf{T}_\mathcal{I} = (T, V)$ defined as follows:

- $T = \{\varepsilon\} \cup \Delta^\mathcal{I} \cup \{n+1, \ldots, m\}$.
- $V(\varepsilon) = \{r\} \cup \{Pij \mid a_i, a_j \in \mathcal{J} \text{ and } \langle a_i^\mathcal{I}, a_j^\mathcal{I} \rangle \in P^\mathcal{I}\}$,
- for each $1 \leq i \leq n$, $V(i) = \{a_j \in \mathcal{J} \mid a_j^\mathcal{I} = i\} \cup \{A \in \mathcal{C}_\mathcal{K} \mid a_j^\mathcal{I} \in A^\mathcal{I} \text{ and } a_j^\mathcal{I} = i\}$,
- for each $n+1 \leq i \leq m$, $V(i) = \{d\}$,
- and for all other nodes $i{\cdot}x$ of T, $V(i{\cdot}x) = \{A \in \mathcal{C}_\mathcal{K} \mid i{\cdot}x \in A^\mathcal{I}\} \cup \{P \in \mathcal{R}_\mathcal{K} \mid (i{\cdot}w, i{\cdot}x) \in P^\mathcal{I} \text{ where } x = w{\cdot}j \text{ for some } j, 1 \leq j \leq k_{C_\mathcal{T}}\}$

Note that the branching degree of $\mathbf{T}_\mathcal{I}$ is bounded by $|\mathcal{J}|$ at the root and by $k_{C_\mathcal{T}}$ at all other levels, so the tree has branching degree $max(k_{C_\mathcal{T}}, |\mathcal{J}|)$.

3.2 Constructing the Automaton

We now construct from $\mathcal{K}$ a 2ATA $\mathbf{A}_\mathcal{K}$ that accepts a given tree $\mathbf{T}$ iff it is the tree representation $\mathbf{T}_\mathcal{I}$ of some canonical model $\mathcal{I}$ of $\mathcal{K}$. A similar construction for deciding concept satisfiability in $\mathcal{ALCQI}b_{reg}$ was presented by Calvanese *et al.* [4]. We adapt and expand that construction to handle the ABox also.

The automaton $\mathbf{A}_\mathcal{K} = (\Sigma_\mathcal{K}, S_\mathcal{K}, \delta, s_0, F_\mathcal{K})$ is defined as follows.

- The alphabet is $\Sigma_\mathcal{K} = 2^{\mathcal{C}_\mathcal{K} \cup \mathcal{R}_\mathcal{K} \cup \mathcal{J} \cup \{r\} \cup \{d\} \cup PI}$, where $PI = \{Pij \mid a_i, a_j \in \mathcal{J} \text{ and } P \in \mathcal{R}_\mathcal{K}\}$;

  According to Definition 3.4, the node labels are sets of atomic concepts, basic roles, and individuals in $\mathcal{J}$, plus the auxiliary symbols r and d and the symbols of the form Pij in PI, which may occur in the root's label.

- The set of states is $S_{\mathcal{K}} = \{s_0, s_1\} \cup Cl_{ext} \cup S_{num} \cup S_{\mathcal{A}\_role} \cup S_{\mathcal{A}\_quant} \cup S_{\mathcal{A}\_num}$ where s_0 is the initial state, s_1 is an additional state, and $Cl_{ext} = Cl^{nnf}(C_{\mathcal{T}}, \mathcal{J} \cup \{r, d\})$ is the $(\mathcal{J} \cup \{r, d\})$-extended closure of $C_{\mathcal{T}}$.

  Each $E \in Cl_{ext}$ is a state in $S_{\mathcal{K}}$, and these are the 'basic' states of $\mathbf{A}_{\mathcal{K}}$. Intuitively, when the automaton $\mathbf{A}_{\mathcal{K}}$ moves to a state $E \in Cl_{ext}$ and visits a node x of the tree, it must check that E holds in x. To this aim, the expressions in Cl_{ext} are inductively decomposed and the tree is navigated accordingly. For some particular concepts (e.g. number restrictions and existential and universal involving ABox individuals), we need additional special states, namely the ones in the sets S_{num}, $S_{\mathcal{A}\_role}$, $S_{\mathcal{A}\_quant}$ and $S_{\mathcal{A}\_num}$. These are defined and briefly explained below.

$$\begin{array}{rl} S_{num} = & \{\langle \geqslant n\, Q.C, i, j\rangle \mid \geqslant n\, Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ 0 \leq i \leq k_{C_{\mathcal{T}}}{+}1,\ 0 \leq j \leq n\} \cup \\ & \{\langle \leqslant n\, Q.C, i, j\rangle \mid \leqslant n\, Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ 0 \leq i \leq k_{C_{\mathcal{T}}}{+}1,\ 0 \leq j \leq n{+}1\} \end{array}$$

  The states in the set S_{num} are used for checking whether the number restrictions are satisfied. Roghly, in one such state, i stores how many successors of the current node have been navigated, and j how many of them are reached through Q and labeled with C.

$$S_{\mathcal{A}\_role} = \{Qij \mid a_i, a_j \in \mathcal{J} \text{ and } Q \text{ a basic role in } Cl^{nnf}(C_{\mathcal{T}})\}$$

  The automaton moves to a state of the form Qij when it needs to to verify whether ABox individuals a_i and a_j are related by a role Q.

$$\begin{array}{rl} S_{\mathcal{A}\_quant} = & \{\langle j, \exists Q.C\rangle \mid \exists Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ Q \text{ a basic role}, 1 \leq j \leq |\mathcal{J}|\} \cup \\ & \{\langle j, \forall Q.C\rangle \mid \forall Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ Q \text{ a basic role}, 1 \leq j \leq |\mathcal{J}|\} \end{array}$$

  These special states allow to automaton to check whether concepts of the form $\exists Q.C$ and $\forall Q.C$ are satisfied by some ABox individual a_j.

$$\begin{array}{rl} S_{\mathcal{A}\_num} = & \{\langle m, \geqslant n\, Q.C, i, j\rangle \mid \geqslant n\, Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ 1 \leq m, i \leq |\mathcal{J}|,\ 0 \leq j \leq n\} \cup \\ & \{\langle m, \leqslant n\, Q.C, i, j\rangle \mid \leqslant n\, Q.C \in Cl^{nnf}(C_{\mathcal{T}}),\ 1 \leq m, i \leq |\mathcal{J}|; 0 \leq j \leq n{+}1\} \end{array}$$

  The states in $S_{\mathcal{A}\_num}$ are used to verify the satisfaction of the number restrictions by the ABox individuals. Similarly as above, i stores how many successors of the individual a_m have been navigated, and j how many of them are reached through Q and labeled with C.

- $F_{\mathcal{K}} = \{\forall R^*.C \mid \forall R^*.C \in Cl^{nnf}(C_{\mathcal{T}})\}$ is a Büchi acceptance condition.

  Observe that concepts of the form $\exists R^*.C$ are not final states. This is sufficient to guarantee that such concepts are satisfied in all accepting runs of the automaton [4].

- The transition function $\delta : S_{\mathcal{K}} \times \Sigma_{\mathcal{K}} \to \mathcal{B}([k] \times S_{\mathcal{K}})$, where $k = max(k_{C_{\mathcal{T}}}, |\mathcal{J}|)$, is defined as follows:

1. First of all, the automaton will verify that the root contains r, that the level one nodes properly represent the individuals in the ABox, that all ABox assertions are satisfied, and that every non-dummy

node at level one is the root of a tree representing a model of $C_{\mathcal{T}}$. Thus, for each σ in $\sigma \in \Sigma_{\mathcal{K}}$ with $r \in \sigma$ and $\sigma \cap \{\mathcal{C}_{\mathcal{K}} \cup \mathcal{R}_{\mathcal{K}}\} = \emptyset$ we define a transition from the initial state s_0

$$\delta(s_0, \sigma) \;=\; B_1 \wedge \cdots \wedge B_8$$

where:

$$B_1 = \bigwedge_{1 \leq i \leq |\mathcal{J}|}((\bigvee_{1 \leq j \leq |\mathcal{J}|}(i, a_j) \wedge (i, \neg d)) \vee (i, d))$$
$$B_2 = \bigwedge_{1 \leq i \leq |\mathcal{J}|} \bigvee_{1 \leq j \leq |\mathcal{J}|}(j, a_i)$$
$$B_3 = \bigwedge_{1 \leq i < j \leq |\mathcal{J}|}(\bigwedge_{1 \leq k \leq |\mathcal{J}|}(i, \neg a_k) \vee (j, \neg a_k))$$
$$B_4 = \bigwedge_{1 \leq i \leq |\mathcal{J}|}((i, \neg r) \wedge (i, s_1))$$
$$B_5 = \bigwedge_{a_i \neq a_j \in \mathcal{A}}(\bigwedge_{1 \leq k \leq |\mathcal{J}|}(k, \neg a_i) \vee (k, \neg a_j))$$
$$B_6 = \bigwedge_{A(a_j) \in \mathcal{A}}(\bigvee_{1 \leq i \leq |\mathcal{J}|}(i, a_j) \wedge (i, A))$$
$$B_7 = \bigwedge_{P(a_i, a_j) \in \mathcal{A}}(0, Pij)$$
$$B_8 = \bigwedge_{1 \leq i \leq |\mathcal{J}|}((i, nnf(C_{\mathcal{T}})) \vee (i, d))$$

- B_1 ensures that each level one node stands for some individual a_k, in which case the label contains a_k and does not contain d, or it is a dummy child labeled d.
- B_2 verifies that the label identifying each individual a_k occurs in some level one node.
- B_3 verifies that a label identifying an individual does not occur in two different level one nodes.
- B_4 checks that on each level one node the label does not contain r, and also moves on each level one node to the state s_1. From s_1 it then further checks that r and each $a_i \in \mathcal{J}$ do not appear at any node below level one in the whole tree. This is accomplished by the following transitions, one for each $\sigma \in \Sigma_{\mathcal{K}}$:

$$\delta(s_1, \sigma) = \bigwedge_{1 \leq i \leq k_{C_{\mathcal{T}}}}((i, \neg r) \wedge (\bigwedge_{1 \leq j \leq |\mathcal{J}|}(i, \neg a_j)) \wedge (i, s_1))$$

- B_5 checks, for each inequality assertion $a_i \neq a_j$ in $\mathcal{A}$, that the labels a_i, a_j do not occur in the label of the same level one node.
- B_6 ensures that the assertions of the form $A(a_i)$ in $\mathcal{A}$ are satisfied, by verifying, for each such assertion, that every node labeled a_i is an instance of A.
- B_7 verifies that each assertions of the form $P(a_i, a_j)$ is satisfied, by going to the state Pij.
- Finally, B_8 checks that every non-dummy node at level one is the root of a tree representing a model of the concept $C_{\mathcal{T}}$.

Note that conditions B_1–B_4 do not depend on the actual ABox assertions. Among these, B_1–B_3 ensure that the level one nodes properly represent the individuals in $\mathcal{J}$. Instead, conditions B_5–B_7 guarantee that all ABox assertions are satisfied, while condition B_8 guarantees that the TBox assertions are satisfied.

2. Now we define transitions that inductively decompose concepts and roles, and move to appropriate states of the automaton and nodes of the tree. Note that the transitions involving $\forall R^*.C$ and $\exists R^*.C$ are slightly different from the others. Instead of decomposing the concept, they treat $\forall R^*.C$ as the equivalent concept $C \sqcap \forall R.\forall R^*.C$, and $\exists R^*.C$ as $C \sqcup \exists R.\exists R^*.C$.

   For each concept or basic role in $Cl^{nnf}(C_{\mathcal{T}})$ and each $\sigma \in \Sigma_{\mathcal{K}}$ there are transitions:

$$\begin{aligned}
\delta(C \sqcap C', \sigma) &= (0, C) \ \wedge \ (0, C') \\
\delta(C \sqcup C', \sigma) &= (0, C) \ \vee \ (0, C') \\
\delta(\forall (R \cup R').C, \sigma) &= (0, \forall R.C) \ \wedge \ (0, \forall R'.C) \\
\delta(\forall (R \circ R').C, \sigma) &= (0, \forall R.\forall R'.C) \\
\delta(\forall R^*.C, \sigma) &= (0, C) \ \wedge \ (0, \forall R.\forall R^*.C) \\
\delta(\forall id(C).C', \sigma) &= (0, nnf(\neg C)) \ \vee \ (0, C') \\
\delta(\exists (R \cup R').C, \sigma) &= (0, \exists R.C) \ \vee \ (0, \exists R'.C) \\
\delta(\exists (R \circ R').C, \sigma) &= (0, \exists R.\exists R'.C) \\
\delta(\exists R^*.C, \sigma) &= (0, C) \ \vee \ (0, \exists R.\exists R^*.C) \\
\delta(\exists id(C).C', \sigma) &= (0, C) \ \wedge \ (0, C') \\
\delta(Q \cap Q', \sigma) &= (0, Q) \ \wedge \ (0, Q') \\
\delta(Q \cup Q', \sigma) &= (0, Q) \ \vee \ (0, Q') \\
\delta(Q \setminus Q', \sigma) &= (0, Q) \ \wedge \ (0, \neg Q')
\end{aligned}$$

   Additionally, for each basic role in $Cl^{nnf}(C_{\mathcal{T}})$, each $\sigma \in \Sigma_{\mathcal{K}}$ and each $i, j \in \mathcal{J}$ there are transitions:

$$\begin{aligned}
\delta(Q \cap Q'ij, \sigma) &= (0, Qij) \ \wedge \ (0, Q'ij) \\
\delta(Q \cup Q'ij, \sigma) &= (0, Qij) \ \vee \ (0, Q'ij) \\
\delta(Q \setminus Q'ij, \sigma) &= (0, Qij) \ \wedge \ (0, \neg Q'ij)
\end{aligned}$$

   Remember that in $Q \setminus Q'$, the roles Q and Q' are either atomic or inverse of atomic, and hence $\neg Q'$ (resp. $\neg Q'ij$) is indeed a state of the automaton.

