Answering regular path queries in expressive Description Logics via alternating tree-automata

Diego Calvanese, Thomas Eiter, Magdalena Ortiz

A R T I C L E   I N F O

Article history:
Received 23 December 2009
Received in revised form 3 April 2013
Available online 18 April 2014

Keywords:
Expressive Description Logics
Query answering
Computational complexity
Automata on infinite trees

A B S T R A C T

Expressive Description Logics (DLs) have been advocated as formalisms for modeling the domain of interest in various application areas, including the Semantic Web, data and information integration, peer-to-peer data management, and ontology-based data access. 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 2-way regular path queries (P2RPQs) over knowledge bases in the expressive DL $\mathcal{ZIQ}$. P2RPQs are more general than conjunctive queries, union of conjunctive queries, and regular path queries from the literature. They allow regular expressions over roles and data joins that require inverse paths. The DL $\mathcal{ZIQ}$ extends the core DL $\mathcal{ALC}$ with qualified number restrictions, inverse roles, safe Boolean role expressions, regular expressions over roles, and concepts of the form $\exists S Self$ in the style of the DL $\mathcal{SRIQ}$. Using techniques based on two-way tree-automata, we first provide as a stepping stone an elegant characterization of TBox and ABox satisfiability testing which gives us a tight ExpTime bound for this problem (under unary number encoding). We then establish a double exponential upper bound for answering P2RPQs over $\mathcal{ZIQ}$ knowledge bases; this bound is tight. Our result significantly pushes the frontier of $2\text{ExpTime}$ decidability of query answering in expressive DLs, both with respect to the query language and the considered DL. Furthermore, by reducing the well known DL $\mathcal{SRIQ}$ to $\mathcal{ZIQ}$ (with an exponential blow-up in the size of the knowledge base), we also provide a tight $2\text{ExpTime}$ upper bound for knowledge base satisfiability in $\mathcal{SRIQ}$ and establish the decidability of query answering for this significant fragment of the new OWL 2 standard.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Description Logics (DLs) [2] is a well-established branch of logics for knowledge representation and reasoning, and today the premier logic-based formalisms for modeling concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). It has 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, many of the standard Web ontologies from the OWL family are based on DLs: the new OWL 2 standard [3] is based on a DL known as $\mathcal{SROIQ}$ [4], whose fragment $\mathcal{SRIQ}$ [5] extends the DL $\mathcal{SHIQ}$ underlying OWL-Lite [6].

Some results of this paper have appeared, in preliminary form, in a conference paper in the Proc. of AAAI 2007 [1].

Corresponding author.
E-mail addresses: calvanese@inf.unibz.it (D. Calvanese), eiter@kr.tuwien.ac.at (T. Eiter), ortiz@kr.tuwien.ac.at (M. Ortiz).
In DLs, reasoning tasks like classification and instance checking, which deal with taxonomic issues, had been traditionally studied. However, the widening range of applications in which DLs are used has motivated an increasing interest in query languages whose expressive power goes beyond that of DL concept and role expressions. The aim of such languages is to allow one to piece together pieces of information in finding the query answer, thus overcoming one of the most significant drawbacks of DLs as languages for data management. Since the initial work of Calvanese et al. [7], many further works have addressed the problem of evaluating complex queries over DL knowledge bases. Special attention has been devoted to \((\text{unions of})\) conjunctive queries (CQs and UCQs) [8], which are the formal counterpart of the most widely used fragments of SQL (or relational algebra) queries, namely \((\text{unions of})\) select-project-join queries. (U)CQs over DL knowledge bases have been studied for many DLs, ranging from weak ones that allow for efficient algorithms, like those of the \(\mathcal{EL}\) [9–11] and DL-Lite families [12], to the very expressive ones of the \(\mathcal{ALC}^H\) and \(\mathcal{SH}\) families, cf. [13–16].

Another important language for querying knowledge bases is that of \textit{regular path queries} (RPQs) [17–19], which allow one to ask for pairs of objects that are connected by a path conforming to a regular expression. Due to their capability of expressing complex navigations in graphs, RPQs are the fundamental mechanism for querying semi-structured and graph-structured data. Indeed, as a query language, RPQs go beyond first-order logic, since they allow one to express a \textit{controlled} form of recursion. This turns out to be essential for querying graph-like structures such as those encountered in several domains that are gaining increasing importance, notably social networks [20] and linked open data [21]. Notice that, in a setting of incomplete information as the one encoded by means of a knowledge base, the use of unrestricted recursion would quickly lead to undecidability, not only of intensional inference tasks such as query containment or equivalence [22], but also of query answering [23,24]. Instead, the restricted form of recursion provided by RPQs and their extensions considered here, provides a good trade-off between the ability to flexibly traverse data whose precise structure is not defined a priori (e.g., in terms of a relational schema), and the decidability of query answering also in the presence of complex domain knowledge encoded in DLs. The complex paths allowed in the query allow one to find in the data complex relations between items, without being constrained by the relations explicitly stated in the data or pre-defined in the ontology, and without having to modify the ontology solely for query answering. Moreover, when also \textit{inverse roles} are allowed to occur in the regular expression, the complex relations expressed by the resulting \textit{two-way} RPQs (2RPQs) are not restricted to the direction initially chosen by the designer to represent relations between data items. 2RPQs are for example present in the property paths in SPARQL 1.1 [25], the new standard RDF query language, and in the XML query language XPath [26]. We consider here the yet more expressive class of \textit{positive (existential) two-way regular path queries} (in short, 2P2RPQs), which are inductively built using conjunction and disjunction from atoms that are regular expressions over direct and inverse roles and allow for testing the objects encountered during navigation for membership in concepts. 2P2RPQs, which subsume CQs and unions of CQs, are also a natural generalization of several extensions of RPQs that have been studied by different authors, e.g., [27,28,19,29–32]. They are, to our knowledge, the most expressive query language considered so far over DL knowledge bases [33,1].

In this paper, we describe a technique, first presented in [1], for deciding the entailment problem for P2RPQs expressed over \(\mathcal{ZIQ}\) knowledge bases. In query entailment, we are given a knowledge base and a Boolean query, i.e., a query that in a given interpretation evaluates either to true or to false, expressed over that knowledge base, and we are asked to determine whether the query evaluates to true in every model of the knowledge base. The DL \(\mathcal{ZIQ}\), also known as \(\mathcal{ALC}^{\mathcal{QI}b}_{\mathcal{Self}}\), 

extends the well known DL \(\mathcal{ALCQIb}\) (to which reasoning in \(\mathcal{SHIQ}\) can be reduced [34]) with regular role expressions [35], Boolean role inclusion axioms, and concepts of the form \(\exists \mathcal{S}\mathcal{SELF}\) [5]. By means of a translation that reduces the query entailment problem over \(\mathcal{SRIQ}\) KBs to \(\mathcal{ZIQ}\) KBs, we also obtain an algorithm for entailment of P2RPQs over \(\mathcal{SRIQ}\) knowledge bases. This is the first algorithm for query entailment (and hence for query answering) that allows both for regular expressions and for conjunctions of atoms in the query, while considering, on the DL side, a logic that extends \(\mathcal{ALC}\) with inverses and counting and, notably, also supports the kind of complex role inclusions that have been advocated in the new OWL standards [3].

Previously, algorithms for query answering over expressive DLs had used a variety of techniques, ranging from query rewriting [7,13,36], over modified tableaux techniques [16], to resolution [37]. We obtain our results by exploiting techniques based on \textit{automata on infinite trees} [38], which have been developed initially in the context of modal logics and program logics [39–43]. While the application of automata techniques in DLs is not novel, cf. [35,44,45], previous work was concerned with deciding satisfiability of a knowledge base consisting of a taxonomy part (TBox) only. Here we address the much more involved task of query answering over a knowledge base, which also has a data part (an ABox). Specifically, we extend previous automata-based algorithms for TBox satisfiability [35,44] and incorporate the ABox part. Then, to decide query entailment over DL knowledge bases, we build on the ideas of Calvanese et al. [30], which had been developed in the context of automata on finite words, and extend them to automata over infinite trees. For deciding query entailment, we implement automata operations that rely on transformations between different kinds of automata, which, from a technical point of view, are more challenging in our case than in the case of finite words. The technique we present here has been recently extended to some DLs that support nominals [33].

In this paper, we make the following contributions (all complexity results hold under unary number encoding):

- As a stepping stone to our main results, we first present an automata-based algorithm for checking the satisfiability of a \(\mathcal{ZIQ}\) knowledge base that comprises both a TBox and an ABox, and that runs in \textit{ExpTime}, which is worst-case optimal.
We then show that answering P2RPQs over $ZIQ$ knowledge bases is feasible in 2ExpTime. By the aforementioned reduction [34], the same bound holds for $SHIQ$. From known hardness results for answering CQs over $ALCt$ [46] and $SH$ [47] KBs, it follows that this is worst case optimal. In fact, a simple adaptation of the proof in [47] shows that the matching lower bounds hold even for answering positive queries (which do not allow regular expressions over roles in atoms) and conjunctive RPQs (i.e., P2RPQs that use only conjunction and no inverse roles) over plain $ALC$ KBs. This shows that, once either inverse roles or role hierarchies and transitivity are allowed in the KB, or alternatively, regular expressions or disjunctions are allowed in the query, one can significantly extend both the query language and the DL considered without further increasing the worst case complexity of the query entailment problem.

We provide a rewriting that, with an unavoidable exponential blow-up, translates a $SRIQ$ knowledge base into a $ZIQ$ knowledge base. In this way we obtain a relevant result: a new tight 2ExpTime upper bound for knowledge base satisfiability in $SRIQ$, the nominal free-fragment of OWL 2.

Furthermore, we show that entailment for P2RPQs is decidable over $SRIQ$ knowledge bases (in fact, the problem is in 3ExpTime); this is the first decidability result for query entailment in an expressive DL with complex role hierarchies, and identifies the first expressive fragment of OWL 2 for which decidability of query entailment has been established. For full OWL 2 (i.e., the DL $SROIQ$), decidability of query entailment remains open.

The rest of this article organized as follows. We first give some technical preliminaries in Section 2. Then, in Section 3, we discuss in detail some properties of $ZIQ$ KBs and present some transformations on them that lie at the core of our automata algorithms. In Section 4, we describe the automata-based technique for satisfiability of $ZIQ$ KBs, and in Section 5 its extension to query entailment. In Section 6, we present the rewriting from $SRIQ$ to $ZIQ$, obtaining algorithms for KB satisfiability and query entailment in this logic. In Section 7, we draw final conclusions. In order to increase readability, technical details of some proofs have been moved to Appendix A.

2. Preliminaries

In this section, we define the main Description Logic (DL) and the query answering problem considered in this article. We also provide some general preliminaries on automata on infinite trees. Throughout the paper, we use $|X|$ to denote the cardinality of a set $X$, and $|X|$ to denote the length of some string encoding $X$. For any word $w$, $|w|$ denotes the length of $w$, i.e., the number of its symbols.

2.1. The Description Logic $ZIQ$

$ZIQ$ is the short name for the DL $ALCQIb^{Self}_{reg}$, which extends the well known DL $ALCQIb$ [34] with regular role expressions, Boolean role inclusion axioms, and concepts of the form $\exists S.Self$ in the style of $SRIQ$ [5]. In turn, $ALCQIb$ extends the basic DL $ALC$ with qualified number restrictions and inverses, which are available in $SHIQ$, $SRIQ$ and other well known DLs, and supports safe Boolean expressions over simple roles. $ZIQ$ is a slight extension of $ALCQIb^{Self}_{reg}$ considered in [35,1].

Definition 2.1 (Concepts and roles). We consider fixed, countably infinite and pairwise disjoint alphabets $C$ of concept names (also called atomic concepts), $R$ of role names, and $I$ of individual names. We assume that $C$ contains the special concepts $\top$ (top) and $\bot$ (bottom), while $R$ contains the top (or universal) role $T$ and the bottom (or empty) role $B$. Concepts $C, C'$ and roles $p, S, S', R, R'$, are formed according to the following syntax, where $A \in C$ and $p \in R \setminus \{T\}$:

$\begin{align*}
C, C' & \rightarrow A | \neg C | C \cap C' | C \cup C' | \forall R.C | \exists R.C | \geq n.S.C | \leq n.S.C | \exists S.Self \\
p & \rightarrow p | \overline{p} \\
S, S' & \rightarrow P | S \cap S' | S \cup S' | S \setminus S' \\
R, R' & \rightarrow T | S | R \cup R' | R \circ R' | \overline{R} | \overline{id(C)}
\end{align*}$

We call roles $P$ atomic, and roles $S, S'$ simple. Note that, as $p \neq T$, the top role $T$ may occur in arbitrary roles $R, R'$ but not in simple roles $S, S'$. A $ZIQ$ expression is a concept or a role. The set of subconcepts (subroles) of a given concept (resp., role) is defined in the natural way considering the syntactic structure of the concept (resp., role).