3. To verify that a concept of the form $\forall Q.C$ or $\exists Q.C$ is satisfied by a node x, (where Q is a basic role), all the nodes that reach or are reachable from x must be navigated. We need different transitions (*i*) for a node x at level one, and (*ii*) for a node x at any other level. In case (*ii*), the predecessor and the successors of x are navigated as usual. In case (*i*), the transitions must consider the other individual nodes that are connected to x via some role, which is recorded in the root label. Therefore, the transitions must send suitable copies of the automaton to navigate the successors, and send a copy of the automaton up to the root, moving to the special states in $S_{\mathcal{A}\text{-}quant}$. Thus, we define the following transitions, for each concept of the form $\exists Q.C$ or $\forall Q.C$ in $Cl^{nnf}(C_{\mathcal{T}})$ and each $\sigma \in \Sigma_{\mathcal{K}}$.

   First, if $\sigma \cap (\mathcal{J} \cup \{d\}) = \emptyset$ (i.e., case (*ii*) above), we define:

$$\begin{aligned}
\delta(\exists Q.C, \sigma) &= ((0, Q^-) \wedge (-1, C)) \ \vee \ \textstyle\bigvee_{1 \le i \le k_{C_{\mathcal{T}}}} ((i, Q) \wedge (i, C)) \\
\delta(\forall Q.C, \sigma) &= ((0, nnf(\neg Q^-)) \vee (-1, C)) \ \wedge \ \textstyle\bigwedge_{1 \le i \le k_{C_{\mathcal{T}}}} ((i, nnf(\neg Q)) \vee (i, C))
\end{aligned}$$

   Otherwise, if $\sigma \cap (\mathcal{J} \cup \{d\}) \neq \emptyset$ (case (*i*) above), we have transitions:

$$\delta(\exists Q.C,\sigma) = \bigvee_{a_j\in\sigma}(-1,\langle j,\exists Q.C\rangle) \vee \bigvee_{1\leq i\leq k_{C_\mathcal{T}}}((i,Q)\wedge(i,C))$$

$$\delta(\forall Q.C,\sigma) = \bigwedge_{a_j\in\sigma}(-1,\langle j,\forall Q.C\rangle) \wedge \bigwedge_{1\leq i\leq k_{C_\mathcal{T}}}((i,nnf(\neg Q))\vee(i,C))$$

For each $\sigma\in\Sigma_\mathcal{K}$, and each $\langle j,\exists Q.C\rangle$ or $\langle j,\forall Q.C\rangle$ in $S_{\mathcal{A}\_quant}$, there is a transition

$$\delta(\langle j,\exists Q.C\rangle,\sigma) = \bigvee_{1\leq i\leq|\mathcal{J}|}\ (\ \bigvee_{1\leq k\leq|\mathcal{J}|}((0,Q_{jk})\wedge(i,a_k)\wedge(i,C)))$$

$$\delta(\langle j,\forall Q.C\rangle,\sigma) = \bigwedge_{1\leq i\leq|\mathcal{J}|}\ (\ \bigwedge_{1\leq k\leq|\mathcal{J}|}((0,nnf(\neg Qjk))\vee(i,\neg a_k)\vee(i,C)))$$

4. For number restrictions, the transitions are slightly more involved since we need to encode a counter into the automaton. Intuitively, a state $\langle \gtrless n\,Q.C,i,j\rangle$, for $\gtrless\in\{\geqslant,\leqslant\}$, is used to check whether a qualified number restriction $\gtrless nQC$ is satisfied in a node x by counting the number of nodes reached from x through Q in which C holds. More precisely, when the automaton is in a state of the form $\gtrless nQC$ and visiting a node x, it will change to the state $\langle \gtrless n\,Q.C,0,0\rangle$. Then it will suitably navigate the neighbours of the node, using the first counter to keep track of those that have already been visited, and the second one to account for those that are reachable through the relation Q and which are an instance of C. Thus the automaton will be in a state $\langle \gtrless n\,Q.C,i,j\rangle$ if among the first $i-1$ neighbours of x there are j nodes reached from $i{\cdot}x$ through Q in which C holds.

   For each concept of the form $\gtrless n\,Q.C$ in $Cl^{nnf}(C_\mathcal{T})$ and each $\sigma\in\Sigma_\mathcal{K}$ there is a transition:

$$\delta(\gtrless n\,Q.C,\sigma) \quad = \quad (0,\langle \gtrless n\,Q.C,0,0\rangle)$$

   Once the counters are set to 0 by the above transition, the automaton starts navigating the successors of the node, which are at most $k_{C_\mathcal{T}}$. This is done with a set of transitions of the following form:

$$\begin{aligned}\delta(\langle \gtrless n\,Q.C,i,j\rangle,\sigma) \quad = \quad & (((i{+}1,nnf(\neg Q))\vee(i{+}1,nnf(\neg C)))\wedge(0,\langle \gtrless n\,Q.C,i{+}1,j\rangle))\vee\\ & ((i{+}1,Q)\wedge(i{+}1,C)\wedge(0,\langle \gtrless n\,Q.C,i{+}1,j{+}1\rangle))\end{aligned}$$

   for all states in S_{num}, with the counters ranging over the following values:

   - $0\leq i<k_{C_\mathcal{T}}$
     i stores how many successors have been counted, and checks the $(i+1)$-th
   - if $\gtrless$ is $\geqslant$, then $0\leq j<n$
     We stop counting if we reach n, as we already know the at-least restriction is satisfied
   - Otherwise, if $\gtrless$ is $\leqslant$, then $0\leq j\leq n$
     We can stop counting if we reach $n+1$, as we know that the at-least restrictions is not satisfied.

After counting the successors of a node, we have to distinguish between the level one nodes and the remaining 'ordinary' nodes. In the latter case the automaton simply moves up and takes the predecessor of the node into account.

Thus, if $\sigma \cap (\mathcal{J} \cup \{d\}) = \emptyset$, we define:

$$\begin{array}{rcl}\delta(\langle \gtrless n\, Q.C, k_{C_\mathcal{T}}, j\rangle, \sigma) & = & (((0, nnf(\neg Q^-)) \vee (-1, nnf(\neg C))) \wedge (0, \langle \gtrless n\, Q.C, k_{C_\mathcal{T}}{+}1, j\rangle)) \vee \\ & & ((0, Q^-) \wedge (-1, C) \wedge (0, \langle \gtrless n\, Q.C, k_{C_\mathcal{T}}{+}1, j{+}1\rangle))\end{array}$$

with j as above, i.e., $0 \leq j < n$ if $\gtrless$ is $\geqslant$, and $0 \leq j \leq n$ otherwise.

For the level one nodes, the automaton moves up to the root and to a state in $S_{\mathcal{A}\_num}$. Note that the first counter is restarted in order to start navigating the level one nodes, while the second counter still stores the number of elements that has been accounted for so far. From this state, the automaton counts how many level one nodes represent an individual which is related to x by the role Q and is an instance of C.

If $\sigma \cap (\mathcal{J} \cup \{d\}) \neq \emptyset$:

$$\delta(\langle \gtrless n\, Q.C, k_{C_\mathcal{T}}, j\rangle, \sigma) \;=\; \textstyle\bigwedge_{a_i \in \sigma}(-1, \langle i, \gtrless nQ.C, 0, j\rangle)$$

and, for all states in $S_{\mathcal{A}\_num}$, we define:

$$\begin{array}{rcl}\delta(\langle m, \gtrless nQ.C, i, j\rangle, \sigma) & = & ((\bigwedge_{1\leq k\leq|\mathcal{J}|}((i, \neg a_k) \vee (0, nnf(\neg Q_{mk}))) \vee (i, \neg C)) \wedge \\ & & \quad (0, \langle m, \gtrless nQ.C, i+1, j\rangle)) \vee \\ & & ((\bigvee_{1\leq k\leq|\mathcal{J}|}((i, a_k) \wedge (0, Q_{mk})) \wedge (i, C)) \wedge \\ & & \quad (0, \langle m, \gtrless nQ.C, i+1, j+1\rangle))\end{array}$$

with $1 \leq m, i \leq |\mathcal{J}|$, if $\gtrless$ is $\geqslant$, then $0 \leq j < n$, and if $\gtrless$ is $\leqslant$, then $0 \leq j \leq n$.

Once all the necessary nodes have been navigated the (un)satisfiaction of the number restrictions can be established:

$$\begin{array}{rcll}\delta(\langle \geqslant n\, Q.C, i, n\rangle, \sigma) & = & \textbf{true}, & \text{for } 0 \leq i \leq k_{C_\mathcal{T}}{+}1 \\ \delta(\langle \geqslant n\, Q.C, k_{C_\mathcal{T}}{+}1, j\rangle, \sigma) & = & \textbf{false}, & \text{for } 0 \leq j \leq n{-}1 \\ \delta(\langle \leqslant n\, Q.C, i, n{+}1\rangle, \sigma) & = & \textbf{false}, & \text{for } 0 \leq i \leq k_{C_\mathcal{T}}{+}1 \\ \delta(\langle \leqslant n\, Q.C, k_{C_\mathcal{T}}{+}1, j\rangle, \sigma) & = & \textbf{true}, & \text{for } 0 \leq j \leq n \\ & & & \\ \delta(\langle m, \geqslant n\, Q.C, i, n\rangle, \sigma) & = & \textbf{true}, & \text{for } 1 \leq i \leq |\mathcal{J}| \\ \delta(\langle m, \geqslant n\, Q.C, |\mathcal{J}|+1, j\rangle, \sigma) & = & \textbf{false}, & \text{for } 0 \leq j \leq n{-}1 \\ \delta(\langle m, \leqslant n\, Q.C, i, n+1\rangle, \sigma) & = & \textbf{false}, & \text{for } 1 \leq i \leq |\mathcal{J}|{+}1 \\ \delta(\langle m, \leqslant n\, Q.C, |\mathcal{J}|{+}1, j\rangle, \sigma) & = & \textbf{true}, & \text{for } 0 \leq j \leq n.\end{array}$$

5. Concepts and roles are recursively decomposed as explained above. When reaching the atomic level, it is checked whether the node label σ contains the corresponding atomic symbol. Thus, for each $\sigma \in \Sigma_\mathcal{K}$, and each $s \in \mathcal{C}_\mathcal{K} \cup \mathcal{R}_\mathcal{K} \cup \mathcal{J} \cup \{d\}$ there are transitions:

$$\delta(s, \sigma) = \begin{cases}\textbf{true}, & \text{if } s \in \sigma \\ \textbf{false}, & \text{if } s \notin \sigma\end{cases} \qquad \text{and} \qquad \delta(\neg s, \sigma) = \begin{cases}\textbf{true}, & \text{if } s \notin \sigma \\ \textbf{false}, & \text{if } s \in \sigma\end{cases}$$

6. Finally, further transitions verify whether ABox individuals are connected via some atomic or inverse of atomic role by checking the label of the root. For each $\sigma \in \Sigma_{\mathcal{K}}$ and $Pij \in S_{\mathcal{A}\_role}$ with $P \in \mathcal{R}_{\mathcal{K}}$ there is a transition:

$$\delta(Pij, \sigma) = \begin{cases} \textbf{true}, & \text{if } (Pij \in \sigma) \text{ or } (P^{-}ji \in \sigma) \\ \textbf{false}, & \text{otherwise} \end{cases}$$

Roughly, a run of $\mathbf{A}_{\mathcal{K}}$ on an infinite tree $\mathbf{T}$ starts in the root, checks that the ABox is properly represented and moves to each node representing some individual in order to check that $C_{\mathcal{T}}$ holds there (item 1 above). To this end, $nnf(C_{\mathcal{T}})$ is recursively decomposed while appropriately navigating the tree (items 2, 3 and 4) until $\mathbf{A}_{\mathcal{K}}$ arrives at atomic elements, which are checked locally (items 5 and 6).

3.3 Soundness and Completeness

Satisfiability of an $\mathcal{ALCQI}b_{reg}$ KB $\mathcal{K}$ can be decided by testing the automaton $\mathbf{A}_{\mathcal{K}}$ for emptiness, since there is a direct correspondence between canonical tree models of $\mathcal{K}$ and the labeled trees accepted by $\mathbf{A}_{\mathcal{K}}$. Indeed, a labeled tree $\mathbf{T} = (T, V)$ over the alphabet Σ_K can represent an interpretation $\mathcal{I}_{\mathbf{T}}$ for $\mathcal{K}$. Furthermore, if $\mathbf{T} = (T, V)$ is accepted by $\mathbf{A}_{\mathcal{K}}$, then it represents a canonical model of $\mathcal{K}$.

We use the notion of a *quasi-interpretation* $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which is very similar to an interpretation. The only difference is that in a quasi-interpretation, $a_i^{\mathcal{I}}$ is a subset of $\Delta^{\mathcal{I}}$ for any individual a_i, instead of a single element of $\Delta^{\mathcal{I}}$. *Canonical quasi-interpretations* are defined similarly to canonical interpretations, only that instead of item 2 in Definition 3.2, they satisfy that $a_i^{\mathcal{I}} \subseteq \{1, \ldots, n\}$ for each $a_i \in \mathcal{J}$.