Definition 2.2 (Knowledge base). A concept inclusion axiom (CIA) is of the form $C \sqsubseteq C'$, where $C$ and $C'$ are arbitrary concepts, while a Boolean role inclusion axiom (BRIA) is of the form $S \sqsubseteq S'$ where $S$ and $S'$ are simple roles. A TBox is a set of CIs and BRIAs. An assertion is of the form $C(a)$, $S(a, b)$, or $a \neq b$, where $C$ is a concept, $S$ is a simple role and $a, b \in I$. An ABox is a set of assertions.

\footnote{We omit parentheses in expressions following the usual conventions.}
A knowledge base (KB) is a pair \( \mathcal{K} = \langle T, A \rangle \), where \( T \) is a TBox and \( A \) is a non-empty ABox.\(^2\) We denote by \( C_\mathcal{K}, R_\mathcal{K}, \) and \( I_\mathcal{K} \) the sets of concept names, role names, and individuals occurring in \( \mathcal{K} \), respectively. Furthermore, we let \( \bar{R}_\mathcal{K} = R_\mathcal{K} \cup \{p^- \mid p \in R_\mathcal{K}\} \). □

**Definition 2.3 (Semantics).** An interpretation \( \mathcal{I} = (\Delta^\mathcal{I}, \mathcal{I}) \) consists of a non-empty domain \( \Delta^\mathcal{I} \) and a valuation function \( \mathcal{I} \) that maps each individual \( a \in I \) to an element \( a^\mathcal{I} \in \Delta^\mathcal{I} \), each concept name \( A \in C \) to a set \( A^\mathcal{I} \subseteq \Delta^\mathcal{I} \), and each role name \( p \in R \) to a set \( p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \), such that:

\[
\begin{align*}
\land^\mathcal{I} &= \emptyset, \\
\bot^\mathcal{I} &= \emptyset.
\end{align*}
\]

The function \( \mathcal{I} \) is inductively extended to all concepts and roles as follows:

\[
\begin{align*}
(-C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I}, \\
(p^-)^\mathcal{I} &= \{ (y, x) \mid (x, y) \in p^\mathcal{I} \} \\
(C \cap C')^\mathcal{I} &= C^\mathcal{I} \cap C'^\mathcal{I}, \\
(S \cap S')^\mathcal{I} &= S^\mathcal{I} \cap S'^\mathcal{I} \\
(C \cup C')^\mathcal{I} &= C^\mathcal{I} \cup C'^\mathcal{I}, \\
(R \cup R')^\mathcal{I} &= R^\mathcal{I} \cup R'^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{ x \mid \forall y.(x, y) \in R^\mathcal{I} \rightarrow y \in C^\mathcal{I} \}, \\
(S \setminus S')^\mathcal{I} &= S^\mathcal{I} \setminus S'^\mathcal{I} \\
(\exists R.C)^\mathcal{I} &= \{ x \mid \exists y.(x, y) \in R^\mathcal{I} \land y \in C^\mathcal{I} \}, \\
(R \circ R')^\mathcal{I} &= R^\mathcal{I} \circ R'^\mathcal{I} = \{ (x, y) \mid \exists z.(x, z) \in R^\mathcal{I} \land (z, y) \in R'^\mathcal{I} \} \\
(\geq n S)^\mathcal{I} &= \{ x \mid \{ y \mid (x, y) \in S^\mathcal{I} \} \geq n \}, \\
(R^\mathcal{I})^* &= \{ y \mid (x, y) \in R^\mathcal{I} \} \\
(\leq n S)^\mathcal{I} &= \{ x \mid \{ y \mid (x, y) \in S^\mathcal{I} \} \leq n \}, \\
(id(\mathcal{C}))^\mathcal{I} &= \{ (x, x) \mid x \in C^\mathcal{I} \},
\end{align*}
\]

where \( \circ \) denotes composition of binary relations and \( \cdot^* \) the reflexive transitive closure of a binary relation; \( \mathcal{I} \) satisfies (or, is a model of)

- a CIA or BRIA \( E \subseteq E' \), if \( E^\mathcal{I} \subseteq E'^\mathcal{I} \);
- an assertion \( C(a) \), if \( a^\mathcal{I} \in C^\mathcal{I} \), an assertion \( S(a, b) \), if \( (a^\mathcal{I}, b^\mathcal{I}) \in S^\mathcal{I} \), and an assertion \( a \not\approx b \), if \( a^\mathcal{I} \not\approx b^\mathcal{I} \);
- an ABox \( A \), if it satisfies every assertion in \( A \);
- a TBox \( T \), if it satisfies every CIA and BRIA in \( T \);
- a KB \( \mathcal{K} = \langle T, A \rangle \), if it satisfies both \( T \) and \( A \).

Satisfaction of a CIA, BRIA, assertion, ABox, etc. \( \eta \) is denoted by \( \mathcal{I} \models \eta \). Knowledge base satisfiability is the problem of deciding, given a KB \( \mathcal{K} \), whether there exists an interpretation \( \mathcal{I} \) such that \( \mathcal{I} \models \mathcal{K} \). □

We remark that we do not make the unique name assumption, which can be simulated using assertions of the form \( a \not\approx b \). Note also that we do not include in the language equality assertions \( a \approx b \), since their addition would not provide additional expressiveness. Indeed, they could be easily compiled away by replacing all individuals in the same equality equivalence class with one representative.

### 2.2. Query answering

We next introduce P2RPQs, which generalize conjunctive regular path queries [32,30] and unions thereof.

**Definition 2.4 (P2RPQs).** A positive 2-way regular path query (P2RPQ) is a formula \( \exists \bar{v}. \varphi(\bar{v}) \), where \( \bar{v} \) is a tuple of variables and \( \varphi(\bar{v}) \) is built using \( \land \) and \( \lor \) from atoms of the form \( C(\bar{v}) \) and \( R(\bar{v}, \bar{v}') \), where \( \bar{v}, \bar{v}' \) are variables from \( \bar{v} \) or individuals, \( C \) is a concept, and \( R \) is a role. If all atomic concepts and roles in \( \varphi \) occur in a KB \( \mathcal{K} \), the query is over \( \mathcal{K} \). We denote by \( \text{At}(q) \) the set of all atoms occurring in a P2RPQ \( q \). Let \( q = \exists \bar{v}. \varphi(\bar{v}) \) be a P2RPQ, and let \( V_q \) and \( I_q \) denote the sets of variables and individuals in \( q \), respectively. Given an interpretation \( \mathcal{I} \), let \( \pi : V_q \cup I_q \rightarrow \Delta^\mathcal{I} \) be a total function such that \( \pi(a) = a^\mathcal{I} \) for each individual \( a \in I_q \). We write \( \mathcal{I}, \pi \models C(\bar{v}) \) if \( \pi(\bar{v}) \in C^\mathcal{I} \), and \( \mathcal{I}, \pi \models R(\bar{v}, \bar{v}') \) if \( (\pi(\bar{v}), \pi(\bar{v}')) \in R^\mathcal{I} \). Let \( \mathcal{I}, \pi \) be the Boolean expression obtained from \( \varphi \) by replacing each atom \( \alpha \) in \( \varphi \) with true, if \( \mathcal{I}, \pi \models \alpha \), and with false otherwise. We say that \( \mathcal{I}, \pi \) is a match for \( \mathcal{I} \) and \( q \), denoted \( \mathcal{I}, \pi \models q \), if \( \mathcal{I} \models q \) evaluates to true. We say that \( \mathcal{I} \) satisfies \( q \), written \( \mathcal{I} \models q \), if there exists some match \( \pi \) 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 \( \mathcal{K} \).

Query entailment consists in verifying, given a KB \( \mathcal{K} \) and a P2RPQ \( q \), whether \( \mathcal{K} \models q \). □

\(^2\) If \( A = \emptyset \), we can always add \( \top(a) \) to \( A \) for some fresh individual name \( a \).
P2RPQs are a generalization of conjunctive queries (CQs), a well known query language widely studied in databases [48,8] and, more recently, in DLs [24,49,16,13]. A CQ is a P2RPQ in which neither disjunction (\(\lor\)) nor regular role expressions occur. We observe that the possibility of using regular role expressions in the query atoms of P2RPQs significantly increases the expressive power of the query language, since it allows one to express complex navigations in the models of the given KB, similar to those possible with (conjunctive) regular path queries studied in graph databases [32,30,19,27].

**Example 2.5.** We consider a genealogy KB \(K_g = (T, A)\), where \(T\) contains the CIs and BRIAs in the left column of Fig. 1, while \(A\) contains the assertions in the right column. We use \(E \equiv E'\) as a shortcut for \(E \subseteq E'\) and \(E' \subseteq E\).

The following query \(q_g\) is a P2RPQ over \(K_g\):

\[
q_g = \exists v_1, v_2, v_3. \text{HasParent}^*(v_1, v_2) \land \text{HasParent}^*(v_1, v_3) \land \text{HasParent}^*(v_2, v_3) \land \text{male}(v_1) \\
\land \text{female}(v_2) \land (\neg \text{deity}(v_1) \lor \neg \text{deity}(v_2))
\]

Informally, \(q_g\) asks whether there are a male and female individual (represented by \(v_1\) and \(v_2\), resp.) who are relatives (i.e., related by the expression \(\text{HasParent}^* \circ \text{HasParent}^*\)), are not both deities, and have a common child \(v_3\). Note that \(K_g \models q_g\), as \(\pi(v_1) = \text{Zeus}^2\), \(\pi(v_2) = \text{Alcmene}^2\) and \(\pi(v_3) = \text{Heracles}^2\) is a match for \(q_g\) in every model \(I\) of \(K_g\).

Note that we have restricted our attention to queries that are formulas without free variables, i.e., so called Boolean queries. We can consider w.l.o.g. the entailment problem for Boolean queries, since query answering for non-Boolean queries is polynomially reducible to query entailment.\(^3\) Note that the problem of deciding whether a given KB has a model can be trivially reduced to query non-entailment. Indeed, a KB \(K\) is satisifiable iff \(K \not\models \exists v. \bot\).

### 2.3. Automata on infinite trees

In the rest of this section, we recall the definitions of infinite labeled trees and of two way alternating automata over such trees [41].

**Definition 2.6.** An (infinite) tree is a prefix-closed set \(T \subseteq \mathbb{N}^*\) of words over the natural numbers \(\mathbb{N}\). If \(T \subseteq \{1, \ldots, k\}^*\) for some \(k \geq 0\), we call it a \(k\)-tree. The elements of \(T\) are called nodes, the empty word \(\varepsilon\) is its root. For every \(x \in T\), the nodes \(x \cdot c\) with \(c \in \mathbb{N}\) are the successors of \(x\), and \(x\) is the predecessor of each \(x \cdot c\): the ancestor relation is the transitive closure of predecessor. By convention, \(x \cdot 0 = x\), and \((x \cdot c) \cdot 0 = x\). We call \(T\) \(k\)-ary if it is a \(k\)-tree and each node in it has \(k\) successors (i.e., \(T = \{1, \ldots, k\}^*\)). An infinite path \(\pi\) of \(T\) is a prefix-closed set \(\pi \subseteq T\) where for every \(i \geq 0\) there exists a unique node \(x \in P\) with \(|x| = i\). A labeled tree over an alphabet \(\Sigma\) (or simply a \(\Sigma\)-labeled tree) is a pair \((T, L)\), where \(T\) is a tree and \(L : T \rightarrow \Sigma\) maps each node of \(T\) to an element of \(\Sigma\).

**2.3.1. Two-way alternating tree automata (2ATAs)**

Now we define two-way alternating tree automata (2ATAs) over infinite trees as introduced in [41], which generalize the standard non-deterministic (one-way) automata on infinite trees (1NTAs) in two ways. First, alternation is a generalization of non-determinism that allows for an elegant and compact encoding of decision problems in several logics [50]. Second, two-way automata are better suited for logics that have ‘backward’ operators, like inverse roles, since they may move up on the input tree or stay at the current position. In contrast, one-way automata navigate (infinite) trees in a strictly top-down manner, moving always to the successors of the current node.

**Definition 2.7.** Given a finite set \(I\) of propositional atoms, let \(B(I)\) be the set of positive Boolean formulas built inductively using \(\land\) and \(\lor\) from true, false and atoms from \(I\). A set \(J \subseteq I\) satisfies a formula \(\varphi \in B(I)\), if assigning true to the atoms in \(J\) and false to those in \(I \setminus J\) makes \(\varphi\) true. A two-way alternating tree automaton (2ATA) (running over \(k\)-ary trees) is a tuple \(A = (k, \Sigma, Q, \delta, s_0, F)\), where:

---

\(^3\) Here we refer to the associated decision problem, i.e., whether a given tuple is in the query answer.
\( \Sigma \) is the input alphabet;

- \( Q \) is a finite set of states;
- \( \delta : [k] \times \Sigma \rightarrow \mathcal{B}(k) \), where \([k] = \{-1, 0, 1, \ldots, k\} \), is the transition function;
- \( s_0 \in Q \) is the initial state; and
- \( F = \{G_1, \ldots, G_n\} \) with \( G_1 \subseteq G_2 \subseteq \cdots \subseteq G_n = Q \) is a (parity) acceptance condition, whose length \( n \), denoted by \( \text{ind}(A) \), is called the index of \( A \).