Now we define the quasi-interpretation $\mathcal{I}_{\mathbf{T}}$ represented by a tree $\mathbf{T}$. Informally, its domain $\Delta^{\mathcal{I}_{\mathbf{T}}}$ is given by the nodes x in $\mathbf{T}$ with $a_i \in V(x)$ for some individual a_i, and the nodes in $\mathbf{T}$ that are reachable from any such x through the roles. The extensions of individuals, concepts and roles are determined by the labels of the nodes in $\mathbf{T}$.

Definition 3.5 A labeled tree $\mathbf{T} = (T, V)$ over the alphabet $\Sigma_{\mathcal{K}}$ is called a *quasi-interpretation tree* if:

- the branching degree is $|\mathcal{J}|$ at the first level and $k_{C_{\mathcal{T}}}$ at all the other levels,
- $r \in V(\varepsilon)$ an $r \notin V(x)$ for every $x \neq \varepsilon$,
- for every level one node $1 \leq x \leq |\mathcal{J}|$, $(\{d\} \cup \mathcal{J}) \cap V(x) \neq \emptyset$ and $\{d, a_i\} \not\subseteq V(x)$ for each $a_i \in \mathcal{J}$,
- $(\{d\} \cup \mathcal{J}) \cap V(x{\cdot}j) = \emptyset$ for every $x \neq \varepsilon$, $j > 0$.

W.l.o.g. we assume that for any pair x, y of level one nodes $x, y \in \{1, \ldots, |\mathcal{J}|\}$, if $a_i \in V(x)$ for some $a_i \in \mathcal{J}$ and $d \in V(y)$, then $x < y$.

Let $\mathbf{T} = (T, V)$ be a quasi-interpretation tree. For each atomic role P, we define:

$$\mathcal{R}_P^1 = \{(x, xi) \mid P \in V(xi)\} \cup \{(xi, x) \mid P^{-} \in V(xi)\}$$

$$\mathcal{R}_P^2 = \{(i, j) \mid Pij \in V(\varepsilon)\} \cup \{(j, i) \mid P^{-}ij \in V(\varepsilon)\}$$

Intuitively, $\mathcal{R}_P^1$ contains all the pairs of nodes (x, y) that are in the extension of the atomic role P, where at least one of x and y is not a level one node. $\mathcal{R}_P^2$ contains all the pairs of level one nodes that are related by the role P. Together, $\mathcal{R}_P^1$ and $\mathcal{R}_P^2$ determine the extension of the atomic role P.

We also define the set $\mathbf{I}$ of the nodes in $\mathbf{T}$ that interpret some individual in $\mathcal{J}$:

$$\mathbf{I} = \{\ x \mid x \in \{1, \ldots, |\mathcal{J}|\}, a_i \in V(x) \text{ for some } a_i \in \mathcal{J}\ \}$$

Note that $\mathbf{I}$ is actually a set of the form $\{1, \ldots, n\}$ for some $n \leq |\mathcal{J}|$. To define the domain of $\Delta^{\mathcal{I}_\mathbf{T}}$, we consider the nodes in $\mathbf{I}$ and those that are in a subtree starting at some element of $\mathbf{I}$ and are reachable though a sequence of roles from $\mathcal{K}$, i.e., the nodes that comprise the tree-shaped models of $C_\mathcal{T}$ rooted at each element of $\mathbf{I}$. For each $i \in \mathbf{I}$, the set of all such nodes can be constructed as follows:

$$\mathbf{D}_i = \{\ x' \mid (i, x') \in (\bigcup_{P \in \mathcal{R}_\mathcal{K}} (\mathcal{R}_P^1 \cup \mathcal{R}_P^{1^-}))^*\ \}$$

Clearly, the set $\mathbf{D}_i$ is a tree over $i \cdot \{1, \ldots, k_{C_\mathcal{T}}\}$ whose root is i.

Definition 3.6 Given a quasi-interpretation tree $\mathbf{T}$, the quasi-interpretation $\mathcal{I}_\mathbf{T}$ for $\mathcal{K}$ is defined as follows:

$$\begin{array}{rcl}
\Delta^{\mathcal{I}_\mathbf{T}} & = & \mathbf{I} \cup \bigcup_{i \in \mathbf{I}} \mathbf{D}_i \\
a_i^{\mathcal{I}_\mathbf{T}} & = & \mathbf{I} \cap \{\ x \mid a_i \in V(x)\ \}, \quad \text{for each } a_i \in \mathcal{J} \\
A^{\mathcal{I}_\mathbf{T}} & = & \Delta^{\mathcal{I}_\mathbf{T}} \cap \{\ x \mid A \in V(x)\ \}, \quad \text{for each atomic concept } A \\
P^{\mathcal{I}_\mathbf{T}} & = & (\Delta^{\mathcal{I}_\mathbf{T}} \times \Delta^{\mathcal{I}_\mathbf{T}}) \cap \mathcal{R}_P, \quad \text{for each atomic role } P, \text{ where } \mathcal{R}_P = \mathcal{R}_P^1 \cup \mathcal{R}_P^2.
\end{array}$$

Note that, by definition, the interpertation $\mathcal{I}_\mathbf{T}$ represented by a quasi-interpretation tree satisfies items 1 and 3 in Definition 3.2, and it also satisfies $a_i^\mathcal{I} \subseteq \{1, \ldots, n\}$ for all individuals. Therefore, $\mathcal{I}_\mathbf{T}$ is a canonical quasi-interpretation.

Proposition 3.7 *Let $\mathbf{T}$ be a labeled tree accepted by $\mathbf{A}_\mathcal{K}$. Then $\mathcal{I}_\mathbf{T}$ is a model of $\mathcal{K}$.*

Proof. Let $\mathbf{T} = (T, V)$ be a labeled tree accepted by $\mathbf{A}_\mathcal{K}$. Then $\mathcal{I}_\mathbf{T}$ is a canonical quasi-interpretation. Furthermore, note that the transition function of $\mathbf{A}_\mathcal{K}$ (item 1, B_1 to B_3) is defined in such a way that each $a_i \in \mathcal{J}$ occurs in the label of exactly one level one node in any tree accepted by $\mathbf{A}_\mathcal{K}$. Since only such nodes can occur in $a_i^{\mathcal{I}_\mathbf{T}}$ for an $a_i \in \mathcal{J}$, $a_i^{\mathcal{I}_\mathbf{T}}$ is a singleton. Thus $\mathcal{I}_\mathbf{T}$ is a canonical interpretation for $\mathcal{K}$.

To see that $\mathcal{I}_\mathbf{T}$ is a model of $\mathcal{K}$, we have to show that (*i*) $\mathcal{I}_\mathbf{T} \models \mathcal{A}$ and that (*ii*) for each $a_i \in \mathcal{J}$, $a_i^{\mathcal{I}_\mathbf{T}} \in C_\mathcal{T}^{\mathcal{I}_\mathbf{T}}$ is a model of $\mathcal{K}$.

First we prove (*i*). Note that item 1 in the transition function of $\mathbf{A}_\mathcal{K}$, together with items 5 and 6 (which ensure that $(\delta(x, E) = \textbf{true}$ iff $E \in V(x)$ for any node x and element of the node labels E), ensure that in every tree $\mathbf{T}$ accepted by $\mathbf{A}_\mathcal{K}$ the following hold:

- If $a_i \neq a_j \in \mathcal{A}$, then there is no node $k \in \{1, \ldots, n\}$ in $\mathbf{T}$ such that $\{a_i, a_j\} \subseteq V(k)$ (by B_5).
- If $A(a_i) \in \mathcal{A}$, then there is some node $k \in \{1, \ldots, n\}$ in $\mathbf{T}$ such that $\{a_i, A\} \subseteq V(k)$ (by B_6).
- for each $P(a_i, a_j) \in \mathcal{A}$, $Pij \in V(\varepsilon)$ (by B_7).

Clearly, by construction of $\mathcal{I}_\mathbf{T}$, this is enough to ensure that every assertion in the ABox is satisfied and $\mathcal{I}_\mathbf{T} \models \mathcal{A}$. In order to prove (*ii*), first observe that by item 1 (B_8) in the definition of the transition function, if $\mathbf{T}$ is accepted by $\mathbf{A}_\mathcal{K}$, then $(i, C_\mathcal{T})$ must evaluate to true for every $i \in \{1, \ldots, |\mathcal{J}|\}$ such that i is in $\Delta^{\mathcal{I}_\mathbf{T}}$. We will show in the following lemma that this implies $i \in C_\mathcal{T}^{\mathcal{I}_\mathbf{T}}$, and thus it is enough to ensure that (*ii*) holds.

Lemma 3.8 *Let $\mathbf{T}$ be a tree accepted by $\mathbf{A}_\mathcal{K}$, let (T, r) be an accepting run of $\mathbf{A}_\mathcal{K}$ over $\mathbf{T}$ and let y be a node of T with $r(y) = (x, E)$, for some node x of $\mathbf{T}$, $x \neq \varepsilon$, and some $E \in Cl(C_T)$ a concept, individual, or basic role. Then $r(y) = (x, E)$ evaluates to true in the run iff*

- *$x \in E^{\mathcal{I}_\mathbf{T}}$ when E is a concept or an individual; and*
- *$\langle x', x\rangle \in E^{\mathcal{I}_\mathbf{T}}$ when E is a role, x' is the predecessor of x in $\mathbf{T}$ and $x' \neq \varepsilon$.*

The proof of the lemma is a straightforward induction on the structure of E. By the definition of $\mathbf{T}_\mathcal{I}$ and the definition of the transition function (item 5) it is trivial for atomic E. For complex concepts and roles, it also follows from the definition of the transition function, particularly item 2, which decomposes them accordingly. Some cases are slightly more involved. For concepts involving quantifiers and number restrictions, the items 3 and 4 must be considered. In both cases the level one nodes must be treated as special cases requiring item 6.

Most cases are straightforward. Only concepts of the form $\forall R^*.C$ and $\exists R^*.C$ are slightly different. These are propagated using the equivalent concepts $C \sqcap \forall R.\forall R^*.C$ and $C \sqcup \exists R.\exists R^*.C$, respectively. Only these concepts may generate infinite branches in a run. In such a branch, $\exists R^*.C$ could always be resolved by choosing the disjunct $\exists R.\exists R^*.C$ but never the disjunct C. The branch then corresponds to an infinite path in the tree $\mathbf{T}$ in which R is iterated forever and C never satisfied. Since the semantics of $\exists R^*.C$ means that C is satisfied after finitely many iterations of R, this path cannot be used to satisfy $\exists R^*.C$. The acceptance condition of $\mathbf{A}_\mathcal{K}$, which requires that each infinite branch contains only states of the form $\forall R^*.C$ infinitely often (and thus each $\exists R^*.C$ will stop occurring at some point on the path) rules out such infinite branches in accepting runs, ensuring that all concepts of the form $\forall R^*.C$ and $\exists R^*.C$ are satisifed. □

Conversely, if $\mathcal{I}$ is a canonical model of $\mathcal{K}$, then $\mathbf{T}_\mathcal{I}$, the tree that represents $\mathcal{I}$ (Definition 3.4) is accepted by $\mathbf{A}_\mathcal{K}$.

Proposition 3.9 *Let $\mathcal{I}$ be a canonical model of $\mathcal{K}$. Then $\mathbf{A}_\mathcal{K}$ accepts $\mathbf{T}_\mathcal{I}$.*

Proof. We have to prove that $\mathbf{A}_\mathcal{K}$ has an accepting run on $\mathbf{T}_\mathcal{I}$. This is the case, since we defined $\mathbf{A}_\mathcal{K}$ in such a way that it will always accept a tree if it manages to verify that the tree represents a model of the knowledge base. The accepting run will start at (ε, s_0). Since the root ε of $\mathbf{T}_\mathcal{I}$ is marked with r and $r \cap \{\mathcal{C}_\mathcal{K} \cup \mathcal{R}_\mathcal{K}\} = \emptyset$, then a transition of the form $\delta(s_0, V(\varepsilon)) = B_1 \wedge \cdots \wedge B_8$ will be defined. Since $\mathcal{I}$ is a canonical model of $\mathcal{K}$, then for every $a_i \in \mathcal{J}$ there will be exactly one j in $\mathbf{T}_\mathcal{I}$ such that $a_i \in V(j)$. Moreover, for every j, $0 \leq j \leq |\mathcal{J}|$, $d \in V(j)$ iff $a_i \notin V(j)$ for every $a_i \in \mathcal{J}$. This ensures that in the run there will be suitable successors of (ε, s_0) of the form (j, a_i), $(j, \neg a_i)$, (j, d) and $(j, \neg d)$ as required by items B_1 to B_3. Similarly, since $\mathcal{I}_\mathbf{T} \models \mathcal{A}$, whenever $a_i \neq a_j \in \mathcal{A}$, there will be no level one node k such that $a_i \in V(k)$ and $a_j \in V(k)$. This ensures that the automaton can navigate to every such k and move either to the state a_i or to the state a_j, as required by B_5. Also, since $\mathcal{I} \models A(a_j)$ for every assertion $A(a_j) \in \mathcal{A}$, the automaton can move to the node i with $a_j \in V(i)$ and change to state A in order to satisfy B_6. As $\mathcal{I} \models P(a_i, a_j)$ for every such ABox assertion, $Pij \in V(\varepsilon)$ will hold by definition of $\mathbf{T}_\mathcal{I}$, and thus the run can successfully satisfy B_7. Finally, as $\mathcal{I}$ is a canonical model of $\mathcal{K}$, $a_i^{\mathcal{I}_\mathbf{T}} \in C_\mathcal{I}^{\mathcal{I}_\mathbf{T}}$ for every $a_i \in \mathcal{J}$. Thus the automaton can send suitable copies to every level one node j, and either move to state $C_\mathcal{I}$ if $a_i \in V(j)$ for some ABox individual, or move to state d otherwise, ensuring that B_8 is also satisfied. The rest of the run is just an inductive decomposition of the concept $C_\mathcal{I}$, starting at every level one node which interprets some ABox individual. Since every such individual is indeed an instance of $C_\mathcal{I}$, by Lemma 3.8 we know that the run concludes successfully and $\mathbf{A}_\mathcal{K}$ accepts $\mathbf{T}_\mathcal{I}$.