We refer to each component \( \Sigma, Q, \) etc. of \( A \) by \( \Sigma(A), Q(A), \) etc., respectively. \( \Box \)

The transition function \( \delta \) maps a state \( s \in Q \) and an input letter \( \sigma \in \Sigma \) to a positive Boolean formula \( \delta(s, \sigma) = \varphi \) over the atoms in \([k] \times Q\). Intuitively, each atom \( (c, s') \) in \( \varphi \) corresponds to a new copy of the automaton going in the direction given by \( c \) and starting in state \( s' \). For example, let \( k = 2 \) and \( \delta(s_1, \sigma) = (1, s_2) \wedge (1, s_3) \vee (s_1, 0) \). If \( A \) is in the state \( s_1 \) and reads a node \( x \) labeled with \( \sigma \), it proceeds by sending off either (i) two copies, in the states \( s_2 \) and \( s_3 \) respectively, to the first successor of \( x \) (i.e., \( x\downarrow 1 \)), or (ii) one copy in the state \( s_1 \) to the predecessor of \( x \) (i.e., \( x\uparrow 1 \)) and one copy in the state \( s_3 \) to \( x \) itself (i.e., \( x \)). For convenience, we may specify the transition function \( \delta \) only partially, and assume that \( \delta(s, \sigma) = \text{false} \) if not specified otherwise.

Acceptance of 2ATAs is defined in terms of runs. Informally, a run of a 2ATA \( A \) over a \( \Sigma \)-labeled tree \((T, L)\) is a labeled tree \((T, r)\) in which each node \( n \) is labeled by an element \( r(n) = (x, s) \in T \times Q \) and describes a copy of \( A \) that is in the state \( s \) and reads the node \( x \) of \( T \); the labels of adjacent nodes must satisfy the transition function of \( A \). Formally, we define the following generalized notion of a run, called \((x, s)\)-run:

**Definition 2.8.** Let \( A = (k, \Sigma, Q, \delta, s_0, F) \) be a 2ATA and let \((T, L)\) be a \( \Sigma \)-labeled \( k \)-ary tree. Moreover, let \( x \in T \) and \( s \in Q \). An \((x, s)\)-run of \( A \) over \((T, L)\) is a \((T \times Q)\)-labeled tree \((T, r)\) satisfying:

1. \( e \in T_r \) and \( r(e) = (x, s) \),
2. Each \( w \in T_r \) satisfies \( \delta \), i.e., if \( r(w) = (y, s') \) and \( \delta(s', L(y)) = \varphi \), then there is a (possibly empty) set \( W = \{(c_1, s_1), \ldots, (c_n, s_n)\} \subseteq [k] \times Q \) such that:
   - \( W \) satisfies \( \varphi \), and
   - for every \( 1 \leq i \leq n \), it holds that \( w \cdot i \in T_r \), \( y \cdot c_i \) is defined, and \( r(w \cdot i) = (y \cdot c_i, s_i) \).

We say that \((T, r)\) visits \( y \) in state \( s' \), if there exists some \( w \in T_r \) with \( r(w) = (y, s') \). An infinite path \( \pi \) of \( T_r \) satisfies the acceptance condition \( F \) of \( A \), if there exists an even \( i \geq 0 \) such that \( \text{Inf}(\pi) \cap G_{i-1} = \emptyset \) and \( \text{Inf}(\pi) \cap G_i \neq \emptyset \), where \( \text{Inf}(\pi) = \{ s \in Q | r(n) = (x, s) \} \) for infinitely many \( n \in \pi \). The \((x, s)\)-run \((T, r)\) is accepting, if all its infinite paths satisfy \( F \).

We call an \((\epsilon, s_0)\)-run a (full) run. A 2ATA \( A \) accepts a \( \Sigma \)-labeled tree \( T \), if there exists a (full) accepting run of \( A \) over \( T \); \( L(A) \) denotes the set of all trees that \( A \) accepts. The nonemptiness problem is to decide whether \( L(A) \neq \emptyset \) for a given \( A \). \( \Box \)

The following result is well-known.

**Theorem 2.9.** (See [41].) Nonemptiness of a given 2ATA \( A \) is decidable in time single exponential in \(|Q(A)|\) and polynomial in \(|\Sigma(A)|\).

We will often take intersections of 2ATAs, relying on the fact that this operation is trivial.

**Proposition 2.10.** Given 2ATAs \( A_1, \ldots, A_n \), it is possible to construct a 2ATA \( A \) with \(|Q(A)| = \sum_{i=1}^n |Q(A_i)| + 1 \) and \( \text{ind}(A) = \max_{i=1}^n \text{ind}(A_i) \) such that \( L(A) = \bigcap_{i=1}^n L(A_i) \).

**Proof (Sketch).** Assuming the state sets \( Q(A_i) \) are pairwise disjoint, introduce a new state \( s_0^g \) and let \( Q(A) \) be the union of all \( Q(A_i) \) plus \( s_0^g \), which is the initial state of \( A \). The transition function \( \delta(A) \) consists of the union of all \( \delta(A_i) \) and has \( \delta(s_0^g, \sigma) = \bigvee_{i=1}^n \delta(s_0, A_i, \sigma) \) for all \( \sigma \in \Sigma \). For each \( A_i \), let \( F(A_i) = (G_1^i, \ldots, G_{m_i}^i) \) and let \( m = \max_{i=1}^n m_i = \max_{i=1}^n \text{ind}(A_i) \). To obtain \( F(A) \), we first ‘fill up’ all \( F(A_i) \) to \( F'(A_i) \) with index \( m \), by setting \( G_{m_i+1}^i = G_{m_i+2}^i = \cdots = G_m = Q(A_i) \). Then \( F(A) \) consists of the component-wise union of all the \( F'(A_i) \). \( \Box \)

Automata on infinite trees provide elegant solutions for decision problems in temporal and program logics [51], which has been widely exploited for providing optimal complexity bounds for the satisfiability problem of many variants of PDL, the \( \mu \)-calculus, and similar logics [41,40]. They have also been explored in DLs, but mostly for deciding concept satisfiability [45,35], given that in many DLs, concepts have tree-shaped models.

2.3.2. (One-way) non-deterministic tree automata (1NTAs)

Standard non-deterministic (one-way) automata on infinite trees can be defined as particular 2ATAs that always move to the \( k \) successors of the current node and switch to states that are given by a tuple of \( k \) states, one for each successor. This can be expressed as a formula in disjunctive form:
Definition 2.11. A one-way non-deterministic automaton (1NTA) is a 2ATA \( \mathbf{A} = (k, \Sigma, Q, \delta, \sigma_0, F) \) such that for every \( s \in Q \) and every \( \sigma \in \Sigma \), \( \delta(s, \sigma) \) is of the form \( ((1, s_1^i) \wedge \cdots \wedge (k, s_k^i)) \vee \cdots \vee ((1, s_1^\ell) \wedge \cdots \wedge (k, s_k^\ell)) \), with \( \ell > 0 \), and \( s_j^i \in Q \) for each \( 1 \leq i \leq \ell \) and each \( 1 \leq \ell \leq k \). \( \square \)

We recall some results on 1NTAs.

Proposition 2.12. (See [41].) Given a 2ATA \( \mathbf{A} \), it is possible to construct a 1NTA \( \mathbf{A}_1 \), with \( |Q(\mathbf{A}_1)| \leq 2^{D(|Q(\mathbf{A})|^c)} \) for some constant \( c \), and \( \text{ind}(\mathbf{A}_1) = O(\text{ind}(\mathbf{A})) \), such that \( \mathcal{L}(\mathbf{A}_1) = \mathcal{L}(\mathbf{A}) \).

The following bounds for automata complementation are given in [52].

Proposition 2.13 (Complementation). For every 1NTA \( \mathbf{A} \) running over \( k \)-ary trees, it is possible to construct a 1NTA \( \mathbf{A}' \) that accepts a \( k \)-ary \( \Sigma \)-labeled tree \( (T, L) \) iff \( (T, L) \notin \mathcal{L}(\mathbf{A}) \), and such that \( |Q(\mathbf{A}')| \leq 2^{D(f(\mathbf{A}))} \) and \( \text{ind}(\mathbf{A}') = O(f(\mathbf{A})) \), where \( f(\mathbf{A}) = \text{ind}(\mathbf{A})|Q(\mathbf{A})| \cdot \log |Q(\mathbf{A})| \).

Another automata theoretic operation we exploit is projection, which restricts the trees in the language of an automaton to a smaller alphabet. For \( \Sigma = 2^Q \) and \( \Sigma' \subseteq \Phi \), and for a \( \Sigma \)-labeled tree \( T = (T, L) \), the \( \Sigma' \)-restriction of \( T \) is the \( \Sigma' \)-labeled tree \( T' = (T', L') \), where \( L' \) is obtained by restricting \( L \) to \( \Sigma' \). The \( \Sigma' \)-projection of a set \( \mathcal{L} \) of \( \Sigma \)-labeled trees is the set consisting of the \( \Sigma' \)-restrictions of all trees in \( \mathcal{L} \).

Proposition 2.14 (Projection). For \( \Sigma \) and \( \Sigma' \) as above, for every 1NTA \( \mathbf{A} \) running over \( \Sigma \)-labeled trees, it is possible to construct a 1NTA \( \mathbf{A}^{\Sigma'} \) with \( |Q(\mathbf{A}^{\Sigma'})| \leq |Q(\mathbf{A})| \) and \( \text{ind}(\mathbf{A}^{\Sigma'}) \leq \text{ind}(\mathbf{A}) \) that accepts the \( \Sigma' \)-projection of \( \mathcal{L}(\mathbf{A}) \).

Proof. To obtain \( \mathbf{A}^{\Sigma'} \) from \( \mathbf{A} \), simply change \( \Sigma \) to \( \Sigma' \) and the transition function to \( \delta' \) as follows. For each \( \sigma \in \Sigma' \) and each \( q \in Q(\mathbf{A}) \), \( \delta'(q, \sigma) = \bigvee_{\sigma' \in \Sigma(\sigma)} \delta(q, \sigma') \), where \( \Sigma(\sigma) = \{ \sigma' \in \Sigma | \sigma' \cap \Phi' = \sigma \} \). \( \square \)

The following bounds for the intersection of two 1NTAs are known:

Proposition 2.15 (Intersection). Given 1NTAs \( \mathbf{A}_1 \) and \( \mathbf{A}_2 \), it is possible to construct a 1NTA \( \mathbf{A} \) such that \( \mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A}_1) \cap \mathcal{L}(\mathbf{A}_2) \) with \( \text{ind}(\mathbf{A}) = O(f(\mathbf{A}_1, \mathbf{A}_2)) \) and \( |Q(\mathbf{A})| \leq 2^{O(f(\mathbf{A}_1, \mathbf{A}_2))} \cdot f(\mathbf{A}_1, \mathbf{A}_2) \cdot |Q(\mathbf{A}_1)| \cdot |Q(\mathbf{A}_2)| \), where \( f(\mathbf{A}_1, \mathbf{A}_2) = \text{ind}(\mathbf{A}_1) + \text{ind}(\mathbf{A}_2) + 1 \).

Finally, testing a 1NTA for emptiness is feasible within the following bounds.

Proposition 2.16. (See [53].) Given a 1NTA \( \mathbf{A} \), the nonemptiness problem is decidable in time \( O(|Q(\mathbf{A})|^\text{ind}(\mathbf{A})) \).

3. Normal form and canonical models

In this section we prove some properties of KBs and define key notions that allow us to develop then the automata algorithm for reasoning in \( ZIQ \).

3.1. Normalizing knowledge bases

First of all, we will prove a quite simple property of KBs that will be useful later: they have connected models, in which every node can be reached from the interpretation of some ABox individual by a sequence of roles.

Let \( K \) be a KB. We say that an element \( d \in \Delta^Z \) is \( K \)-connected to an element \( d_0 \in \Delta^Z \) in an interpretation \( \mathcal{I} \), if there is some sequence \( d_0, \ldots, d_n \) such that \( d = d_n \) and for each \( 0 < i < n \) we have \( (d_i, d_{i+1}) \in P^Z \) for some \( P \in \mathbb{R}_K \). An interpretation \( \mathcal{I} \) is called \( K \)-connected, if each \( d \in \Delta^Z \) is \( K \)-connected to \( a^Z \) for some \( a \in I_K \).

Lemma 3.1 (Connected model property). Let \( K \) be a \( ZIQ \) KB. Then, for every \( P2RPQ \) \( q \), \( K \models q \) implies that there is a \( K \)-connected model \( \mathcal{I} \) of \( K \) with \( \mathcal{I} \not\models q \).