□

From Propositions 3.7 and 3.9 and the canonical model property of $\mathcal{ALCQI}b_{reg}$(Theorem 3.3), we get the following result:

Theorem 3.10 *An $\mathcal{ALCQI}b_{reg}$KB $\mathcal{K}$ is satisfiable iff the set of trees accepted by $\mathbf{A}_{\mathcal{K}}$ is nonempty.*

3.4 Complexity

Recall that $\mathcal{C}_{\mathcal{K}}$ and $\mathcal{R}_{\mathcal{K}}$ represent respectively the atomic concepts in $\mathcal{K}$ and the atomic roles in $\mathcal{K}$ together with their inverses; $\mathcal{J}$ represents the ABox individuals in $\mathcal{K}$; $\mathbf{n}$ represents the maximal n occurring in a concept of the form $\geqslant n\, Q.C$ in $Cl^{nnf}(C_{\mathcal{T}})$, and $k_{C_{\mathcal{T}}} = |Cl^{nnf}(C_{\mathcal{T}})| \times \mathbf{n}$. Clearly, all of $|\mathcal{C}_{\mathcal{K}}|$, $|\mathcal{R}_{\mathcal{K}}|$ and $|\mathcal{J}|$ are linear in the size of $\mathcal{K}$, and so is $|Cl(C_{\mathcal{T}})|$. Moreover, assuming that the numbers in the number restrictions are encoded in unary, $\mathbf{n}$ is linear and $k_{C_{\mathcal{T}}}$ quadratic in the size of $\mathcal{K}$. Under this assumptions, we obtain:

- The size $\Sigma_{\mathcal{K}}$ is bounded by $2^{O(M)}$, where $M = |\mathcal{C}_{\mathcal{K}}| + |\mathcal{R}_{\mathcal{K}}| + |\mathcal{J}| + |PI|$ and $|PI| = |\mathcal{R}_{\mathcal{K}}| \times |\mathcal{J}|^2$, so we have that $\Sigma_{\mathcal{K}}$ is single exponential in the size of $\mathcal{K}$.
- The number of states in $S_{\mathcal{K}}$ is polynomial in the size of $\mathcal{K}$.
  More specifically, $|S_{\mathcal{K}}| = |Cl(C_{\mathcal{T}}, \mathcal{J}, d)| + |S_{num}| + |S_{\mathcal{A}\_role}| + |S_{\mathcal{A}\_quant}| + |S_{\mathcal{A}\_num}| + 1$, and we have the following bounds:

$$\begin{array}{rcl} |S_{num}| & \leq & |Cl(C_{\mathcal{T}})|^2 \times \mathbf{n}^2 \\ |S_{\mathcal{A}\_role}| & \leq & |Cl(C_{\mathcal{T}})| \times |\mathcal{J}|^2 \\ |S_{\mathcal{A}\_quant}| & \leq & |Cl(C_{\mathcal{T}})| \times |\mathcal{J}| \\ |S_{\mathcal{A}\_num}| & \leq & |Cl(C_{\mathcal{T}})| \times |\mathcal{J}|^2 \times \mathbf{n} \end{array}$$

  So the cardinality $S_{\mathcal{K}}$ is bounded by $O(L^2)$, where $L = |Cl(C_{\mathcal{T}})| + |\mathcal{J}| + \mathbf{n}$.
- Any Büchi condition F can be converted into a parity one of length 3 by taking $G_1 = \emptyset$, $G_2 = F$, $G_3 = Q$ (see 2.2). Thus the automaton $\mathbf{A}_{\mathcal{K}}$ has a parity acceptance condition of constant size.

Thus, by Theorems 2.3 and 3.10, we get an optimal upper bound for KB satisfiability.

Corollary 3.11 *For $\mathcal{ALCQI}b_{reg}$, KB satisfiability is decidable in* EXPTIME.

This is worst-case optimal, since a matching lower bound holds already for much weaker DLs [1].

4 Query answering via automata

We address now the problem of entailment of P2RPQs in $\mathcal{ALCQI}b_{reg}$KBs. Consider a (Boolean) P2RPQ q over a KB $\mathcal{K}$. We first show that, in order to check whether $\mathcal{K} \models q$, it is sufficient to restrict attention to canonical models. This can be proven in the same way as Theorems 3.1 and 3.3. Indeed, an arbitrary counterexample model for entailment can be transformed into a canonical model by unraveling and collapsing/eliminating nodes. Since the query does not contain negative information, there will still be no match for the unraveled model and the query.

Lemma 4.1 *$\mathcal{K} \not\models q$ if and only if there is a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

This result allows us to exploit tree-automata based techniques also for query entailment. Specifically, we build an automaton that accepts exactly the set of canonical models of the knowledge base where there is no match for the query. Once this automaton has been constructed, we can decide the entailment of the query by checking for emptiness the language it accepts.

Roughly, this automaton is obtained by intersecting two automata: one that accepts the canonical models of the knowledge base $\mathcal{K}$ (i.e., $\mathbf{A}_\mathcal{K}$ as constructed in the previous section) and another one that accepts the set of trees, labeled with the same alphabet and that have same branching degree as the models of $\mathcal{K}$, where there is no match for q. The latter will be constructed in this section.

To this aim, we first construct an automaton that accepts the trees where there is a match for q. We will see later how it can be transformed into an automaton that accepts the tree where there is *no match* for q. We consider trees over the alphabet $\Sigma_\mathcal{K}$ extended with additional atomic concepts, one for each variable in q, and we enforce each of these new concepts to be satisfied in a single node of the tree. The intuition behind the use of such trees is that, since the existentially quantified "variables" appear explicitly in the tree, a 2ATA $\mathbf{A}_q$ can easily check the existence of a match for (the interpretation corresponding to) the tree and q. We show now how to construct such a 2ATA.

4.1 Constructing the Automaton

Let $q = \exists\vec{x}.\varphi(\vec{x})$ be a P2RPQ over $\mathcal{K}$, let $atoms(q)$ be the set of atoms appearing in q, and let $\mathcal{X} = \{x_1, \ldots, x_\ell\}$ be the set of variables in $\vec{x}$. We denote by $\mathcal{C}_q$ and $\mathcal{R}_q$ the set of atomic concepts and the set of atomic and inverse of atomic roles that occur in q. Recall that $\mathcal{J}$ denotes the individuals in $\mathcal{K}$. Note that in the following, we will consider both the variables in the set $\mathcal{X}$ and the individuals in $\mathcal{J}$ as atomic concepts, which can be used to build complex roles, like the elements of $\mathcal{C}_q$. Let $U = (\bigcup_{P\in\mathcal{R}_q}(P \cup P^-))^*$. We define for each atom $\alpha \in atoms(q)$ a concept C_α:

$$C_\alpha = \begin{cases} \exists U.(C \sqcap z), & \text{if } \alpha = C(z) \\ \exists U.(z \sqcap \exists R.z'), & \text{if } \alpha = R(z, z') \end{cases}$$

where $z, z' \in \mathcal{J} \cup \mathcal{X}$.

We define the 2ATA $\mathbf{A}_q = (\Sigma_q, S_q, \delta, s_0, F_q)$ as follows.

- $\Sigma_q = \Sigma_\mathcal{K} \cup 2^\mathcal{X}$; i.e., nodes are labeled with elements of $\Sigma_\mathcal{K}$ possibly extended with elements of $\mathcal{X}$. Recall that the individual names in $\mathcal{J}$ were already in $\Sigma_\mathcal{K}$, thus we don't need to extend our alphabet with them.

- S_q is defined similarly as for $\mathbf{A}_\mathcal{K}$, except that we use the closure of the atoms C_α instead of the closure of $C_\mathcal{T}$. More specifically, let $\mathbf{C} = \bigcup_{\alpha\in atoms(q)} Cl^{nnf}(C_\alpha, \mathcal{J} \cup \mathcal{X} \cup \{r, d\})$.

  Recall that we need to consider trees with the same branching degree as those accepted by $\mathbf{A}_\mathcal{K}$, which is given by $|\mathcal{J}|$ at level one and by $k_{C_\mathcal{T}}$ at all other levels.

  The set of states S_q is defined as follows:

$$S_q = \{s_0, s_1\} \cup \mathbf{C} \cup S_\mathcal{X} \cup S_{q,num} \cup S_{q,\mathcal{A}\_role} \cup S_{q,\mathcal{A}\_quant} \cup S_{q,\mathcal{A}\_num}$$

  with:

$$S_\mathcal{X} = \{X_i^+, X_i^- \mid 1 \le i \le |\mathcal{X}|\}$$

$$
\begin{aligned}
S_{q,num} &= \{\langle \geqslant n\, Q.C, i, j\rangle \mid \geqslant n\, Q.C \in \mathbf{C},\ 0 \leq i \leq k_{C_\mathcal{T}}{+}1,\ 0 \leq j \leq n\} \cup \\
&\quad \{\langle \leqslant n\, Q.C, i, j\rangle \mid \leqslant n\, Q.C \in \mathbf{C},\ 0 \leq i \leq k_{C_\mathcal{T}}{+}1,\ 0 \leq j \leq n{+}1\} \\
S_{q,\mathcal{A}\_role} &= \{Qij \mid a_i, a_j \in \mathcal{J} \text{ and } Q \text{ a basic role in } \mathbf{C}\} \\
S_{q,\mathcal{A}\_quant} &= \{\langle j, \exists Q.C\rangle \mid \exists Q.C \in \mathbf{C},\ Q \text{ a basic role}, 1 \leq j \leq |\mathcal{J}|\} \cup \\
&\quad \{\langle j, \forall Q.C\rangle \mid \forall Q.C \in \mathbf{C},\ Q \text{ a basic role}, 1 \leq j \leq |\mathcal{J}|\} \\
S_{q,\mathcal{A}\_num} &= \{\langle m, \geqslant n\, Q.C, i, j\rangle \mid \geqslant n\, Q.C \in \mathbf{C},\ 1 \leq m, i \leq |\mathcal{J}|,\ 0 \leq j \leq n\} \cup \\
&\quad \{\langle m, \leqslant n\, Q.C, i, j\rangle \mid \leqslant n\, Q.C \in \mathbf{C},\ 1 \leq m, i \leq |\mathcal{J}|,\ 0 \leq j \leq n{+}1\}
\end{aligned}
$$

- $F_q = F_\mathbf{C} \cup F_\mathcal{X}$, where
  - $F_\mathbf{C}$ is defined as for $\mathbf{A}_\mathcal{K}$, but using the closure of the concepts C_α, i.e., $F_\mathbf{C} = \{\forall R^*.C \mid \forall R^*.C \in \mathbf{C}\}$;
  - $F_\mathcal{X} = \{X_i^- \mid 1 \leq i \leq |\mathcal{X}|\}$, and ensures that each atomic concept $x \in \mathcal{X}$ actually occurs in the tree, as will be clear from the following.
- The transitions from the initial state are defined for all labels σ containing the symbol r (identifying the root node) as $\delta(s_0, \sigma) = B_\varphi \wedge B_1 \wedge B_2 \wedge B_3 \wedge B_4 \wedge B_\mathcal{X}$, where:
  - B_φ is obtained from $\varphi(\vec{x})$ by replacing each atom α with $(0, C_\alpha)$ (and by considering $\wedge$ and $\vee$ as the analogous connectives in a 2ATA transition);
  - B_1, B_2, B_2, B_3, and B_4 are as for $\mathbf{A}_\mathcal{K}$, and ensure again that the level one nodes properly represent the individuals in $\mathcal{J}$, and that r and each $a_i \in \mathcal{J}$ do not appear at any node below level one in the whole tree. For the latter, an additional state s_1 with an appropriate transition is defined, exactly as for $\mathbf{A}_\mathcal{K}$;
  - Finally, $B_\mathcal{X}$ checks that each atomic concept $x \in \mathcal{X}$ appears exactly once in the tree, i.e.,

$$B_\mathcal{X} = \bigwedge_{1 \leq i \leq |\mathcal{X}|} \Big(\bigvee_{1 \leq j \leq |\mathcal{J}|} ((j, X_i^+) \wedge \bigwedge_{1 \leq j' \leq |\mathcal{J}|,\ j' \neq j} (j', X_i^-)) \Big)$$

    with the following additional transitions defined for each $1 \leq i \leq |\mathcal{X}|$ and each $\sigma \in \Sigma_q$:

$$
\begin{aligned}
\delta(X_i^+, \sigma) &= ((0, x_i) \wedge \textstyle\bigwedge_{1 \leq j \leq k_{C_\mathcal{T}}} (j, X_i^-)) \vee \\
&\quad ((0, \neg x_i) \wedge \textstyle\bigvee_{1 \leq j \leq k_{C_\mathcal{T}}} ((j, X_i^+) \wedge \bigwedge_{1 \leq j' \leq |\mathcal{J}|,\ j' \neq j} (j', X_i^-))) \\
\delta(X_i^-, \sigma) &= (0, \neg x_i) \wedge \textstyle\bigwedge_{1 \leq j \leq k_{C_\mathcal{T}}} (j, X_i^-)
\end{aligned}
$$

When $\mathbf{A}_q$ is in a state C_α in the root node (the only node labeled r), it does not "decompose" C_α as usual. Instead, it checks that the concept C_α is satisfied in a node at level one representing some ABox individual. This is done by the following transitions, for each $\alpha \in atoms(q)$ and σ containing r:

$$\delta(C_\alpha, \sigma) = \textstyle\bigvee_{1 \leq i \leq |\mathcal{J}|} ((i, C_\alpha) \wedge \bigvee_{1 \leq j \leq |\mathcal{J}|} (i, a_j))$$

$\mathbf{A}_q$ contains transitions analogous to those of $\mathbf{A}_\mathcal{K}$ to check that the various concepts C_α are satisfied. These transitions are:

- For each concept or basic role in $\mathbf{C}$, each $\sigma \in \Sigma_q$ and each $i, j \in \mathcal{J}$ there are transitions like the ones given in item 2 for $\mathbf{A}_\mathcal{K}$, which inductively decompose concepts and roles.

- There are transitions like the ones in 3 and 4 for each $\sigma \in \Sigma_q$ and for each concept of the form $\exists Q.C$, $\exists Q.C$ or $\geqslant n\, Q.C$ in $\mathbf{C}$ respectively. The transitions are suitably defined for all states in $S_{q,\mathcal{A}\_quant}$, $S_{q,num}$ and $S_{q,\mathcal{A}\_num}$.
- Atomic transitions similar to those defined in items 5 and 6 are also defined for $\mathbf{A}_q$, but for each $\sigma \in \Sigma_q$, for each $s \in \mathcal{C}_q \cup \mathcal{R}_q \cup \mathcal{J} \cup \{d\} \cup \mathcal{X}$ and for each $Pij \in S_{q,\mathcal{A}\_role}$ with $P \in \mathcal{R}_q$.

The automaton we have defined accepts trees similar to the ones accepted by $\mathbf{A}_\mathcal{K}$, except that these are over the extended alphabet Σ_q, i.e., the node labels may contain query variables. We see such trees as a representation of a canonical quasi-interpretation extended to interpret each query variable as a concept. If in the canonical quasi-interpretation each element of $\mathcal{X} \cup \mathcal{J}$ is interpreted as a singleton set, then we can easily establish a correspondence between the extended quasi-interpretation and a standard quasi-interpretation for $\mathcal{K}$ where there is a *candidate match* for the query. I.e. by mapping each query variable/individual to the single individual in the extension of its interpretation, we obtain a potential match for q.

Definition 4.2 An *extended quasi-interpretation tree* is a labeled tree $\mathbf{T}' = (T, V')$ over the alphabet Σ_q such that the labeled tree $\mathbf{T} = (T', V)$ over $\Sigma_\mathcal{K}$ obtained from $\mathbf{T}'$ by restricting the labeling function V' to $\Sigma_\mathcal{K}$ is a quasi-interpretation tree. A *quasi-interpretation tree with a candidate match* is an extended quasi-interpretation tree $\mathbf{T}' = (T, V')$ that additionally satisfies that there is exactly one $w \in T$ with $x \in V'(w)$ for each $x \in \mathcal{J} \cup \mathcal{X}$. In this case, w is called *the candidate match for x in $\mathbf{T}'$*.
For a quasi-interpretation tree with a candidate match $\mathbf{T}$, the canonical quasi-interpretation $\mathcal{I}_\mathbf{T}$ represented by $\mathbf{T}$ is defined as in Definition 3.5, by considering each $x \in \mathcal{X}$ as a new atomic concept. We denote by $\pi_\mathbf{T}$ the function $\mathcal{J} \cup \mathcal{X} \to \Delta^{\mathcal{I}_\mathbf{T}}$ defined as $\pi_\mathbf{T}(x) = w$, where $x \in \mathcal{J} \cup \mathcal{X}$ and w is the candidate match for x in $\mathbf{T}$.

The following lemma states that the concepts of the form C_α correctly capture the semantics of the query atoms. Indeed, the satisfaction of the concept C_α in a canonical quasi-interpretation ensures the entailment of α.

Lemma 4.3 *Let $\mathbf{T}$ be a quasi-interpretation tree with a candidate match $\mathbf{T}$, and let $\mathcal{I}_\mathbf{T}$ be the canonical quasi-interpretation represented by $\mathbf{T}$. Let $U = (\bigcup_{P \in \mathcal{R}_q}(P \cup P^-))^*$ and C_α as above. Let q be a 2PRPQ and let α be an atom in q. Then the following are equivalent:*

1. $\mathcal{I}_\mathbf{T}, \pi_\mathbf{T} \models \alpha$
2. *there are $w, w' \in \Delta^{\mathcal{I}_\mathbf{T}}$ with $w \in a_i^{\mathcal{I}_\mathbf{T}}$ for some individual a_i such that $(w, w') \in U^{\mathcal{I}_\mathbf{T}}$, and additionally:*
   - *$w' \in z^{\mathcal{I}_\mathbf{T}} \cap C^{\mathcal{I}_\mathbf{T}}$ if α is of the form $C(z)$, and*
   - *$w' \in z^{\mathcal{I}_\mathbf{T}} \cap (\exists R.z')^{\mathcal{I}_\mathbf{T}}$ if α is of the form $R(z, z')$.*
3. *there is some $w \in \Delta^{\mathcal{I}_\mathbf{T}}$ such that $w \in (C_\alpha)^{\mathcal{I}_\mathbf{T}}$ and $w \in a_i^{\mathcal{I}_\mathbf{T}}$ for some individual a_i.*

Proof. We only have to verify that 1 and 2 are equivalent, as the equivalence of 2 and 3 follows trivially from the semantics of C_α. Assume $\mathcal{I}_\mathbf{T}, \pi_\mathbf{T} \models \alpha$. Suppose $\alpha = C(z)$. Then we have that $\pi_\mathbf{T}(z) \in C^{\mathcal{I}_\mathbf{T}}$. Let $w' = \pi_\mathbf{T}(z)$. Since $w' \in z^{\mathcal{I}_\mathbf{T}}$ by construction of $\mathcal{I}_\mathbf{T}$, then $w' \in z^{\mathcal{I}_\mathbf{T}} \cap C^{\mathcal{I}_\mathbf{T}}$ holds. Analogously, if $\alpha = R(z, z')$ we have $(\pi_\mathbf{T}(z), \pi_\mathbf{T}(z')) \in R^{\mathcal{I}_\mathbf{T}}$. As $\pi_\mathbf{T}(z') \in (z')^{\mathcal{I}_\mathbf{T}}$, it follows that $\pi_\mathbf{T}(z) \in (\exists R.z')^{\mathcal{I}_\mathbf{T}}$. By setting $w' = \pi_\mathbf{T}(z)$ we get $w' \in z^{\mathcal{I}_\mathbf{T}} \cap (\exists R.z')^{\mathcal{I}_\mathbf{T}}$ as desired. In both cases, it is only left to prove the

existence of a suitable w. By construction of $\mathcal{I}_{\mathbf{T}}$, $w' \in \mathbf{I}$ or $w' \in \mathbf{D}_j$ for some $j \in \mathbf{I}$. In the former case, we can take $w = w'$. In the latter, if $w' \in \mathbf{D}_j$, then $w = j$. In both cases, $w \in a_i^{\mathcal{I}_{\mathbf{T}}}$ holds for some a_i. Finally, $(w, w') \in U^{\mathcal{I}_{\mathbf{T}}}$ clearly follows from the definition of U and of $\mathcal{I}_{\mathbf{T}}$. The proof of the other direction is also straightforward. If $w \in a_i^{\mathcal{I}_{\mathbf{T}}}$ for some a_i, then $w \in \mathbf{I}$. If $(w, w') \in U^{\mathcal{I}_{\mathbf{T}}}$ then either $w = w' \in \mathbf{I}$ or $w' \in \mathbf{D}_j$ for some j. In both cases, $w, w' \in \Delta^{\mathcal{I}_{\mathbf{T}}}$. If $w' \in z^{\mathcal{I}_{\mathbf{T}}} \cap C^{\mathcal{I}_{\mathbf{T}}}$, then $\pi_{\mathbf{T}}(z) = w'$ and $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models C(z)$. If $w' \in z^{\mathcal{I}_{\mathbf{T}}} \cap (\exists R.z')^{\mathcal{I}_{\mathbf{T}}}$, then $\pi_{\mathbf{T}}(z) = w'$ and $(\pi_{\mathbf{T}}(z), \pi_{\mathbf{T}}(z')) \in R^{\mathcal{I}_{\mathbf{T}}}$ follow, so $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models R(z, z')$ as desired. □

The following lemma can be proved in exactly the same way as Lemma 3.8. It will be useful for proving that the automaton $\mathbf{A}_q$ accepts exactly the set of quasi-interpretation trees where there is a match for q.

Lemma 4.4 *Let $\mathbf{T}$ be a tree accepted by $\mathbf{A}_q$, let (T, r) be an accepting run of $\mathbf{A}_q$ over $\mathbf{T}$ and let y be a node of T with $r(y) = (x, E)$, for some node x of $\mathbf{T}$, $x \neq \varepsilon$, and some $E \in \mathbf{C}$ a concept, a basic role, an individual in $\mathcal{J}$ or a variable in $\mathcal{X}$. Then $r(y) = (x, E)$ evaluates to true in the run iff*

- *$x \in E^{\mathcal{I}_{\mathbf{T}}}$ when E is a concept, an individual or a variable;*
- *$\langle x', x\rangle \in E^{\mathcal{I}_{\mathbf{T}}}$ when E is a role, x' is the predecessor of x in $\mathbf{T}$ and $x' \neq \varepsilon$.*

Proposition 4.5 *Let $\mathbf{T}$ be a quasi-interpretation tree. If $\mathbf{A}_q$ accepts $\mathbf{T}$, then $\mathbf{T}$ has a candidate match and $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$.*

Proof. Consider any extended quasi-interpretation tree $\mathbf{T}$ accepted by $\mathbf{A}_q$ and let (T_r, r) be a successful run of $\mathbf{A}_q$ on $\mathbf{T}$. $\mathcal{I}_{\mathbf{T}}$ is an extended quasi-interpretation tree by definition. Moreover, $r(\varepsilon) = (0, F_m \wedge F_v)$ must evaluate to true (T_r, r), and thus F_v will ensure that $\mathbf{T}$ is a quasi-interpretation tree with a candidate match. It is only left to verify that $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$. The satisfaction of F_m ensures the existence of suitable query atoms $\alpha_1, \ldots, \alpha_m$ such that (*i*) if $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models \alpha_j$ for every j, then $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$; and (*ii*) for each α_j there are nodes y, y' in T_r and some level one node w in $\mathbf{T}$ such that $r(y) = (w, C_{\alpha_j})$ and $r(y') = (w, a_i)$ evaluate to true true in the run. From (*ii*) and Lemma 4.4, it follows that for each such C_{α_j} there is some $w \in \Delta^{\mathcal{I}_{\mathbf{T}}}$ such that $w \in (C_{\alpha_j})^{\mathcal{I}_{\mathbf{T}}}$ and $w \in a_i^{\mathcal{I}_{\mathbf{T}}}$ for some individual a_i. Thus $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models \alpha_j$ by lemma 4.3, and $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$ by (*i*) above. □

Proposition 4.6 *Let $\mathbf{T}$ be a quasi-interpretation tree with a candidate match such that $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$. Then $\mathbf{A}_q$ accepts $\mathbf{T}$.*

Proof. We have to verify that $\mathbf{A}_{\mathcal{K}}$ has a accepting run (T_r, r) on $\mathbf{T}$. Due to the construction of $\mathbf{A}_q$, this can be done in a straightforward way. The accepting run will start at (ε, s_0). Since the root ε of $\mathbf{T}$ is marked with r, then a transition of the form $\delta(s_0, V(\varepsilon)) = F_m \wedge F_v$ will be defined. Since $\mathbf{T}$ is a quasi-interpretation tree with a candidate match, there will be exactly one $w \in T$ with $x \in V'(w)$ for each $x \in \mathcal{J} \cup \mathcal{X}$. Moreover, for each $x \in \mathcal{J}$, w will be level one node. This ensures the satisfaction of F_v.

As for F_m, since $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models q$, there must be atoms $\alpha_1, \ldots, \alpha_m$ such that $\mathcal{I}_{\mathbf{T}}, \pi_{\mathbf{T}} \models \alpha_j$ for every $1 \leq j \leq m$, which imply the entailment of q and thus the satisfaction of F_m. By Lemma4.3, for each such α_j, there is some $w \in \Delta^{\mathcal{I}_{\mathbf{T}}}$ such that $w \in (C_\alpha)^{\mathcal{I}_{\mathbf{T}}}$ and $w \in a_i^{\mathcal{I}_{\mathbf{T}}}$ for some individual a_i, Thus, there will be some node x of $\mathbf{T}$ such that the run of the automaton continues by moving nodes y, y' of T_r with $r(y) = (x, a_i)$ and $r(y') = (x, C_\alpha)$. By Lemma 3.8, we know that they evaluate to true and the run concludes successfully, thus $\mathbf{A}_q$ accepts $\mathbf{T}$. □

4.2 Deciding Query Entailment

Let $\mathcal{K}$ be an $\mathcal{ALCQI}b_{reg}$ KB and q a P2RPQ over $\mathcal{K}$. Roughly, the procedure to decide query entailment works as follows. The automaton $\mathbf{A}_q$ accepts a tree over the extended alphabet Σ_q if it represents a canonical quasi-interpretation for $\mathcal{K}$ where there is a match for q. We project out the query variables from the automaton $\mathbf{A}_q$ to obtain an automaton that accepts the same trees, but restricted to the alphabet $\Sigma_\mathcal{K}$. The new automaton accepts exactly the set of canonical quasi-interpretations for $\mathcal{K}$ where there is a match for q, no matter where the match is. The next step is to complement the automaton, so that it accepts exactly the canonical quasi-interpretations where there is no match for q. Finally, we intersect this automaton with the automaton $\mathbf{A}_\mathcal{K}$ to obtain an automaton $\mathbf{A}_{\mathcal{K}\not\models q}$ which accepts the trees that represent a canonical model of $\mathcal{K}$ where there is no match for q. We know, by Lemma 4.1, that $\mathcal{K} \not\models q$ iff there is some such model. So we have that $\mathcal{K} \not\models q$ iff the language accepted by $\mathbf{A}_{\mathcal{K}\not\models q}$ is not empty, and query entailment can be decided by testing the automaton $\mathbf{A}_{\mathcal{K}\not\models q}$ for emptiness. Now we will discuss these steps in detail, and analyze the computational complexity of each of them. We will denote by $||\mathcal{K}||$ and $||q||$ the sizes (of the strings representing) $\mathcal{K}$ and q respectively.