Proof (Sketch). We simply take some model \( \mathcal{I} \) of \( K \) with \( \mathcal{I} \not\models q \) and restrict it to the elements that are \( K \)-connected to \( a^Z \) for some \( a \in I_K \); the resulting interpretation \( \mathcal{I}' \) is \( K \)-connected by construction. It is trivial to verify that \( \mathcal{I}' \models K \). Indeed, for each \( d \in \Delta^Z \), removing elements not reachable from \( d \) does not alter the satisfaction of any concept at \( d \), nor the

---

4 To our knowledge, these bounds are unpublished. A tighter bound of \( |Q(\mathbf{A})| = f'(\mathbf{A}_1, \mathbf{A}_2) \cdot f'(\mathbf{A}_1, \mathbf{A}_2) \cdot |Q(\mathbf{A}_1)| \cdot |Q(\mathbf{A}_2)| \) was confirmed through personal communication with Yoad Lustig and Nir Piterman, to whom we are very grateful.
participation of \( d \) in the extension \( R^2 \) of any role \( R \) occurring in \( K \). Hence no CIA or BRIA is violated in \( T' \). The ABox also remains satisfied, since in \( T' \) all domain elements interpreting some ABox individual remain unchanged, and they participate in the same concepts and roles as in \( T \). Finally, since every query match in \( T' \) would also be a query match in \( T \), \( T \models q \) implies \( T' \models q \).

Now we present some simple reductions to rewrite a KB \( K = \langle T, A \rangle \) into a normal form in which the TBox contains only CIs, negation occurs only at the atomic level, and \( T, \bot, B, T \) do not occur.

1. **ABox reduction.** We transform \( A \) into an extensionally reduced ABox, i.e., an ABox in which only concept and role names are used:
   - We replace each assertion \( C(a) \), where \( C \) is not a concept name, by \( AC(a) \) for a fresh \( AC \in C \), and we add to \( T \) the CIA \( AC \sqsubseteq C \).
   - We replace each assertion \( S(a, b) \), where \( S \) is not a role name, by \( p_S(a, b) \) for a fresh \( p_S \in R \), and we add to \( T \) the BRIA \( p_S \sqsubseteq S \).

2. **Elimination of T.** Assume that \( a_1, \ldots, a_n \) are the individuals in \( I_K \). Using a fresh role name \( p_U \), we add to \( A \) assertions \( p_U(a_i, a_{i+1}) \), for all \( 1 \leq i < n \). We add to \( T \) the BRIA \( \bigcup_{R \in R_K} R \subseteq p_U \), and we replace in \( K \) each occurrence of \( T \) by the role \( p_U^* \).

3. **Elimination of \( \bot \) and B.** We replace in \( K \) the empty role box \( p_B \), and we add to \( T \) the CIA \( \top \sqsubseteq \forall p_B \bot \). We replace the concepts \( T \) and \( \bot \) respectively by fresh concept names \( A_T \) and \( A_\bot \), and we add to \( T \) the CIs \( A \sqcup \forall A \sqsubseteq A_T \) and \( A_T \sqsubseteq \neg A_\bot \), for an arbitrary concept name \( A \).

4. **BRIA elimination.** We replace each BRIA \( S \sqsubseteq T' \) in \( T' \) by the CIA \( \exists S \setminus T' \). \( T' \sqsubseteq \bot \) (cf. \([54]\)).

5. **Negation normal form.** Finally, we transform each concept and role occurring in \( K \) into negation normal form (NNF), i.e., where negation is pushed inwards until \( \neg \) is applied only to atomic concepts, and \( \setminus \) only to atomic roles. For this, we first replace each concept of the form \( \exists S. \text{Self} \) by a fresh concept name \( A_{3S.\text{Self}} \), and add to \( T \) the CIs \( A_{3S.\text{Self}} \sqsubseteq \exists S.\text{Self} \) and \( \exists S.\text{Self} \sqsubseteq A_{3S.\text{Self}} \). We also replace each \( S \setminus S' \) by \( S \setminus \neg S' \). Then we move \( \neg \) inwards over concepts and roles using the well-known equivalences:

\[
\neg(C \cap C') = \neg C \cup \neg C', \quad \neg(\leq n S.C) = \geq (n + 1) S.C, \quad \neg(\forall R.C) = \exists R.\neg C, \quad \neg(S \cup S') = \neg S \setminus \neg S'
\]

\[
\neg(C \cup C') = \neg C \cap \neg C', \quad \neg(\geq n S.C) = \leq (n - 1) S.C, \quad \neg(\exists R.C) = \forall R.\neg C, \quad \neg(S \cap S') = S \setminus \neg S'
\]

We apply these equivalences until \( C \in C \) for every concept of the form \( \neg C \), and \( R \in R \cup \{ p \setminus p \in R \} \) for every role of the form \( \neg R \). Afterwards we convert \( \neg \) back to \( \setminus \), replacing where necessary \( \neg S \) with \( p_U \setminus S \), where \( p_U \) is the role introduced in step 2.

It will be convenient to assume in what follows that also queries are normalized in such a way that they do not contain \( T, \bot, \top, \) or \( B \), and they are in NNF. Given a KB \( K = \langle T, A \rangle \), which has already been normalized, and a query \( q \), we normalize \( q \) with respect to \( T \) as follows. We replace:

- each occurrence of \( T \) by the concept \( A_T \) (introduced in step 3 of the normalization of \( K \)),
- each occurrence of \( \bot \) by the concept \( A_\bot \) (introduced in step 3 of the normalization of \( K \)),
- each occurrence of \( \top \) by the role \( p_U^* \) (introduced in step 2 of the normalization of \( K \)), and
- each occurrence of \( B \) by the role \( p_B \) (introduced in step 3 of the normalization of \( K \)).

Finally, all concepts and roles in \( q \) are rewritten into NNF as explained above.

**Definition 3.2** (Normalized knowledge base, normalized query). A ZIQ knowledge base \( K = \langle T, A \rangle \) is normalized, if (i) \( A \) is extensionally reduced, (ii) \( T \) contains only CIs, (iii) \( T, \bot, \top, \) and \( B \) do not occur in \( K \), and (iv) all concepts and roles in \( K \) are in NNF. A P2RPQ \( q \) is normalized, if \( T, \bot, \top, \) and \( B \) do not occur in \( q \) and all concepts and roles in \( q \) are in NNF.

Each of the steps in the KB and query normalizations is linear and preserves all the properties enforced by the preceding transformations. Since they also preserve query entailment, we obtain:

**Proposition 3.3.** Given a ZIQ KB \( K \) and a P2RPQ \( q \), it is possible to construct in time linear in \( ||K|| \) and \( ||q|| \) a normalized knowledge base \( K' \) and a normalized query \( q' \) such that \( K \models q \) iff \( K' \models q' \).

**Proof.** We normalize \( K \) using steps 1–5 above, and then normalize \( q \) with respect to \( K \) (using the same role names to simulate \( T, \bot, \top, \) and \( B \)). A simple inspection of the transformations 1 to 5 suffices to see that the properties enforced by the preceding transformations are always preserved. Hence it is easy to see that their application results in a normalized KB \( K' \). Similarly for the query \( q' \).

The normalization also preserves satisfiability and query entailment. In fact, step 4 results in a logically equivalent KB, while steps 1, 3, and 5 result in a model conservative extension of the original KB. Indeed, they introduce fresh concept and
role names, but every interpretation \( I \) can be extended to an interpretation \( I' \) by interpreting these names as \( A_C^I = \{ a^I \mid a \in I_C, a^I \in C^I \} \), \( p_B^I = \{ (a^I, b^I) \mid a, b \in I_C, (a^I, b^I) \in S^I \} \), \( p_B^I = \emptyset \), \( A^I = \Delta^I \), \( A^I = \emptyset \), and \( A^I_{\text{Self}} = (35.\text{Self})^I \). In this way we ensure that \( I' \models K' \) iff \( I \models K \), and \( I \) and \( I' \) coincide on all matches for \( q \). If we use the same \( A_T \), \( A_\bot \) and \( p_B \) to replace in \( q \) respectively \( T \), \( \bot \) and \( B \), then \( I \) and \( I' \) also coincide on all matches for the query \( q' \) in which \( T \), \( \bot \) and \( B \) have been eliminated. Step 2 results in an extension of \( K \) that preserves \( K \)-connected models. By interpreting \( p_U \) as \( p_U^I = \{ (a^I, a^I_{i+1}) \mid a_i, a_{i+1} \text{ occur in } A_T \cup \{d, d' \mid (d, d') \in P^I \text{ for some } p \in R^I \} \) we can obtain from each \( K \)-connected interpretation \( I \) a new \( I' \) such that \( (p_U^I)^I = \Delta^I_T \times \Delta^I_T \), hence \( I' = \models K' \) iff \( I \models K \), and \( I \) and \( I' \) coincide on all matches for \( q \). Similarly as above, they also coincide on matches for the query \( q' \) in which \( T \) has been replaced by \( p_U^I \). Whenever \( K \not\models q \), there is a \( K \)-connected model \( I \) of the original \( K \) such that \( I \not\models q \) (cf. Lemma 3.1), and hence also a model \( I' \) of the rewritten \( K' \) such that \( I' \not\models q \). This shows that the normalization preserves query entailment.

Finally, to show that the transformation is linear, it suffices to observe that each step can be executed in linear time. Steps 1 and 4 only replace a linear number of expressions, and they replace them by expressions of the same size plus a new constant number of CIAs, and step 2 adds a linear number of assertions to \( S \). Hence they can be applied in linear time. Steps 2 and 3 replace a linear number of expressions by a new linear number of CIAs, and is also Fischer–Ladner closed in the style of a similarly defined closure for PDL [55]. The notation extends naturally to sets of concepts.

We define here the closure for the DL \( \text{ALCQI} \text{B}^{\text{Self}} \), which is similar to \( ZIQ \), but instead of role difference \( S \setminus S' \), it has negation \( \neg S \) as a simple role constructor. Semantically, \( \neg S^I = (\Delta^I \times \Delta^I) \setminus S^I \), hence \( S \setminus S' \) can be expressed as \( S \cap \neg S' \). We call an \( \text{ALCQI} \text{B}^{\text{Self}} \) expression \textit{safe}, if it is equivalent to an expression in \( ZIQ \); unsafe Boolean roles are convenient for a simple definition of syntactic closure.

In what follows, \( \sim E \) denotes the NNF of \( \neg E \), for every concept or simple role \( E \). The symbol \( \exists \) is generic for \( \geq \) and \( \leq \); \( Q \) for \( \forall \) and \( \exists \); \( \text{not} \) for \( \cap \) and \( \cup \); and \( \text{no} \) for \( \cap \) and \( \cup \). For a role name \( p \in R \), the \textit{inverse of} \( p \) is \( p^- \) and that of \( p^- \) is \( p \). The inverse of an atomic role \( p \) is denoted by \( \text{inv}(p) \). For a simple role \( S \), \( \text{inv}(S) \) denotes the role obtained by replacing each atomic role \( p \) occurring in \( S \) by its inverse \( \text{inv}(p) \). As usual, \( C \) and \( C' \) stand for concepts, \( S \) and \( S' \) for simple roles, and \( R \) and \( R' \) for arbitrary roles.

**Definition 3.4.** The closure \( \text{Cl}(D) \) of an \( \text{ALCQI} \text{B}^{\text{Self}} \) concept \( D \) is the smallest set that contains the NNF of \( D \) and is closed under the rules of Table 1. For a set \( D \) of concepts, \( \text{Cl}(D) = \bigcup_{D \in D} \text{Cl}(D) \). □

It is easy to see that \(|\text{Cl}(D)| \) (resp., \(|\text{Cl}(D)|\)) is linear in the length of \( D \) (resp., \( D \)). Note that \( \text{Cl}(D) \) may contain unsafe expressions even if \( D \) is a \( ZIQ \) concept. For example, the concept \( D = 3 \text{HasFather.male} \) (which is in negation normal form) has \( \neg \text{HasFather} \) (\( = T \setminus \text{HasFather} \)) in \( \text{Cl}(D) \), which is not expressible in \( ZIQ \).

### 3.3. Canonical model property

Like many DLs, \( ZIQ \) enjoys some form of forest model property. In fact, every satisfiable concept \( C \) has a model that can be seen as a tree, possibly having loops at some nodes. This extends to TBoxes. For knowledge bases, we must suitably extend tree-shaped models to forest-shaped models.\(^5\)

First, we observe that in \( ZIQ \) every TBox \( T \) can be \textit{internalized} into a concept \( C_T \), such that the satisfiability of \( T \) can be established by obtaining a model of \( C_T \) [57].

\(^{5}\) Unlike the results of the previous subsection, these results do not hold for \( \text{ALCQI} \text{B}^{\text{Self}} \); see [56] for discussion.
Definition 3.5 (TBox internalization, $C_T$). Given a normalized $\mathcal{ZIQ}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we let $C_T$ be the negation normal form of the concept

$$\forall p_U^* \cdot \bigcap_{C_1 \subseteq C_2 \in \mathcal{T}} \neg (C_1 \cup C_2),$$

where $p_U$ is the role introduced in step 2 of the normalization of $\mathcal{K}$, or if this $p_U$ is not present in $\mathcal{K}$, then $p_U = \bigcup_{P \in \mathcal{K}} P$. □

If $C_T$ holds everywhere in an interpretation $\mathcal{I}$, then $\mathcal{I} \vDash \mathcal{T}$. Furthermore, if a domain element satisfies $C_T$, then the same holds for every element that is connected to it in $\mathcal{I}$.