1. First consider the automaton $\mathbf{A}_q$. Note that both $|\mathbf{C}| = |\bigcup_{\alpha\in atoms(q)} Cl^{nnf}(C_\alpha, \mathcal{J}, d)|$ and $|\mathcal{X}|$ are linear in $||q||$. Since $\Sigma_q = \Sigma_\mathcal{K} \cup 2^\mathcal{X}$, we have that $|\Sigma_q| = 2^{O(M+|\mathcal{X}|)}$ where $O(M) = O((|\mathcal{C}_\mathcal{K}| + |\mathcal{R}_\mathcal{K}| + |\mathcal{J}|)^2)$, as in Section 3.4. Thus $|\Sigma_q|$ is single exponential in $||q|| + ||\mathcal{K}||$. Analogously as for $S_\mathcal{K}$, we have that the cardinality S_q is bounded by $O(N^2)$, for $N = |\mathbf{C}| + |\mathcal{J}| + \mathbf{n}$ (recall that $\mathbf{n}$ represents the maximal number occurring in the number restrictions of $\mathcal{K}$), so $|S_q|$ is polynomial in $||\mathcal{K}|| + ||q||$. The parity acceptance condition of $\mathbf{A}_q$ is of constant length.

2. We convert $\mathbf{A}_q$ into an equivalent 1NTA $\mathbf{A}^1_q$. By Theorem 2.3, the number of states of $\mathbf{A}^1_q$ is $2^{O(N^2)}$, so it is single exponential in $||\mathcal{K}|| + ||q||$. The parity acceptance condition of $\mathbf{A}^1_q$ remains constant.

3. We project out variables from $\mathbf{A}^1_q$ obtaining a 1NTA $\mathbf{A}^2_q$. Projecting out symbols on one-way automata is trivial. We want to simply remove from the input trees the symbols that represent the query variables in $\mathcal{X}$, so that we obtain an automaton that runs over the alphabet $\Sigma_\mathcal{K}$ and accepts a tree iff it is the restriction to this alphabet of a tree accepted by $\mathbf{A}^1_q$. Recall that, since $\mathbf{A}^1_q$ is a 1NTA, the transitions are of the form $\delta(q,\sigma) = t$ where t is a disjunction of conjuncts of the form $(1, q_1)\wedge\ldots\wedge(k_{C_\mathcal{T}}, q_{k_{C_\mathcal{T}}})$, with $\sigma \in \Sigma_q$ and $q, q_1, \ldots, q_{k_{C_\mathcal{T}}}$ states of $\mathbf{A}^1_q$.

   We project out the variables in $\mathcal{X}$ from $\mathbf{A}^1_q$ as follows. For each variable $x \in \mathcal{X}$, we consider each $\sigma \in \Sigma_q$ and each state q of $\mathbf{A}^1_q$. There are two possibilities:

   - If $x \notin \sigma$ and there is a transition $\delta(q,\sigma) = t$, this transition remains unchanged.
   - If $x \in \sigma$ and there is a transition $\delta(q,\sigma) = t$, then let $\sigma' = \sigma \setminus \{x\}$. If there is some transition $\delta(q,\sigma') = t'$ in $\mathbf{A}^1_q$, then we replace it by the new transition $\delta(q,\sigma') = t' \vee t$. Otherwise, if no transition of the form $\delta(q,\sigma')$ exists, we add a new transition $\delta(q,\sigma') = t$. In both cases, the transition $\delta(q,\sigma) = t$ is removed.

   After this steps have been followed for every $x \in \mathcal{X}$, we have a new automaton $\mathbf{A}^2_q$ such that:
   (*i*) the alphabet of $\mathbf{A}^2_q$ is $\Sigma_\mathcal{K}$, and thus not larger than the alphabet Σ_q of $\mathbf{A}^1_q$;
   (*ii*) $\mathbf{A}^2_q$ has (at most) as many states as $\mathbf{A}^1_q$ and an accepting condition of (at most) the same length;
   (*iii*) $\mathbf{A}^2_q$ accepts a tree $\mathbf{T}$ iff $\mathbf{T}$ is the restriction to $\Sigma_\mathcal{K}$ of a tree $\mathbf{T}'$ accepted by $\mathbf{A}^1_q$.

Concerning (*ii*), note that the projection of variables as defined above does not modify the state set of the automaton. It can be the case that, when projecting away the variables, all the occurrences of some state q of $\mathbf{A}_q^1$ are eliminated from the transition function. If this happens then the state q can be safely eliminated from the state set and the acceptance condition of $\mathbf{A}_q^2$.

Importantly, we know that an extended quasi-interpretation tree $\mathbf{T}'$ is accepted by $\mathbf{A}_q$ iff $\mathcal{I}_{\mathbf{T}'}, \pi_{\mathbf{T}'} \models q$. Since $\mathbf{A}_q$ and $\mathbf{A}_q^1$ accept exactly the same trees, this also holds for $\mathbf{A}_q^1$. I.e. $\mathbf{A}_q^1$ accepts $\mathbf{T}'$ iff there is a match for q in $\mathcal{I}_{\mathbf{T}'}$, where the match for each query variable x is given by the only element $\pi_{\mathbf{T}'}(x)$ of $\mathcal{I}_{\mathbf{T}'}$ such that $\pi_{\mathbf{T}'}(x) \in x^{\mathcal{I}_{\mathbf{T}'}}$. Let $\mathbf{T}$ be the restriction of $\mathbf{T}'$ to $\Sigma_{\mathcal{K}}$. By construction, we know that $\mathbf{A}_q^2$ accepts $\mathbf{T}$ iff $\mathbf{A}_q^1$ accepts $\mathbf{T}'$, thus $\mathbf{A}_q^2$ accepts $\mathbf{T}$ iff $\mathcal{I}_{\mathbf{T}'}, \pi_{\mathbf{T}'} \models q$. Clearly, a match for q in an extended quasi-interpretation $\mathcal{I}'$ is also a match for q in the quasi-interpretation $\mathcal{I}$ obtained by restricting it to the concepts in $\mathcal{K}$ (which are the only concepts that may occur in the query) and mapping each variable x to the only individual o in $\mathcal{I}$ with $o \in x^{\mathcal{I}}$. Moreover, any match π for q in a quasi-interpretation $\mathcal{I}$ is also a match in the extended quasi-interpretation $\mathcal{I}'$ obtained by adding a new concept x for each query variable and setting $x^{\mathcal{I}'} = \pi(x)$. Therefore, we have that a tree $\mathbf{T}$ is accepted by $\mathbf{A}_q^2$ iff there is a match for $\mathcal{I}_{\mathbf{T}}$ and q.

4. We complement $\mathbf{A}_q^2$, obtaining a 1NTA $\mathbf{A}_{\neg q}$. Let n be the number of states and $m-1$ the length of the parity condition of $\mathbf{A}_q^2$. By [20], $\mathbf{A}_q^2$ can be transformed into an equivalent one $\mathbf{A}_q^{2'}$ with a coparity condition of length m. Following [20], we can construct an automaton $\mathbf{A}'_{\neg q}$ that accepts the complement language of that of $\mathbf{A}_q^{2'}$, i.e., $\mathbf{A}'_{\neg q}$ accepts a tree $\mathbf{T}$ over $\Sigma_{\mathcal{K}}$ iff $\mathbf{A}_q^{2'}$ does not accept $\mathbf{T}$. The number of states of $\mathbf{A}'_{\neg q}$ is no bigger than $2^{O(mn \log n)}$, so it is single exponential in m and in n. Recall that n is bounded by $2^{O(N^2)}$ and is N linear in $||\mathcal{K}|| + ||q||$. Since m is bounded by a constant, the number of states of $\mathbf{A}'_{\neg q}$ is double exponential in $||\mathcal{K}|| + ||q||$. The automaton $\mathbf{A}'_{\neg q}$ has a coparity acceptance condition whose length is bounded by $O(mn \log n)$, so it is single exponential in $||\mathcal{K}|| + ||q||$.[4] Finally, we convert the coparity condition of $\mathbf{A}'_{\neg q}$ into a parity one again, to obtain $\mathbf{A}_{\neg q}$ (the length of the parity condition remains single exponential). Since $\mathbf{A}_q^2$ accepts the trees in which there is a match for q, we have that a tree $\mathbf{T}$ is accepted by $\mathbf{A}_{\neg q}$ iff there is no match for $\mathcal{I}_{\mathbf{T}}$ and q.

5. We convert $\mathbf{A}_{\mathcal{K}}$ to a 1NTA $\mathbf{A}_{\mathcal{K}}^1$. By Theorem 2.3, the number of states of $\mathbf{A}_{\mathcal{K}}^1$ is $2^{O(n)}$ and the length of its parity condition is $O(m)$, where n and m are the number of states and the length of the parity condition of $\mathbf{A}_{\mathcal{K}}$ respectively. From Section 3.4, we know that n is bounded by $O(L^2)$, where L is linear in $||\mathcal{K}||$, while m is a constant. So we have that the number of states of $\mathbf{A}_{\mathcal{K}}^1$ is single exponential in $||\mathcal{K}||$ and it has a parity acceptance condition of constant length.

6. Finally, we construct a 1NTA $\mathbf{A}_{\mathcal{K} \not\models q}$ that accepts the intersection of the languages accepted by $\mathbf{A}_{\mathcal{K}}^1$ and $\mathbf{A}_{\neg q}$, i.e., that accepts exactly the set of trees that represent a canonical model of $\mathcal{K}$ in which there is no match for q. Since a tree $\mathbf{T}$ is accepted by $\mathbf{A}_{\mathcal{K}}$ iff $\mathcal{I}_{\mathbf{T}}$ is a canonical model of $\mathcal{K}$, while it is accepted by $\mathbf{A}_{\neg q}$ iff there is no match for $\mathcal{I}_{\mathbf{T}}$ and q, every tree accepted by $\mathbf{A}_{\mathcal{K} \not\models q}$ represents a counterexample to $\mathcal{K} \models q$. On the other hand, if a tree $\mathbf{T}$ is not accepted by $\mathbf{A}_{\mathcal{K} \not\models q}$, then either it is not accepted by $\mathbf{A}_{\mathcal{K}}$, in which case $\mathcal{I}_{\mathbf{T}}$ is not a model of $\mathcal{K}$, or it is not accepted by $\mathbf{A}_{\neg q}$, in which case it is accepted by $\mathbf{A}_q^2$, i.e., there is a match for $\mathcal{I}_{\mathbf{T}}$ and q. Hence the tree does not represent a counterexample to $\mathcal{K} \models q$.

[4] Similar bounds can be obtained using the construction in [18], defined for the more general *Streett* automata whose acceptance condition is a relaxation of the coparity one.

Let n_1 and n_2, m_1 and m_2 denote the number of states of $\mathbf{A}^1_{\mathcal{K}}$ and of $\mathbf{A}_{\neg q}$, and the length of the parity conditions of $\mathbf{A}^1_{\mathcal{K}}$ and $\mathbf{A}_{\neg q}$ respectively. By the results in [8], the automaton $\mathbf{A}_{\mathcal{K}\not\models q}$ has $2^{m_1 m_2} n_1 n_2$ states, and a parity condition of length $O(m_1 + m_2)$. For N and L as above (items 1 and 5), we have the following bounds : n_1 is single exponential in L, n_2 is double exponential in N, m_2 is single exponential in N and m_1 a constant. Thus the size of $\mathbf{A}_{\mathcal{K}\not\models q}$ is is double exponential in N and is single exponential in L, and the length of its parity condition is single exponential in N. Since both N and L are linear in $||\mathcal{K}|| + ||q||$, $\mathbf{A}_{\mathcal{K}\not\models q}$ has double exponential size and single exponential parity condition in $||\mathcal{K}|| + ||q||$.