Proposition 3.6. Let $\mathcal{I}$ be an interpretation and let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a normalized $\mathcal{ZIQ}$ KB. If for each element $d \in \Delta^I$ there is some $d_0 \in C_T^\mathcal{I}$ such that $d$ is $K$-connected to $d_0$, then $\mathcal{I} \vDash \mathcal{T}$.

Proof. Suppose that $d_0 \in C_T^\mathcal{I}$. Then for every sequence $d_0, \ldots, d_n$, $n \geq 0$, such that for each $0 \leq i < n$ we have $(d_i, d_{i+1}) \in p^I$ for some $P \in \mathcal{K}$, it holds that $d_n \in \bigcap_{C_1 \subseteq C_2 \in \mathcal{T}} (\neg C_1 \cup C_2)^I$. By $K$-connectedness, for every element $d \in \Delta^I$ some sequence of this form exists such that $d = d_n$, so it follows that $\bigcap_{C_1 \subseteq C_2 \in \mathcal{T}} (\neg C_1 \cup C_2)^I = \Delta^I$; that is, $\mathcal{I} \vDash C_1 \subseteq C_2$ for each $C_1 \subseteq C_2 \in \mathcal{T}$, which means $\mathcal{I} \vDash \mathcal{T}$. □

Now we define a canonical model of a normalized $\mathcal{ZIQ}$ KB, which has a certain forest-shaped structure that allows us to recognize it using tree automata.

Definition 3.7. For $k \geq 0$, an interpretation $\mathcal{I} = (\Delta^I, \mathcal{T})$ for a $\mathcal{ZIQ}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is $k$-canonical, if there exists a non-empty finite set $\text{Roots}(\mathcal{I}) \subseteq \Delta^I$ such that:

1. $\{e\} \cup \Delta^I$ is a tree. (Note that this implies $\Delta^I \subseteq \mathbb{N}^\ast$.)
2. $\text{Roots}(\mathcal{I}) = \{a^I | a \in \mathcal{I}_k\} \subseteq \mathbb{N}$.
3. Each element of $\Delta^I$ is of the form $i \cdot x$ with $i \in \text{Roots}(\mathcal{I})$ and $x \in \{1, \ldots, k\}^\ast$.
4. For each pair $x, y \in \Delta^I$ with $y$ the form $i \cdot x$, there exists some atomic role $P \in \mathcal{K}$ such that $(x, y) \in p^I$.
5. If $(x, y) \in p^I$ for some role name $p$ and some $x, y \in \Delta^I$, then either (a) $x, y \in \text{Roots}(\mathcal{I})$, or (b) for some $i \in \text{Roots}(\mathcal{I})$, $x$ is of form $i \cdot w$, $y$ of form $i \cdot w'$, and either $w = w'$, or $w'$ is a successor of $w$, or $w'$ is the predecessor of $w$.

The elements of $\text{Roots}(\mathcal{I})$ are called the roots of $\mathcal{I}$. □

Since every node in a canonical interpretation is $K$-connected to a root, by Proposition 3.6, satisfaction of $C_T$ at the roots ensures satisfaction of $\mathcal{T}$.

Proposition 3.8. Let $\mathcal{I}$ be a $k$-canonical interpretation for a normalized $\mathcal{ZIQ}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, for some $k \geq 0$. Then $\mathcal{I} \vDash \mathcal{T}$ iff $\text{Roots}(\mathcal{I}) \subseteq C_T^\mathcal{I}$.

Proof. (Only If) Suppose $\mathcal{I} \vDash \mathcal{T}$. Then for every element $x \in \Delta^I$ and every $C_1 \subseteq C_2 \in \mathcal{T}$ we have $x \in (\neg C_1 \cup C_2)^I$. It follows that $\bigcap_{C_1 \subseteq C_2 \in \mathcal{T}} (\neg C_1 \cup C_2)^I = \Delta^I$, which implies $C_T^\mathcal{I} = \Delta^I$; hence $\text{Roots}(\mathcal{I}) \subseteq C_T^\mathcal{I}$.

(If) Suppose $\text{Roots}(\mathcal{I}) \subseteq C_T^\mathcal{I}$. Observe that, as $\mathcal{I}$ is a $k$-canonical interpretation, it is $K$-connected by definition. That is, for every $x \in \Delta^I$, there is some $a \in \mathcal{I}_k$ such that $x$ is $K$-connected to $a^I$ (by conditions (2)–(4) in Definition 3.7). Moreover, $a^I \in \text{Roots}(\mathcal{I})$ (by condition (2)), hence $a^I \in C_T^\mathcal{I}$. Then $\mathcal{I} \vDash \mathcal{T}$ follows from Proposition 3.6. □

Now we can establish the canonical model property of $\mathcal{ZIQ}$, by a straightforward adaptation of a similar property of related logics [40,45]. It states that, to decide query entailment, it suffices to consider the $k$-canonical models of the given KB $\mathcal{K}$. Here the branching factor $k$ depends on the size of the closure of all concepts and roles in $\mathcal{T}$ and $\mathcal{A}$, and on the number restrictions and existential concepts that occur therein.

Definition 3.9 (Branching). For a set $D$ of concepts, let $br(D) =$ $|\text{Cl}(D)| \cdot n_{\text{max}}$, where $n_{\text{max}} = \max\{|n | \geq n.S.C \in \text{Cl}(D) \cup \{0\}\}$.

Given a normalized $\mathcal{ZIQ}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a normalized P2RPQ $q$, we define $k_T = \text{br}([C_T])$ and $k_T.q = \text{br}([C_T] \cup D_q)$, where $D_q =$ $|C | C(v) \in \text{At}(q)) \cup \{\exists R.A | R(v, v') \in \text{At}(q))$ for an arbitrary concept name $A$.

Note that, by definition of $\text{Cl}(D)$, if there is some concept $\exists R.C \in \text{Cl}(D)$, then there is also some concept $\exists 15.C \in \text{Cl}(D)$. Hence $\text{br}(D) = 0$ only if there are no existential and universal concepts, and no number restrictions in $\text{Cl}(D)$.
**Theorem 3.10 (Canonical model property).** Let $K = \langle T, A \rangle$ be a normalized $\mathcal{ZITQ}$ KB, and let $q$ be a normalized $P2RPQ$. If $K \models q$, then there exists some $k_{T,q}$-canonical model $I$ of $K$ such that $I \models q$. Furthermore, if $K$ is satisfiable, then it has a $k_T$-canonical model.

**Proof (Sketch).** Following [43,41], with minor adaptations to properly handle ABoxes, Boolean role constructs and Self, one can show that every model $I$ of $K$ that admits no match for $q$ can be unraveled into a $k_{T,q}$-canonical model $I'$ that also admits no match. We refer to Appendix A for details. The second part of the claim follows from the fact that KB satisfiability reduces to query non-entailment using a simple query $q$ such that $br(D_q) = 0$. \( \square \)

We remark that the reason why the branching degree of the counterexample canonical model depends on $q$ is that we allow for complex concepts in the query. In fact, given a KB $K = \langle T, A \rangle$, then for every $P2RPQ q$ where only concept names (and arbitrary roles) occur, if $K \models q$, then there is some $k_{T,q}$-canonical model $I$ of $K$ such that $I \models q$.

By **Theorem 3.10**, we can restrict to $k_T$-canonical (resp., $k_{T,q}$-canonical) forest-shaped models for deciding KB satisfiability (resp., query entailment). To solve these problems using tree automata, we represent canonical interpretations as infinite labeled trees, as described in the next sections.

4. Deciding KB satisfiability via automata

The lack of tree-shaped models complicates a straight use of tree automata for KB reasoning, and adaptations are needed to exploit the weaker canonical model property of Section 3.3. An example of such an adaptation is the pre-completion technique [45], in which after a reasoning step on the ABox, automata are used to verify the existence of a tree-shaped model of the TBox rooted at each ABox individual. We follow a different approach, introduced in [1], namely to represent forest-shaped canonical interpretations as trees, and to encode $K$ into an automaton $A_K$ that accepts exactly the set of trees that represent canonical models of $K$. To the best of our knowledge, this is the first approach handling ABox assertions and individuals directly in the automaton; importantly, the resulting automata-based algorithm can be extended to query answering, which we do in Section 5.

4.1. Representing canonical models as trees

In the following, let $K = \langle T, A \rangle$ be a normalized $\mathcal{ZITQ}$ KB, and let $b_K = \max(k_T, |I_K|)$, where $k_T = br(\{C_T\})$ is as in **Definition 3.9**. To represent interpretations for $K$, we define interpretation trees, which are labeled $b_K$-ary trees. Each node is labeled with a (possibly empty) set of atomic concepts and roles, and special symbols $p_{ij}$ (used to indicate that the pair $(i, j)$ of roots is in the extension of the role $p$) and $p_{Self}$ (used to indicate that a pair $(x, x)$ is in the extension of $p$). The label of the root $\epsilon$ contains the special identifier $r$, and its children may contain individual names from $I_K$ in their labels; if the latter holds we call them individual nodes.

**Definition 4.1.** Given $K$, let $PL_K = \{p_{ij} \mid p \in R_K, i, j \in [1, \ldots, b_K]\}$, and $PS_K = \{p_{Self} \mid p \in R_K\}$. An interpretation tree for $K$ is a labeled $b_K$-ary tree $T = (T, L)$ over the alphabet $\Sigma_K = 2^{C_K \cup R_K \cup \{r\} \cup PL_K \cup PS_K}$ such that:

1. $r \in L(\epsilon)$, and $r \notin L(x)$ for every node $x \in T$ with $x \neq \epsilon$.
2. For every $a \in I_K$ there is exactly one node $x \in T$ with $1 \leq x \leq b_K$ and $a \in L(x)$.
3. $I_K \cap L(x:j) = \emptyset$ for every node $x \neq \epsilon$ and $j > 0$ such that $x \cdot j \in T$. \( \square \)

4.1.1. From canonical interpretations to trees

For a canonical interpretation $I$, we now define its tree representation $T_I$, which informally is built as follows. Since the domain of $I$ is always contained in a $b_K$-ary tree, we only need to add a root $\epsilon$ and enough `dummy' nodes to ensure the correct branching.

The interpretations of individuals, concepts, and roles are represented using node labels from the alphabet $\Sigma_K$. Roughly speaking, each element $x \in A^T$ is labeled with a set $L(x)$ that contains (i) the atomic concepts $A$ such that $x \in A^T$; (ii) the atomic roles $P$ connecting the predecessor of $x$ to $x$, and (iii) the special symbol $p_{Self}$ for each role name $p$ such that $(x, x) \in p^T$. The label $L(i)$ of each root $i$ of $I$ contains the names of the individuals in $I_K$ it interprets, and the atomic concepts to which $i$ belongs, but it does not contain basic roles; the relations $p$ between individual nodes are stored in the root label $L(\epsilon)$ via symbols $p_{ij}$. Formally:

**Definition 4.2.** Let $I$ be a canonical interpretation for $K$ with $n$ roots. The tree representation of $I$ is the interpretation tree $T_I = (T, L)$ where:

- $T = \{1, \ldots, b_K\}^*$,
from trees to canonical interpretations

With each interpretation tree T, we can in turn associate a canonical interpretation I_T. Informally, its domain Δ_T is given by (i) the set I of all the nodes x in T having some individual a in their label L(x), and (ii) the nodes in T reachable from any such x through the roles in K. Note that each node with an empty label and all its descendants are not included in the interpretation I_T.

The extensions of individuals, concepts, and roles are determined by the node labels in T. We build the extension of each role name p as the union of two sets of pairs \( R_p^1 \) and \( R_p^2 \), where \( R_p^1 \) contains the pairs (x, y) of p-neighbors x, y in the tree that are not both individual nodes, and \( R_p^2 \) contains the pairs i, j of individual nodes related by p (which is represented by the special label \( p_{ij} \) at the root). Formally:

- \( L(ε) = \{ r \} \cup \{ p_{ij} \mid i, j ∈ Roots(T) \}, p ∈ R_K, \) and \((i, j) ∈ p^T)\)
- for each \( 1 ≤ x ≤ b_K \), \( L(x) = \{ a ∈ I_K \mid a^T = x \} \cup \{ A ∈ C_K \mid x ∈ A^T \}\)
- for all other nodes x of T (i.e., those with \(|x| > 1\)\), \( L(x) = \{ A ∈ C_K \mid x ∈ A^T \} \cup \{ p ∈ R_K \mid (x · −1, x) ∈ p^T \} \cup \{ p_{Self} \mid p ∈ R_K \) and \((x, x) ∈ p^T)\) □

Note that \( L(x) = \emptyset \) for every dummy node \( x ∈ T \setminus (Δ_T \cup \{ ε \}) \) that does not represent a domain element.

**Example 4.3.** The left part of Fig. 2 depicts a fragment of an infinite canonical model I_δ of K_δ. Its roots are 1 = Zeus^I_δ, 2 = Heracles^I_δ, 3 = Alcmena^I_δ, 4 = Eleckyron^I_δ, and 5 = Perseus^I_δ. They are depicted as large dots, and each of them is labeled with the name of the individual it interprets, and with the concept names from C_K to whose interpretation it belongs. Other domain elements are represented by smaller dots, and are also labeled with the concept names to whose interpretation they belong. For readability, we use the following label names: \( L_1 = \{ male, deity \}, L_2 = \{ female, deity \}, L_3 = \{ female, mortal \}, L_4 = \{ male \}, \) and \( L_5 = \{ female \}\). Every non-root element has two successors, which are the fullfillers of the HasMother and HasFather relations, respectively. The HasFather relation is represented by solid arrows, while HasMother is represented by dashed arrows. HasParent is the union of HasMother and HasFather and is not depicted explicitly.

The right part of the figure depicts the tree representation of I_δ. For readability, we use only the initial letter of each individual name in the labels. The label of the root is LR = \{r, HasFather\_2, HasFather\_34, HasFather\_45, HasFather\_51, HasMother\_23, HasParent\_21, HasParent\_23\_45, HasParent\_51, HasParent\_23\}. The labels of the level 1 nodes are given explicitly in the figure, while for the other nodes we use the labels \( L_1' = L_1 \cup \{ HasFather, HasParent \}, L_2' = L_2 \cup \{ HasMother, HasParent \}, L_3' = L_3 \cup \{ HasMother, HasParent \}, L_4' = L_4 \cup \{ HasFather, HasParent \}, \) and \( L_5' = L_5 \cup \{ HasMother, HasParent \} \). □

4.1.2. From trees to canonical interpretations

- \( L(ε) = \{ r \} \cup \{ p_{ij} \mid i, j ∈ Roots(T) \}, p ∈ R_K, \) and \((i, j) ∈ p^T)\)
- for each \( 1 ≤ x ≤ b_K \), \( L(x) = \{ a ∈ I_K \mid a^T = x \} \cup \{ A ∈ C_K \mid x ∈ A^T \}\)
- for all other nodes x of T (i.e., those with \(|x| > 1\)\), \( L(x) = \{ A ∈ C_K \mid x ∈ A^T \} \cup \{ p ∈ R_K \mid (x · −1, x) ∈ p^T \} \cup \{ p_{Self} \mid p ∈ R_K \) and \((x, x) ∈ p^T)\) □

Note that \( L(x) = \emptyset \) for every dummy node \( x ∈ T \setminus (Δ_T \cup \{ ε \}) \) that does not represent a domain element.

**Example 4.3.** The left part of Fig. 2 depicts a fragment of an infinite canonical model I_δ of K_δ. Its roots are 1 = Zeus^I_δ, 2 = Heracles^I_δ, 3 = Alcmena^I_δ, 4 = Eleckyron^I_δ, and 5 = Perseus^I_δ. They are depicted as large dots, and each of them is labeled with the name of the individual it interprets, and with the concept names from C_K to whose interpretation it belongs. Other domain elements are represented by smaller dots, and are also labeled with the concept names to whose interpretation they belong. For readability, we use the following label names: \( L_1 = \{ male, deity \}, L_2 = \{ female, deity \}, L_3 = \{ female, mortal \}, L_4 = \{ male \}, \) and \( L_5 = \{ female \}\). Every non-root element has two successors, which are the fullfillers of the HasMother and HasFather relations, respectively. The HasFather relation is represented by solid arrows, while HasMother is represented by dashed arrows. HasParent is the union of HasMother and HasFather and is not depicted explicitly.

The right part of the figure depicts the tree representation of I_δ. For readability, we use only the initial letter of each individual name in the labels. The label of the root is LR = \{r, HasFather\_2, HasFather\_34, HasFather\_45, HasFather\_51, HasMother\_23, HasParent\_21, HasParent\_23\_45, HasParent\_51, HasParent\_23\}. The labels of the level 1 nodes are given explicitly in the figure, while for the other nodes we use the labels \( L_1' = L_1 \cup \{ HasFather, HasParent \}, L_2' = L_2 \cup \{ HasMother, HasParent \}, L_3' = L_3 \cup \{ HasMother, HasParent \}, L_4' = L_4 \cup \{ HasFather, HasParent \}, \) and \( L_5' = L_5 \cup \{ HasMother, HasParent \} \). □

4.1.2. From trees to canonical interpretations

With each interpretation tree T, we can in turn associate a canonical interpretation I_T. Informally, its domain Δ_T is given by (i) the set I of all the nodes x in T having some individual a in their label L(x), and (ii) the nodes in T reachable from any such x through the roles in K. Note that each node with an empty label and all its descendants are not included in the interpretation I_T.

The extensions of individuals, concepts, and roles are determined by the node labels in T. We build the extension of each role name p as the union of two sets of pairs \( R_p^1 \) and \( R_p^2 \), where \( R_p^1 \) contains the pairs (x, y) of p-neighbors x, y in the tree that are not both individual nodes, and \( R_p^2 \) contains the pairs i, j of individual nodes related by p (which is represented by the special label \( p_{ij} \) at the root). Formally:

- \( L(ε) = \{ r \} \cup \{ p_{ij} \mid i, j ∈ Roots(T) \}, p ∈ R_K, \) and \((i, j) ∈ p^T)\)
- for each \( 1 ≤ x ≤ b_K \), \( L(x) = \{ a ∈ I_K \mid a^T = x \} \cup \{ A ∈ C_K \mid x ∈ A^T \}\)
- for all other nodes x of T (i.e., those with \(|x| > 1\)\), \( L(x) = \{ A ∈ C_K \mid x ∈ A^T \} \cup \{ p ∈ R_K \mid (x · −1, x) ∈ p^T \} \cup \{ p_{Self} \mid p ∈ R_K \) and \((x, x) ∈ p^T)\) □
Definition 4.4. Let $T = (T, L)$ be an interpretation tree. For each role name $p \in R_K$, we define:

$$R^1_p = \{(x, x' \mid p \in L(x \cdot i) \cup \{x \cdot i, x \mid \text{Inv}(p) \in L(x \cdot i) \cup \{x, x \mid p_{\text{Self}} \in L(x)\}$$
$$R^2_p = \{(i, j) \mid p_{\text{Self}} \in L(\varepsilon)\}$$

Then we let $I_T = \{i \in \{1, \ldots, b_K\} \mid a \in L(i) \text{ for some } a \in I_K\}$ and, for each $i \in I_T$,

$$D_i = \\left\{x' : (i, x') \in \left( \bigcup_{p \in R_K} (R^1_p \cup (R^1_p)^{-}) \right)^* \right\}$$

where $(R^1_p)^{-}$ denotes the inverse of relation $R^1_p$. The interpretation $I_T$ represented by $T$ is defined by:

$$\Delta^{IT} = I_T \cup \bigcup_{i \in I_T} D_i \quad \text{and}$$
$$a^{IT} = x \in I_T \quad \text{such that } a \in L(x), \text{ for each } a \in I_K$$
$$A^{IT} = \left\{x \in \Delta^{IT} \mid A \in L(x) \Delta^{IT} \cap \{x \mid A \in L(x)\} \right\} \text{ for each concept name } A \in C_K,$$
$$p^{IT} = (\Delta^{IT} \times \Delta^{IT}) \cap (R^1_p \cup R^2_p) \text{ for each role name } p \in R_K$$

Note that, by condition (12) in Definition 4.1, for each $a \in I_K$ there is exactly one $x$ such that $a^{IT} = x$. The set $I_T$ contains the roots of $I_T$, and $\{\varepsilon\} \cup \Delta^{IT}$ is a $b_K$-tree. Note also that $i$ is a root of $I_T$ iff $I_K \cap L(i) \neq \emptyset$.

Lemma 4.5. If $T$ is an interpretation tree, then $I_T$ is a canonical interpretation, and if $\mathcal{I}$ is a canonical interpretation, then $\mathcal{I} = I_{T_{\mathcal{I}}}$.

Proof. It is not hard to verify that $I_T$ satisfies the conditions (1)–(5) of Definition 3.7 (for $k = b_K$):

1. $\{\varepsilon\} \cup \Delta^{IT}$ is a tree (note that $\varepsilon$ is not in $\Delta^{IT}$);
2. Roots$(I_T) = \{a^{IT} \mid a \in I_K\} \subseteq \{1, \ldots, b_K\} \subseteq \mathbb{N}$;
3. Each element of $\Delta^{IT}$ is of the form $i \cdot x$ with $i \in \text{Roots}(I_T)$ and $x \in \{1, \ldots, b_K\}^*$;
4. For every pair $x, y \in \Delta^{IT} \cup \{x \mid A \in L(x)\}$ there exists some atomic role $P$ such that $(x, y) \in P^{IT}$;
5. If $(x, y) \in P^{IT}$ for some role name $p$ and some $x, y \in \Delta^{IT}$, then $(a)$ $x, y \in \text{Roots}(I_T)$, or $(b)$ for some $i \in \text{Roots}(I_T)$, $x$ is of form $i \cdot w$, $y$ of form $i \cdot w'$, and either $w = w'$ or $w'$ is a successor of $w$, or $w'$ is the predecessor of $w$.

Thus $I_T$ is $b_K$-canonical. As for the second part, the domain of $I_{T_{\mathcal{I}}}$ coincides with $\Delta^{IT}$: $\Delta^{IT}$ is connected and hence in the construction of $I_{T_{\mathcal{I}}}$ from $T_{\mathcal{I}}$, the set $I_{T_{\mathcal{I}}}$ coincides with Roots$(I_T)$ and $I_{T_{\mathcal{I}}} \cup \bigcup_{i \in I_{T_{\mathcal{I}}}} D_i$ contains all elements of $\Delta^{IT}$ (and no others). Furthermore, by construction $a^{IT} = a^{IT_{\mathcal{I}}}$ for each $a \in I_K$ and for each $x, y \in I_{T_{\mathcal{I}}}$ and concept name $A$, we can verify that $x \in A^{IT_{\mathcal{I}}}$ iff $x \in A^{IT}$ and for each $x, y \in I_{T_{\mathcal{I}}}$ and role name $p$, that $(x, y) \in p^{IT_{\mathcal{I}}}$ iff $(x, y) \in p^{IT}$. Hence $\mathcal{I} = I_{T_{\mathcal{I}}}$.

4.2. Constructing the automaton to verify KB satisfaction

In this section, we show how to construct from a normalized $\mathcal{I}$ $Q$ KB $\mathcal{K}$ an automaton $A_{\mathcal{K}}$ that accepts the $\Sigma_K$-labeled trees that are tree representations of canonical models of $\mathcal{K}$. We thus can decide the satisfiability of $\mathcal{K}$ by testing $A_{\mathcal{K}}$ for emptiness.

To simplify the technical details, we construct $A_{\mathcal{K}}$ in four steps: (1) We construct from $\mathcal{K}$ a 2ATA $A_T$ that accepts a given tree $T$ iff it is an interpretation tree for $\mathcal{K}$. (2) We construct another 2ATA $A_A$ that accepts $T$ iff the represented interpretation satisfied all assertions in $\mathcal{A}$. (3) We construct a third 2ATA $A_T$ which verifies whether each individual in $I_K$ satisfies $C_T$. (4) Finally, by intersecting the three 2ATAs, we obtain $A_{\mathcal{K}}$. We note that all these automata, which are summarized in Table 2, run over $b_K$-ary trees labeled with the alphabet $\Sigma_K$ from Definition 4.1. As we will see in the next section, their size is polynomial in $\mathcal{K}$.

4.2.1. Automaton $A_T$ verifying interpretation trees

We start with the automaton $A_T$ that verifies whether an input tree is an interpretation tree, that is, whether it correctly represents an interpretation that is canonical for $\mathcal{K}$.

Definition 4.6. Let $A_T = (b_K, \Sigma_K, Q_T, \delta_T, s^T_0, F_T)$, where:

- The set of states is $Q_T = \{s^T_0, s_1, r, \neg r\} \cup I_K \cup \{-s \mid s \in I_K\}$.
- The transition function $\delta_T : Q_T \times \Sigma_K \rightarrow B(\{b_K\} \times Q_T)$ contains the following transitions:
1. for each \( \sigma \in \Sigma_K \) with \( r \in \sigma \), a transition \( \delta_T(s_0^T, \sigma) = B_1 \land B_2 \land B_3 \), where

\[
B_1 \land \bigvee_{a \in I_K} \bigwedge_{1 \leq i \leq b_K} (i, a)
\]
\[
B_2 \land \bigwedge_{1 \leq i < j \leq b_K} \bigwedge_{a \in I_K} ((i, \neg a) \lor (j, \neg a))
\]
\[
B_3 \land \bigwedge_{1 \leq i \leq b_K} ((i, \neg r) \land (i, s_1))
\]

and for each \( \sigma \in \Sigma_K \), a transition \( \delta_T(s_1, \sigma) = \bigwedge_{1 \leq i \leq b_K} ((i, \neg r) \land (\bigwedge_{a \in I_K} (i, \neg a)) \land (i, s_1)) \);

2. for each \( \sigma \in \Sigma_K \) and each \( s \in I_K \cup \{r\} \), transitions

\[
\delta_T(s, \sigma) = \begin{cases} 
\text{true,} & \text{if } s \in \sigma \\
\text{false,} & \text{if } s \notin \sigma \\
\end{cases}
\]

The acceptance condition is \( F_T = (\emptyset, Q_T) \). □

The transitions in item 2 simply verify whether the label of a node reached in state \( s \) contains the symbol \( s \). Overall, the definition of \( \delta_T \) ensures that every tree accepted by \( A_T \) satisfies conditions (t1)-(t3) in Definition 4.1:

- \( B_1 \) verifies that the label identifying each individual \( a \) occurs in some node of the first level.
- \( B_2 \) verifies that a label identifying an individual does not occur in two different level 1 nodes.
- \( B_3 \) checks that the labels of the nodes of level 1 do not contain \( r \), and switches from such states to the state \( s_1 \). From \( s_1 \) it further checks that \( r \) and all \( a \in I_K \) do not occur anywhere below level 1 in the whole tree.