Let $\mathbf{T}'$ be an extended quasi-interpretation tree and let $\mathbf{T}$ be the quasi-interpretation obtained by restricting $\mathbf{T}'$ to $\Sigma_{\mathcal{K}}$. Summing up, we have the following:

- $\mathbf{A}_q$ accepts $\mathbf{T}'$ iff $\mathcal{I}_{\mathbf{T}'}, \pi_{\mathbf{T}'} \models q$.
- $\mathbf{A}^1_q$ accepts $\mathbf{T}'$ iff $\mathcal{I}_{\mathbf{T}'}, \pi_{\mathbf{T}'} \models q$.
- $\mathbf{A}^2_q$ accepts $\mathbf{T}$ iff there is a match for $\mathcal{I}_{\mathbf{T}}$ and q.
- $\mathbf{A}_{\neg q}$ accepts $\mathbf{T}$ iff there is no match for $\mathcal{I}_{\mathbf{T}}$ and q.
- $\mathbf{A}^1_{\mathcal{K}}$ accepts $\mathbf{T}$ iff $\mathbf{T}$ is a canonical model of $\mathcal{K}$.
- $\mathbf{A}_{\mathcal{K}\not\models q}$ accepts $\mathbf{T}$ iff $\mathbf{T}$ is a canonical model of $\mathcal{K}$ and there is no match for $\mathcal{I}_{\mathbf{T}}$ and q.

As a consequence, we get:

Lemma 4.7 *There exists a canonical counterexample to $\mathcal{K} \models q$ iff the set of trees accepted by $\mathbf{A}_{\mathcal{K}\not\models q}$ is not empty.*

By Lemma 4.1, we get the following result.

Theorem 4.8 *For every $\mathcal{ALCQI}b_{reg}$knowledge base $\mathcal{K}$ and P2RPQ query q, we have that $\mathcal{K} \models q$ iff the set of trees accepted by $\mathbf{A}_{\mathcal{K}\not\models q}$ is not empty.*

Moreover, by Theorem 2.3, since the number of states of $\mathbf{A}_{\mathcal{K}\not\models q}$ is double exponential bound and the length of its parity condition linear in the sum of the sizes of $\mathcal{K}$ and q, we get:

Theorem 4.9 *$\mathcal{K} \models q$ is decidable in double exponential time in the size of q and the number of atomic concepts, roles, and individuals in $\mathcal{K}$.*

4.3 Data complexity

A brief remark on the contribution of $\mathcal{K}$ to overall complexity of the algorithm is in place. Most of the existing query answering algorithms in expressive DLs are double exponential in $||\mathcal{K}|| + ||q||$, but just single exponential in $||\mathcal{K}||$. This is not the case of our algorithm, however. The definition of P2RPQs allows for arbitrary concepts in query atoms. It is a common practice to restrict the queries and allow only atomic concepts instead. If we impose this restriction (even if we allow for arbitrary roles) then the automaton $\mathbf{A}_q$ does not have to deal with number restrictions, since they do not occur in the concepts C_α representing the query atoms. As a result its set of states would be simpler ($S_q = \{s_0\} \cup \mathbf{C} \cup S_{q,\mathcal{A}\_role} \cup S_{q,\mathcal{A}\_quant}$)

and N would depend only on $||q||$ and $|\mathcal{J}|$. This means that if $\mathcal{J}$ is bounded, then the size and length of parity condition of $\mathbf{A}_{\mathcal{K}\not\models q}$ are single exponential and constant in $||\mathcal{K}||$ respectively, resulting in a decision procedure which is single exponential in $||\mathcal{K}||$. In the general case the number of states and size of parity condition of $\mathbf{A}_q$ are already double and single exponential in $|\mathcal{J}|$. This implies that our decision procedure requires time double exponential in the size of the size of the ABox, i.e., it has doubly exponential data complexity. Although the data complexity of query answering in $\mathcal{ALCQIb}_{reg}$has not been studied so far, we conjecture that this is not optimal. In fact, for the related description logic $\mathcal{SHIQ}$, a much lower coNP lower bound for data complexity is known [11]. It is not clear whether the automaton $\mathbf{A}_q$ could be designed in such a way that it size would not depend on $||\mathcal{K}||$ at all in order to obtain an exponentially better data complexity upper bound.

5 Query entailment with complex role inclusion axioms

The description logic $\mathcal{RIQ}$, proposed in [15], extends the well known $\mathcal{SHIQ}$ with *role inclusion axioms* of the form $R{\cdot}S \sqsubseteq T$, where R, S and T are roles. A $\mathcal{RIQ}$ knowledge base, like an $\mathcal{ALCQIb}_{reg}$one, comprises a TBox and and ABox, but additionally it has an *RBox*, which is a set of role inclusion axioms. The TBox and ABox are defined as in $\mathcal{ALCQIb}_{reg}$, except for the fact that only atomic or inverse of atomic roles are allowed (i.e., the only rule defining roles is $R \longrightarrow P \mid P^-$). In the rule defining concepts, Q stands for a *simple role*, and only such roles are allowed to occur in the number restrictions. The definition of *simple roles* for $\mathcal{RIQ}$ is slightly more involved than for $\mathcal{ALCQIb}_{reg}$, we refer to [15] for details. In order to preserve decidability of reasoning, every $\mathcal{RIQ}$ RBox must satisfy a condition called *regularity*, which avoids cyclic dependencies between roles. Very roughly, if a role hierarchy $\mathcal{R}$ is regular, then for every role R there is a regular expression ρ_R such that, for every chain of roles $S_0{\cdot}\ldots{\cdot}S_N$, it holds that $S_0{\cdot}\ldots{\cdot}S_N$ is in the language denoted by ρ_R iff $(S_0{\cdot}\ldots{\cdot}S_N)^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ in very model $\mathcal{I}$ of $\mathcal{R}$. Since all implications between chains of roles can be captured by a set of regular expressions, it is not surprising that query entailment in $\mathcal{RIQ}$ can be reduced to query entailment in $\mathcal{ALCQIb}_{reg}$.

In this section we show that the algorithm we have presented is also a decision procedure for answering P2RPQs in $\mathcal{RIQ}$, and thus in $\mathcal{SHIQ}$. Following [15], for a regular role hierarchy $\mathcal{R}$ and a (possible inverse) role R occurring in $\mathcal{R}$, ρ_R represents the regular expression denoting exactly the chains of roles which imply R, as discussed above. We also denote by $\mathsf{exp}(\mathcal{R})$ the set of expressions of the form $\rho_R \sqsubseteq R$ for each R occurring in $\mathcal{R}$. The following result will be useful:

Lemma 5.1 *[Lemma 1 in [15]] An interpretation $\mathcal{I}$ is a model of a regular RBox $\mathcal{R}$ iff $\mathcal{I}$ is a model of* $\mathsf{exp}(\mathcal{R})$.

Since R itself is in the language of ρ_R, we easily obtain the following corollary:

Corollary 5.2 *Let $\mathcal{I}$ be a model of a regular RBox $\mathcal{R}$. Then $R^{\mathcal{I}} = (\rho_R)^{\mathcal{I}}$. Furthermore, let C be a $\mathcal{RIQ}$ concept, and let C' be the $\mathcal{ALCQIb}_{reg}$concept obtained by replacing every occurrence in C of each role R by the regular expression ρ_R. Then $C^{\mathcal{I}} = (C')^{\mathcal{I}}$.*

This allows for a natural reduction of a $\mathcal{RIQ}$ TBox into an $\mathcal{ALCQIb}_{reg}$one. Since only atomic concepts and roles are allowed in ABoxes, we can not use this simple procedure to translate $\mathcal{A}$. Instead, we have to explicitly add to the ABox all relations between individuals that are implied by the RBox.

Definition 5.3 Let $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ be a $\mathcal{RIQ}$ knowledge base such that $\mathcal{R}$ is regular. We denote by $\mathcal{K}' = \langle \mathcal{A}', \mathcal{T}' \rangle$ the $\mathcal{ALCQI}b_{reg}$KB obtained from $\mathcal{K}$ as follows:

1. The ABox $\mathcal{A}'$ is the smallest set of assertions closed under the following rules:

   (i) $\mathcal{A} \subseteq \mathcal{A}'$;

   (ii) if $R_1(a,b) \in \mathcal{A}'$ and $R_1 \sqsubseteq R_2 \in \mathcal{R}$, then $R_2(a,b) \in \mathcal{A}'$; and

   (iii) if $R_1(a,b) \in \mathcal{A}'$, $R_2(b,c) \in \mathcal{A}'$ and $R_1 \cdot R_2 \sqsubseteq R_3 \in \mathcal{R}$, then $R_3(a,c) \in \mathcal{A}'$;

   where a, b, c are any individuals from $\mathcal{A}$ and R_1, R_2, R_3 are any roles occurring in $\mathcal{R}$.

2. The TBox $\mathcal{T}'$ is obtained from T by replacing every concept inclusion axiom $C \sqsubseteq D$ by $C' \sqsubseteq D'$, where C' and D' are the concepts obtained by replacing every occurrence of each role R by the regular expression ρ_R in C and D respectively.

Lemma 5.4 *If $\mathcal{I} \models \mathcal{K}$ then $\mathcal{I} \models \mathcal{K}'$.*

Proof. Assume that $\mathcal{I} \models \mathcal{K}$. Clearly $\mathcal{I}$ satisfies all the assertions in $\mathcal{A}$. The satisfaction of the assertions added to $\mathcal{A}'$ when closing the ABox as defined above is directly implied by the fact that $\mathcal{I} \models \mathcal{R}$. So we have that $\mathcal{I} \models \mathcal{A}'$. Since $\mathcal{I} \models \mathcal{R}$, by Corollary 5.2, we have $C^{\mathcal{I}} = (C')^{\mathcal{I}}$ and $D^{\mathcal{I}} = (D')^{\mathcal{I}}$. As $\mathcal{I} \models \mathcal{T}$, then $\mathcal{I}$ satisfies every $C' \sqsubseteq D'$ in $\mathcal{T}'$ and thus $\mathcal{I} \models \mathcal{T}'$. □

The converse of Lemma 5.4 holds in a slightly weaker version. Indeed, not every model of $\mathcal{K}'$ is necessarily a model of $\mathcal{K}$, since the models of $\mathcal{K}'$ need not be closed under the RBox. However, the models of $\mathcal{K}'$ and $\mathcal{K}$ may only differ in the interpretation of some 'implied' roles, and if we add the missing implied roles to a model of $\mathcal{K}'$, we obtain a model of $\mathcal{K}$.

Definition 5.5 Let $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ be a $\mathcal{RIQ}$ knowledge base such that $\mathcal{R}$ is regular. Let $\mathcal{K}'$ be an $\mathcal{ALCQI}b_{reg}$KB obtained from $\mathcal{K}$ as in Definition 5.3 and let $\mathcal{I}$ be a model of $\mathcal{K}'$. The *extension of $\mathcal{I}$ to $\mathcal{R}$*, denoted $\mathcal{I}^{\mathcal{R}}$, is defined as follows:

- $\Delta^{\mathcal{I}^{\mathcal{R}}} = \Delta^{\mathcal{I}}$,
- for every atomic concept A, $(A)^{\mathcal{I}^{\mathcal{R}}} = (A)^{\mathcal{I}}$,
- for every atomic role P occurring in $\mathcal{R}$, if $(x,y) \in (\rho_P)^{\mathcal{I}}$ or $(y,x) \in (\rho_{P^-})^{\mathcal{I}}$ then $(x,y) \in (P)^{\mathcal{I}^{\mathcal{R}}}$.

Lemma 5.6 *If $\mathcal{I} \models \mathcal{K}'$, then $\mathcal{I}^{\mathcal{R}} \models \mathcal{K}$.*

Proof. If $\mathcal{I} \models \mathcal{K}'$, then $\mathcal{I} \models \mathcal{A}'$, and since $\mathcal{A} \subseteq \mathcal{A}'$, then $\mathcal{I} \models \mathcal{A}$. Every ABox assertion that is true in $\mathcal{I}$ is also true in $\mathcal{I}^{\mathcal{R}}$, since the latter only extends the former by possibly adding tuples to the interpretation of roles. Therefore, $\mathcal{I}^{\mathcal{R}} \models \mathcal{A}$. By the last item in Definition 5.5, $\mathcal{I}^{\mathcal{R}} \models \mathsf{exp}(\mathcal{R})$, and thus by Lemma 5.1, $\mathcal{I}^{\mathcal{R}} \models \mathcal{R}$. By the second item in the definition, and since $(\rho_R)^{\mathcal{I}} = (R)^{\mathcal{I}^{\mathcal{R}}}$ holds for every role R, it is easy to verify that $(C')^{\mathcal{I}} = (C)^{\mathcal{I}^{\mathcal{R}}}$ for every concept C occurring in $\mathcal{T}$. Thus $\mathcal{I} \models \mathcal{T}'$ implies that $\mathcal{I}^{\mathcal{R}} \models \mathcal{T}$ □

In virtue of Lemmas 5.4 and 5.6, we can reduce the task of checking satisfiability of a $\mathcal{RIQ}$ knowledge base $\mathcal{K}$, to checking satisfiability of the $\mathcal{ALCQI}b_{reg}$one $\mathcal{K}'$, and therefore we can decide it using the automata algorithm described in Section 3. We also provide a method to decide entailment of 2PRPQs. To this aim, we will also rewrite a query q over $\mathcal{K}$ into a query q' over $\mathcal{K}'$, in such a way that query entailment is preserved.

Definition 5.7 Let q be a P2RPQ over a $\mathcal{RIQ}$ knowledge base $\mathcal{K}$ such that the RBox is regular. We denote by q' the P2RPQ over K' that results from substituting every occurrence of each role R by ρ_R in q.

Note that the resulting query may contain regular expression even when the original one does not, i.e., our technique reduces positive (resp. conjunctive) queries over $\mathcal{RIQ}$ to positive (resp. conjunctive) regular path queries over $\mathcal{ALCQIb}_{reg}$.