**Lemma 4.7.** \( L(A_T) = \{T \mid T \text{ is an interpretation tree for } K\} \).

**Proof.** \( A_T \) accepts an input \( \Sigma_K\)-labeled \( b_K \)-ary tree \( T = (T, L) \) iff there is an accepting run of \( A_T \) over \( T \). If \( T \) is an interpretation tree, by condition (t2) in Definition 4.1, there is exactly one node \( x \in T \) with \( 1 \leq x \leq b_K \) and \( a \in L(x) \), which we denote by \( x_a \). Then we can build an accepting run \( (T, r) \) as follows. The root is labeled \( r(\varepsilon) = (\varepsilon, s_0^T) \), and it has the following children:

1. For each \( a \in I_K \), there is a child \( c_a \) labeled \( (x_a, a) \), which is a leaf of \( T_r \) (i.e., it has no children). These \( c_a \) ensure that \( B_1 \) is satisfied at the root, and since \( a \in L(x_a) \) for each \( x_a \) and \( \delta_T(a, \sigma) = \text{true} \) whenever \( a \in \sigma \), each \( c_a \) already satisfies (R2) in Definition 2.8.

2. For each \( a \in I_K \) and each pair \( i, j \) with \( 1 \leq i < j \leq b_K \), if \( i \neq x_a \) then \( \varepsilon \) has a child \( c_{aij} \) labeled \( (i, \neg a) \), and if \( i = x_a \) then \( \varepsilon \) has a child \( c_{aij} \) labeled \( (j, \neg a) \). In the former case, \( i \neq x_a \) implies \( a \notin L(i) \), hence \( \delta_T(\neg a, L(i)) = \text{true} \) and (R2) in Definition 2.8 holds for \( c_{aij} \). In the latter case, \( i = x_a \) implies \( j \neq x_a \), so we also have \( a \notin L(j) \) and \( \delta_T(\neg a, L(j)) = \text{true} \) as required. Moreover, the children \( c_{aij} \) ensure that \( \varepsilon \) satisfies \( B_2 \).

3. Finally, to satisfy \( B_3 \), \( \varepsilon \) has for each \( 1 \leq i \leq b_K \) a child \( c_i \) labeled \( (i, \neg r) \) (which satisfies (R2) in Definition 2.8 as \( r \notin L(i) \) by condition (t1) and hence \( \delta_T(\neg r, L(i)) = \text{true} \)), and a child \( c_i^c \) labeled \( (i, s_1) \). To satisfy (R2) in Definition 2.8, each \( c_i^c \) has children as follows:

(a) a leaf \( c_{wij} \) labeled \( (w, j, \neg r) \) for each \( 1 \leq j \leq b_K \). Since \( r \notin L(w, j) \), \( \delta_T(\neg r, L(w, j)) = \text{true} \) as required.

(b) a leaf \( c_{wji} \) labeled \( (w, j, \neg a) \) for each \( 1 \leq j \leq b_K \) and \( a \in I_K \). Since \( a \notin L(w, j) \), \( \delta_T(\neg a, L(w, j)) = \text{true} \).

(c) a node \( c_{wij}^c \) labeled \( (w, j, s_1) \) for each \( 1 \leq j \leq b_K \), which in turn has children as described by items (a)-(c).

The only infinite paths in \( (T, r) \) are the sequences of nodes \( c_{wij}^c \) labeled \( (w, j, s_1) \). Since such a path visits \( s_1 \) infinitely often the acceptance condition is satisfied and the run is accepting.

Conversely if a run \( (T, r) \) accepts \( T \), we must have \( r \in L(\varepsilon) \), and to satisfy the formulas \( B_1, B_2 \) and \( B_3 \), \( \varepsilon \) must have children analogous to those described in items 1–3 above, which enforce conditions (t1)–(t3). Namely:
1. For each \( a \in I_T \), a child \( c_a \) labeled \((x, a)\) for some \( 1 \leq x \leq b_K \) with \( a \in L(x) \). This ensures part of (t2): that each \( a \) is in \( L(x) \) for some \( 1 \leq x \leq b_K \).

2. For each pair \( i, j \) with \( 1 \leq i < j \leq b_K \) and each \( a \in I_{K} \), a child \( c_{a[ij]} \) labeled \((x, ¬a)\) for some \( x \in [i, j] \) with \( a \notin L(x) \). This ensures that the \( x \) with \( a \in L(x) \) is unique, and together with item 1, it ensures (t2).

3. For \( 1 \leq i \leq b_K \) a child \( c_i \) labeled \((i, ¬r)\), which implies that \( r \notin L(i) \), and a child \( c'_{i} \) labeled \((i, s_1)\). Since each node \( y \) in \( T_r \) with label of the form \((w, s_1)\) must have a child \( c_{w[j]} \) with \( r(w-j, ¬r) \) for each \( 1 \leq j \leq b_K \), and a child \( c_{w[j]} \) labeled \((w, j, ¬a)\) for each \( 1 \leq j \leq b_K \) and \( a \in I_K \), it follows that \( L(w.j \cap (I_K \cup [r]) = \emptyset \). Moreover, \( y \) must also have a child with label \( r(w-j, s_1) \) that in turn has analogous successors, thus \( L(w' \cap (I_K \cup [r]) = \emptyset \) for all children \( w' \) of \( w-j \). Hence \( r \) and the symbols in \( I_K \) cannot occur in any label below the level 1 nodes. Thus (t1) and (t3) hold and \( T \) is an interpretation tree. ∎

4.2.2. Automaton \( A_\mathcal{A} \) verifying ABox satisfaction

Next, we define the automaton \( A_\mathcal{A} \) that verifies whether the interpretation represented by a given \( \Sigma_K \)-labeled \( b_K \)-ary tree \( T \) satisfies all assertions in \( \mathcal{A} \), assuming that \( T \) is an interpretation tree. In what follows, we use \( C_\mathcal{A} \) and \( R_\mathcal{A} \) to denote the sets of concept and role names occurring in \( \mathcal{A} \), respectively.

**Definition 4.8.** Let \( A_\mathcal{A} = (b_K, \Sigma_K, Q_{\mathcal{A}}, \delta_{\mathcal{A}}, s_0, F_{\mathcal{A}}) \), where:

- The set of states is \( Q_{\mathcal{A}} = \{ s_0 \} \cup I_{K} \cup \{ s \in I_K \} \cup C_{\mathcal{A}} \cup \{ p_{ij} \mid p \in R_{\mathcal{A}} \text{ and } i, j \in \{1, \ldots, b_K\} \} \).
- The transition function \( \delta_{\mathcal{A}} : \Sigma_K \times I_{K} \rightarrow B(b_K \times Q_{\mathcal{A}}) \) contains the following transitions:
  1. for each \( \sigma \in \Sigma_K \) with \( r \in \sigma \), a transition \( \delta_{\mathcal{A}}(s_0, \sigma) = B_4 \land B_5 \land B_6 \), where
    \[
    B_4 = \bigwedge_{a \neq b, e \in A, 1 \leq i \leq b_K} (¬\sigma) \lor (i, ¬b)
    \]
    \[
    B_5 = \bigvee_{A(a) \in A, 1 \leq i \leq b_K} (i, a) \land (i, A)
    \]
    \[
    B_6 = \bigvee_{p(a, b) \in A, 1 \leq i \leq b_K, 1 \leq j \leq b_K} (0, p_{ij}) \land (i, a) \land (j, b)
    \]
  2. for each \( \sigma \in \Sigma_K \) and each \( s \in I_K \), transitions
    \[
    \delta_{\mathcal{A}}(s, \sigma) = \begin{cases}
    \text{true}, & \text{if } s \in \sigma \\
    \text{false}, & \text{if } s \notin \sigma
    \end{cases}
    \]
    and for each \( \sigma \in \Sigma_K \) and each \( t \in C_{\mathcal{A}} \cup \{ p_{ij} \mid p \in R_{\mathcal{A}} \text{ and } i, j \in \{1, \ldots, b_K\} \} \), transitions
    \[
    \delta_{\mathcal{A}}(t, \sigma) = \begin{cases}
    \text{true}, & \text{if } t \in \sigma \\
    \text{false}, & \text{if } t \notin \sigma
    \end{cases}
    \]
- The acceptance condition is \( F_{\mathcal{A}} = (\emptyset, Q_{\mathcal{A}}) \). ∎

Similarly to \( A_\mathcal{T} \), the transitions in item 2 verify the presence of atomic symbols in the node labels. The rest of the transitions verify the following conditions, starting from the root of the input tree:

- \( B_4 \) checks, for each assertion \( a \neq b \) in \( \mathcal{A} \), that \( a \) and \( b \) do not occur both in the label of the same node.
- \( B_5 \) ensures that each assertion \( A(a) \) in \( \mathcal{A} \) is satisfied, by verifying that the node labeled \( a \) is also labeled with \( A \).
- \( B_6 \) ensures that each assertion \( p(a, b) \) is satisfied, by finding the individual nodes \( i \) and \( j \) that represent the individuals \( a \) and \( b \), respectively, and checking \( p_{ij} \) at the root.

Hence, assuming that \( T \) is an interpretation tree, the transition function verifies that all the ABox assertions are satisfied in the corresponding interpretation.

**Lemma 4.9.** If \( T \) is an interpretation tree for a KB \( K = (T, \mathcal{A}) \), then \( T \in L(A_\mathcal{A}) \iff \mathcal{T}_T \models \mathcal{A} \).

**Proof (Sketch).** The argument is similar to the one in the proof of Lemma 4.7. Again, by design of the formulas \( B_4-B_6 \), for an interpretation tree \( T \) that encodes an interpretation \( \mathcal{T}_T \) where \( \mathcal{A} \) is satisfied, we can find an accepting run \((T_r, r)\) over \( T \) (in fact, a finite such run). On the other hand, every accepting run \((T_r, r)\) over \( T \) must satisfy \( B_4-B_6 \) at the root \( e \), and hence by design of \( \delta_{\mathcal{T}} \), each assertion in \( \mathcal{A} \) must be satisfied by the interpretation \( \mathcal{T}_T \), i.e., \( \mathcal{T}_T \models \mathcal{A} \). ∎
4.2.3. Automaton

An automaton $A_T$ represented by an input tree $T$ is shown in Table 3.

Initialization

For each $\sigma \in \Sigma_C$ with $r \in \sigma$, we have: $\delta_T(\sigma, \sigma) = (\bot \cup S_{\text{Sat}}) (\bot \cup \{\sigma\})$. This will be verified by the 2ATA $A_T$ for the interpretation $I_T$ represented by an input tree $T$. The definition of $A_T$ is rather involved, given that $C_T$ might be a complex concept that is formed using many of the different constructors available in $IZ\tilde{Q}$.

Definition 4.10. Let $A_T = (B_K, \Sigma_C, Q_T, \delta_T, S_T, F_T)$, where:

- The set of states $Q_T$ is shown in Table 3.
- The transition function $\delta_T: Q_T \times \Sigma_C \rightarrow B([B_K] \times Q_T)$ is given by the following groups of transitions:
  - (I) initialization (Table 4),
  - (II) concept and role checking (Table 4),
  - (III) atomic checks (Table 4), and
  - (IV) number restriction checking (Table 5).
- The acceptance condition is $F_T = (\bot, [\forall R^+.C \lor \forall R^+.C \in C(T_C)]), Q_T$.

Informally, the states in $C_{\text{Sat}}$ are used to check whether the node satisfies the corresponding concept, resp. it is (not) a particular ABox individual. The states in $S_{\text{Sat}}$ are used to check whether a role connects a non-ABox individual with itself; the states in $Q_{A_{\text{role}}}$ are used to check whether a role connects two ABox individuals (i.e., level 1 nodes); and the states in $Q_{\text{num}}$ (resp., $Q_{A_{\text{role}}}$) serve for checking number restrictions of a non-ABox individual (resp., ABox individual) node.

To explain the state set in more detail, and the intuition behind the transition function of $A_T$, we describe informally a run on a given interpretation tree $T$.

1. Initialization. $A_T$ starts a run over $T$ at the root $e$, in state $s_0$, and reading some $\sigma \in \Sigma_C$ such that $r \in \sigma$. Then, by the initialization transitions in item I of Table 4, it moves to each successor $i$ and switches to the state $C_T$ if $i$ represents some ABox individual $a$ (which is the case if $i$ has some $a$ in its label); otherwise, no further steps from $i$ are made, as the label of $i$ is empty.
Table 5
Transitions of the automaton $A_T$, part 2: number restriction checking.