Proposition 5.8 *Let $\mathcal{K}$ be a $\mathcal{RIQ}$ knowledge base with a regular RBox and let q a 2PRPQ over $\mathcal{K}$. Let $\mathcal{K}'$ be the $\mathcal{ALCQIb}_{reg}$KB obtained from $\mathcal{K}$ as described in Definition 5.3 and let q' be the query over K' as in Definition 5.7. Then $\mathcal{K} \models q$ iff $\mathcal{K}' \models q'$.*

Proof. First we will prove the following:
Claim A: If $\mathcal{I} \models q'$ and $\mathcal{I} \models \mathcal{R}$, then $\mathcal{I} \models q$.
Let α be any atom of q'. If α is a concept atom of the form $C'(x)$, then the corresponding atom in q will be $C(x)$. Since $\mathcal{I} \models \mathcal{R}$, by Corollary 5.2, we have $C^{\mathcal{I}} = (C')^{\mathcal{I}}$. Analogously, if α is a role atom $\rho_R(x, y)$, obtained by substituting R by ρ_R, then there is a corresponding atom $R(x, y)$ in q, and we know that $R^{\mathcal{I}} = (\rho_R)^{\mathcal{I}}$. Thus, any match π for q' in $\mathcal{I}$ is also a match for q.
Claim B: If $\mathcal{I}^{\mathcal{R}} \models q$, then $\mathcal{I} \models q'$.
The proof is similar. Take any atom α of q. If α is a concept atom of the form $C(x)$, then the corresponding atom in q will be $C'(x)$. By definition of $\mathcal{I}^{\mathcal{R}}$m we know that $(C')^{\mathcal{I}} = (C)^{\mathcal{I}^{\mathcal{R}}}$ holds. If α is a role atom $R(x, y)$, then there is a corresponding atom $\rho_R(x, y)$ in q. Again, $(\rho_R)^{\mathcal{I}} = (R)^{\mathcal{I}^{\mathcal{R}}}$ holds for every role R. Thus, any match π for q in $\mathcal{I}^{\mathcal{R}}$ is also a match for q in $\mathcal{I}$.

The proof of the proposition is now trivial. Assume $\mathcal{K} \models q$. Take any model $\mathcal{I}$ of $\mathcal{K}'$. Then $\mathcal{I}^{\mathcal{R}} \models \mathcal{K}$ by Lemma 5.6, thus $\mathcal{I}^{\mathcal{R}} \models q$ and by Claim B above, then $\mathcal{I} \models q'$. For the other direction, assume $\mathcal{K}' \models q'$ and consider an arbitrary $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$. By Lemma 5.4 we know that $\mathcal{I} \models \mathcal{K}'$. Therefore $\mathcal{I} \models q'$, and since $\mathcal{I} \models \mathcal{R}$, then $\mathcal{I} \models q$. □

As a result, the decision procedure given here can be used to answer 2PRPQs over $\mathcal{RIQ}$ knowledge bases. Concerning the complexity of the algorithm, all steps are clearly polynomial in the size of $\mathcal{A}$, $\mathcal{T}$ and $\mathsf{exp}(\mathcal{R})$. However, the size of $\mathsf{exp}(\mathcal{R})$ can be exponential in the size of $\mathcal{R}$. It is also well known that for an specific kind of hierarchies, called *simple role hierarchies* in [15], this blowup can be avoided. As a result, we get the following:

Theorem 5.9 *Let $\mathcal{K}$ be a $\mathcal{RIQ}$ knowledge base with a regular RBox and let q be a 2PRPQ over $\mathcal{K}$. $\mathcal{K} \models q$ is decidable in double exponential time in the combined size of q, $\mathcal{C}_{\mathcal{K}}$, $\mathcal{J}$ and $\mathsf{exp}(\mathcal{R})$; and in triple exponential time in the combined size of q, $\mathcal{C}_{\mathcal{K}}$, $\mathcal{J}$ and $\mathcal{R}_{\mathcal{K}}$. Furthermore, if the RBox in $\mathcal{K}$ is simple, then $\mathcal{K} \models q$ is decidable in double exponential time in the combined size of q, $\mathcal{C}_{\mathcal{K}}$, $\mathcal{J}$ and $\mathcal{R}_{\mathcal{K}}$.*

5.1 Extending the algorithm to $\mathcal{SRIQ}$

In [14] the logic $\mathcal{RIQ}$ was extended to $\mathcal{SRIQ}$. This logic has gained increasing attention recently, since it is closely related to the logic $\mathcal{SROIQ}$ underlying OWL 1.1., the most recent standard for Web Ontology Languages. Apart from the standard role inclusion axioms, a $\mathcal{SRIQ}$ RBox may explicitely state other properties of roles, like transitivity, (ir)reflexivity, disjointness, etc. Some of this additions are just syntactic sugar, while others slightly increase the expressivity of the logic and make it more suitable for ontology engineering, while requiring only minor adaptations of the reasoning techniques.

Indeed, most of the role assertions of a $\mathcal{SRIQ}$ knowledge base can be expressed in $\mathcal{RIQ}$ extended with concepts of the form $\exists \mathsf{Self}.R$, whose extension is the set of individuals x such that the pair (x, x) is in the extension of the role R. These concepts can not be directly expressed in the logic $\mathcal{ALCQI}b_{reg}$, but the automata algorithm we have presented can be easily extended to handle them. Roughly, when a node x is related by R to itself, a new label is added to the node representing this R-loop. The automata can then navigate the tree as usual and use these labels to verify the satisfaction of the $\exists \mathsf{Self}.R$ concepts.

The *role disjointness* assertions of $\mathcal{SRIQ}$ are not expressible in $\mathcal{RIQ}$ (even extended with $\exists \mathsf{Self}.R$ concepts), but they can be easily simulated in $\mathcal{ALCQI}b_{reg}$due to the presence of Boolean role constructors; e.g., we can add, for each assertion expressing the disjointness of two roles R and S, a TBox axiom of the form $\forall R \sqcap S.\bot$ (note that such assertions in $\mathcal{SRIQ}$ are restricted to simple roles). Finally, $\mathcal{SRIQ}$ supports negative role assertions, which we did not consider, but they can be easily incorporated by using suitable labels at the root and adapting the transition from the initial state that checks that the ABox assertions are satisfied. With these simple adaptations, our technique also provides a 2EXPTIME procedure for deciding decidability of $\mathcal{SRIQ}$ knowledge bases, and a 3EXPTIME procedure for the entailment of P2RPQs over them.

6 EXPSPACE-Hardness of Query Answering

In this section, we provide the following lower bound on answering P2RPQs over $\mathcal{ALCQI}b_{reg}$ KBs.

Theorem 6.1 *Given a P2RPQ q and a $\mathcal{ALCQI}b_{reg}$ knowledge base $\mathcal{K}$, deciding whether $\mathcal{K} \models q$ is* EXPSPACE*-hard.*

The proof is by a reduction from tiling problems, inspired by a similar reduction to query containment over semi-structured data [6].

A tiling problem consists of a finite set Δ of tile types, two binary relations H and V over Δ, representing horizontal and vertical adjacency relations, respectively, and two distinguished tile types $t_S, t_F \in \Delta$. Deciding whether for a given a number n in unary, a region of the integer plane of size $2^n \times k$, for some k, can be tiled consistently with H and V, such that the left bottom tile of the region has type t_S and the right upper tile has type t_F, can be shown to be EXPSPACE-complete [26].

We construct an $\mathcal{ALC}$ KB $\mathcal{K}$ and a query q such that $\mathcal{K} \models q$ iff there is no correct tiling, as follows. A tiling is spanned row by row by a sequence of objects. Each object represents one tile and is connected by a specific role to the next tile. For the connections, we use the following two roles:

- N connecting tiles within the same row;
- L connecting the last tile of row i to the first of row $i+1$.

The properties (i.e., the atomic concepts) attached to an object are the n bits $B_1, \ldots, B_n$ of a counter for its address within the row, and its type. For that, we use pairwise disjoint concepts $D_1, \ldots, D_k$, where $\Delta = \{t_1, \ldots, t_k\}$.

We encode in $\mathcal{K}$ the following two conditions:

1. The first ensures that the counters progress correctly. It consists of $O(n)$ standard axioms involving $B_1, \ldots, B_n$ and N, which encode a counter bit by bit. Further axioms ensure that, if at least one of the bits is 0, there is an N successor but no L successor, and reset the counter:

$$\begin{aligned} \neg B_1 \sqcup \cdots \sqcup \neg B_n &\sqsubseteq \exists N.\top \sqcap \forall L.\bot \\ B_1 \sqcap \cdots \sqcap B_n &\sqsubseteq \exists L.(\neg B_1 \sqcap \cdots \sqcap \neg B_n) \sqcap \forall N.\bot \end{aligned}$$

2. The second ensures that there are no errors w.r.t. the horizontal adjacency relation H: For each tile type D_i,
$$D_i \sqsubseteq \bigsqcup_{(D_i,D_j)\in H}(\forall N.D_j \sqcap \forall L.D_j).$$

The query q checks the failure of the vertical adjacency V on the candidate tilings given by the models of $\mathcal{K}$. It asks whether two objects exist at distance 2^n (i.e., representing vertically adjacent tiles) with an error according to V. That the objects are exactly 2^n steps apart is achieved by ensuring that they have the same n bits and are connected by a (possibly void) sequence of N-steps, followed by one L-step, and by a (possibly void) sequence of N-steps. We have

$$q = \exists x, y.\mathit{Vert} \wedge \mathit{Err} \wedge G_1 \wedge \cdots \wedge G_n, \quad \text{where}$$
$$\begin{aligned}\mathit{Vert} &= (N^* \circ L \circ N^*)(x,y),\\ \mathit{Err} &= \textstyle\bigvee_{(D_i,D_j)\notin V}(D_i(x) \wedge D_j(y)),\\ G_i &= (B_i(x) \wedge B_i(y)) \vee (\neg B_i(x) \wedge \neg B_i(y)), \text{ for } 1 \le i \le n.\end{aligned}$$

The complete KB $\mathcal{K}$ entails q iff there is no correct tiling. Note that only Vert uses a regular expression. If we have transitive roles and role hierarchies, we can replace it in q by

$$\begin{aligned}\mathit{Vert}' = (&N_t(x,z_1) \wedge L(z_1,z_2) \wedge N_t(z_2,y)) \vee\\ &(N_t(x,z_1) \wedge L(z_1,y)) \vee (L(x,z_2) \wedge N_t(z_2,y))\end{aligned}$$

where N_t is a transitive super-role of N, and z_1 and z_2 are existentially quantified variables. This shows that answering positive (existential) queries without regular expressions over KBs in $\mathcal{ALC}$ plus transitive roles and role hierarchies, and hence in $\mathcal{SHIQ}$, is EXPSPACE-hard.

Finally, using an encoding closer to [6] where each tile is a block of $n+1$ objects, and the bits and tile types are encoded by roles, one can show that conjunctive regular path queries over KBs which only use existential roles and disjunction are EXPSPACE-hard.

7 Conclusion

In this paper, we have substantially pushed the frontier of decidable query answering over expressive DLs, which is an active area of research driven by the growing interest to deploy DLs to various application areas related to AI. As we have shown, the rich class of positive (existential) two-way regular path queries (P2RPQs) is decidable on $\mathcal{ALCQI}b_{reg}$ KBs by means of automata-techniques; on the other hand, query answering has an EXPSPACE-lower bound already in settings where one of $\mathcal{K}$ and Q is rather plain.

Several tasks remain for future investigation. The precise complexity of answering P2RPQs remains open, even though the gap between the EXPSPACE lower bound and the 2EXPTIME upper bound is relatively small. Similarly, the class of positive existential queries without regular expressions over $\mathcal{SHIQ}$ KBs remains to be further analyzed, in particular whether the EXPSPACE bound is tight. Finally, it would be interesting to see how far automata-based techniques similar as in this paper can be utilized to push the decidability frontier of query answering in expressive DLs, both on the side of the query and the knowledge base.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[3] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. 2ATAs make DLs easy. In *Proceedings of the 2002 Description Logic Workshop (DL 2002)*, pages 107–118. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-53/`, 2002.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 23, pages 1581–1634. Elsevier Science Publishers, 2001.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proceedings of the Seventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 176–185, 2000.

[7] D. Calvanese, T. Eiter, and M. Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proceedings of the Twentysecond AAAI Conference on Artificial Intelligence (AAAI 2007)*, pages 391–396, 2007.

[8] O. Carton. Chain automata. *Theoretical Computer Science*, 161(1-2):191–203, 1996.

[9] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the Thirtysecond Annual Symposium on the Foundations of Computer Science (FOCS'91)*, pages 368–377, 1991.

[10] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[11] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 399–404, 2007.

[12] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In B. Parsia, U. Sattler, and D. Toman, editors, *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, volume 189, Windermere, Lake District, United Kingdom, May 2006.

[13] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[14] I. Horrocks, O. Kutz, and U. Sattler. The irresistible $\mathcal{SRIQ}$. In *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED 2005)*, 2005.

[15] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*. Morgan-Kaufmann Publishers, 2003.

[16] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proceedings of the Eleventh International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, pages 21–35, 2004.

[17] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.

[18] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logics*, 69(2–3):243–268, 1994.

[19] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[20] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.

[21] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, Mar. 2006. Available at `http://www.inf.unibz.it/~calvanese/papers/orti-calv-eite-TR-2006-03.pdf`.

[22] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. In B. Parsia, U. Sattler, and D. Toman, editors, *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, volume 189, Windermere, Lake District, United Kingdom, May 2006.

[23] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[24] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory*, volume III, pages 389–455. 1997.

[25] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[26] P. van Emde Boas. The convenience of tilings. In A. Sorbi, editor, *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture Notes in Pure and Applied Mathematics*, pages 331–363. Marcel Dekker Inc., 1997.

[27] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of the Twentyfifth International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.