<table>
<thead>
<tr>
<th>(IV) Number restriction checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. For each $\not\exists S C$ in $C(S_T)$ and $\sigma \in \Sigma_C$, we have: $\delta_T (\not\exists S C, \sigma) = (0, (\not\exists S C, 0, 0))$</td>
</tr>
<tr>
<td>2. For each state $(\not\exists S C, i, j)$ in $Q_{num}$ and $\sigma \in \Sigma_C$, we have:</td>
</tr>
<tr>
<td>2.1. $\delta_T (\not\exists S C, i, j, ) = (((i + 1, \sim S) \vee (i + 1, \sim C)) \wedge (0, (\not\exists S C, i + 1, j))) \vee ((i + 1, S) \wedge (i + 1, C) \wedge (0, (\not\exists S C, i + 1, j + 1)))$</td>
</tr>
<tr>
<td>Additionally, if $\sigma \cap I_K = \emptyset$, we have:</td>
</tr>
<tr>
<td>2.2. $\delta_T (\not\exists S C, b_c k, j, ) = (((0, \sim S a b t) \wedge (0, \sim S a b t)) \wedge (0, (\not\exists S C, b_c k + 1, j))) \wedge (0, (\not\exists S C, b_c k + 1, j + 1))$</td>
</tr>
<tr>
<td>2.3. $\delta_T (\not\exists S C, b_c k + 1, j, ) = (((0, \sim I m w s) \wedge (0, \sim I m w s)) \wedge (0, (\not\exists S C, b_c k + 1, j) \wedge (0, (\not\exists S C, b_c k + 2, j + 1))))$</td>
</tr>
<tr>
<td>Otherwise, if $\sigma \cap I_K \neq \emptyset$, we have:</td>
</tr>
<tr>
<td>2.4. $\delta_T (\not\exists S C, b_c k, j, ) = \delta_T (\emptyset, (\not\exists S C, 0, j))$</td>
</tr>
<tr>
<td>For all states in $Q_{num}$ and all $\sigma \in \Sigma_C$ with $r \in \sigma$, we have:</td>
</tr>
<tr>
<td>2.5. $\delta_T (\not\exists S C, i, j, ) = \delta_T (\emptyset, (\not\exists S C, i, j)) \wedge (0, (\not\exists S C, i + 1, j)) \wedge (0, (\not\exists S C, i + 1, j + 1))$</td>
</tr>
<tr>
<td>where the counters $i$ and $j$ range over the following values:</td>
</tr>
<tr>
<td>$0 \leq i &lt; b_c k, {0 \leq j &lt; n, \text{if } \not\exists S C \geq 0 \leq j \leq n, \text{if } \not\exists S C &lt; 0 }$</td>
</tr>
<tr>
<td>3. For each $\sigma \in \Sigma_C$, we have:</td>
</tr>
<tr>
<td>$\delta_T (\not\exists S C, i, n), ) = true, \text{for } 0 \leq i &lt; b_c k + 2$</td>
</tr>
<tr>
<td>$\delta_T (\not\exists S C, b_c k + 2, j, ) = false, \text{for } 0 \leq j &lt; n - 1$</td>
</tr>
<tr>
<td>$\delta_T (\not\exists S C, b_c k + 1, j, ) = false, \text{for } 0 \leq j &lt; b_c k$</td>
</tr>
<tr>
<td>$\delta_T (\not\exists S C, b_c k + 1, j, ) = true, \text{for } 0 \leq j &lt; b_c k$</td>
</tr>
<tr>
<td>$\delta_T (\not\exists S C, b_c k + 1, j, ) = true, \text{for } 0 \leq j &lt; b_c k$</td>
</tr>
</tbody>
</table>
| II. Concept and role checking

Next, from state $C_T$ and each node representing an individual, $A_T$ recursively decomposes $C_T$ and navigates its formula tree, in order to establish its satisfaction. This recursive decomposition comprises the 'core' of the run. It uses the states in $C(S_T)$ and the transitions in the left column of item II in Table 4. The automaton moves to a state $C \in C(S_T)$ and a node $x$ in order to check whether $x$ represents an instance of $C$. Complex concepts and the non-simple roles occurring in them are decomposed, and $T$ is navigated accordingly. Please note that $A_T$ decomposes non-simple roles inside universal and existential restrictions until it reaches expressions of the form $\exists S C$ and $\forall S C$ with $S$ simple, then it verifies their satisfaction by moving to states $\geq 1 S C \leq 0 S C$, $\sim C$, respectively, and uses the transitions for number restrictions from Table 5 (which are explained below). Observe also that all transitions in item II of Table 4 move to states with strictly less complex expressions, except for the transitions that move from states of the form $\forall R.T$.C and $\exists R.T$.C to states of the form $\forall R.\forall R.T$.C and $\exists R.\exists R.T$.C. As we will discuss below, these are the only transitions that may cause infinite runs.

The transitions for number restrictions, which will be explained next, may move to a state $S$ corresponding to a simple role in $C(S_T)$, in order to verify whether $S$ holds between a node $x$ and its predecessor. To this aim, $S$ is also recursively decomposed using the first group of transitions in the right column of item II in Table 4 (note that $S$ and all its subroles are in $Q_{num}$ and thus are states of $A_T$). If $A_T$ must verify $S$ between individual nodes $i$ and $j$, it proceeds similarly but uses the second group of transitions and the states in $Q_{num}$. Finally, if the automaton must verify whether $S$ connects a node to itself (this is relevant for the satisfaction of the number restrictions, and of the concepts of the form $\exists S S e l f$), it uses the last group of transitions and the states in $Q_{sel}$.

III. Atomic checks

Modulo the generation of possibly infinite sequences of states containing $\exists R.T$.C and $\forall R.T$.C, the decomposition of concepts and roles (item (II)) always stops when $A_T$ reaches the 'atomic level' that comprises possibly negated atomic concepts and roles, special symbols in $P_{IK}$, and individual names $I_K$. These are checked locally at the node labels, using corresponding states, by the respective transitions given in Table 4.

IV. Number restriction checking

To verify the satisfaction of a number restriction, $A_T$ needs to navigate all nodes to which the current node can be connected via some role. We say that a node $y$ is a potential neighbor of a node $x \neq y$, if either (i) $y = x$, (ii) $y$ is a successor of $x$, (iii) $y \neq x$ is the predecessor of $x$, or (iv) both $x$ and $y$ are level one nodes. Note that, by definition, the potential neighbors include all successors of a node, also the 'dummy' nodes in $T$ that do not correspond to an object in $\Delta T$. For a level 1 node, its potential neighbors also include all the level 1 nodes, both actual individual nodes and 'dummy' nodes. We order the potential neighbors of a node $x$ as follows: the first $b_c k$ ones are its successors, in the order they occur in $T$. If $x$ is not a level 1 node, then the $(b_c k + 1)$-th potential neighbor of $x$ is $x$ itself, and the $(b_c k + 2)$-th is its predecessor. Otherwise, the $(b_c k + 1)$-th to $(2b_c k)$-th potential neighbors are the level 1 nodes, in the order they appear in $T$.

Moreover, for a simple role $S$ and a concept $C$, we say that a node $y$ is an $(S, C)$-neighbor of a node $x$, if $(x, y)$ is in $S^{\Delta T}$ and $y$ is in $C^{\Delta T}$. Note that for every $S$ and $C$, the $(S, C)$-neighbors of $x$ are contained in the potential neighbors of $x$ and, in contrast to the latter, the $(S, C)$-neighbors are necessarily elements of $\Delta T$. Moreover, we order the potential neighbors of a node $x$ as follows: the first $b_c k$ ones are its successors, in the order they occur in $T$. If there is no $a \in I_K$ such that $a^{\Delta T} = x$, then the $(b_c k + 1)$-th potential neighbor of $x$ is $x$ itself, and the $(b_c k + 2)$-th is its predecessor. Otherwise, the $(b_c k + 1)$-th to $(2b_c k)$-th potential neighbors are the level 1 nodes, in the order they appear in $T$. 


To verify that a node \( x \) satisfies a number restriction, the automaton traverses all potential neighbors of \( x \), and counts how many of them are actually its \((S, C)\)-neighbors. This requires us to encode counters using the auxiliary states in \( Q_{num} \) and \( Q_{Anum} \). Intuitively, in a state \((\geq nS.C, i, j)\) of \( Q_{num} \), the number \( i \) records how many potential neighbors have been navigated, and \( j \) how many of them are actually \((S, C)\)-neighbors. More precisely, when the automaton is in the state \((\geq nS.C)\) and visits a node \( x \), it changes to the state \((\geq nS.C, 0, 0)\) (item IV1 in Table 5) and then navigates the potential neighbors of \( x \). The automaton navigates first the at most \( k_T \) many successors of the node, using the transitions in item IV(2)1. While doing this, it increases the counters accordingly. It will be in state \((\geq nS.C, i, j)\), if exactly \( j \) among the first \( i - 1 \) potential neighbors of \( x \) are \((S, C)\)-neighbors of \( x \). Each transition of this kind has two disjunctions. The first disjunct covers the case when the \( i \)-successor is not an \((S, C)\)-neighbor of \( x \), hence only the \( j \)-counter is increased. The second disjunct covers the case where the \( i \)-successor is an \((S, C)\)-neighbor of \( x \) and increases both counters.

After all successors have been navigated, if \( x \) is a non-individual node, the automaton checks whether \( x \) is its own \((S, C)\)-neighbor with the transitions in item IV(2)2, and finally it checks the predecessor of \( x \) using the transitions in item IV(2)3. Otherwise, the current node \( x \) is an individual node, and the automaton must navigate all individual nodes at level 1 (including \( x \)) to count the \((S, C)\)-neighbors of the node. This is done by the transitions in items IV(2)4 and IV(2)5, which use the states in \( Q_{Anum} \) of the form \((a, \geq nQ.C, i, j)\). From an individual node \( \ell \), the automaton moves to some state \((a, \geq nQ.C, 0, j)\) for some \( a \) represented by \( \ell \) (item IV(2)4). From this state, it uses the counter \( i \) to navigate all the \( \ell \) nodes looking for \((S, C)\)-neighbors of \( \ell \), and stores in \( j \) how many of them it has found (item IV(2)5). Note that \( a \) can be any individual represented by \( \ell \), since we only use it to identify it. At each \( i \) the automaton verifies whether the root is labeled \( S_{\ell i} \), and \( i \) is labeled \( C \) and represents some individual \( b \). It increases the value of \( i \) and \( j \) if this is the case, and only the value of \( i \) otherwise.

In all the transitions of item IV2, the counters \( i \) and \( j \) range over the following values:

- \( 0 \leq i < b_K \): successors (resp., level 1 nodes) \( 1, 2, \ldots, i - 1 \) have been navigated;
- if \( i \) is \( > \), then \( 0 \leq j < n \): we stop counting if we reach \( n \), as we already know that the at-least restriction is satisfied;
- otherwise, if \( i \) is \( \leq \), then \( 0 \leq j \leq n \): similarly, we can stop counting if we reach \( n + 1 \), as we know that the at-most restrictions is not satisfied.

Once all the necessary nodes have been navigated, the (non-)satisfaction of the number restrictions is established with the transitions of item IV3 in Table 5.

**Example 4.11.** We give an example of how satisfaction of the concept \( \leq 2p_2.B \) is verified at node \( x \) of some canonical interpretation. The relevant fragment of the interpretation is depicted on the left hand side of Fig. 3, and its tree representation on the right hand side. Here, we assume \( b_K = 2 \). Note that \( \leq 2p_2.B \) is satisfied at \( x \), since its only \((p_2, B)\)-neighbor is \( x \) itself.

To verify this, the automaton uses the following transitions. First, it sets the counters to zero with a transition as in item IV1 of Table 5:

\[
\delta_T(\leq 2p_2.B, \{A, B, p_1, p_{2Self}\}) = (0, (\leq 2p_2.B, 0, 0))
\]

Then, it checks whether \( x - 1 \) is a \((p_2, B)\)-neighbor of \( x \) using a transition from item IV(2)1 in Table 5:

\[
\delta_T((\leq 2p_2.B, 0, 0), \{A, B, p_1, p_{2Self}\})
\]

\[
= (((1, \neg p_2) \lor (1, \neg B)) \land (0, (\leq 2p_2.B, 1, 0))) \lor ((1, p_2) \land (1, B) \land (0, (\leq 2p_2.B, 1, 1)))
\]

Since this is not the case, the first disjunct, \(((1, \neg p_2) \lor (1, \neg B)) \land (0, (\leq 2p_2.B, 1, 0)))\) is chosen to satisfy the transition, and only the first counter \( i \) is increased, switching to state \((\leq 2p_2.B, 1, 0)\). To satisfy the first conjunct \((1, \neg p_2) \lor (1, \neg B)\) within the chosen disjunct, the automaton can in turn choose either of its disjuncts, so it chooses the first, \((1, \neg p_2)\). That is, it moves to \( x - 1 \) at state \( \neg p_2 \), and checks the satisfaction of this atomic concept using a transition in item III of Table 4.

\[
\delta_T(\neg p_2, \{A, p_1\}) = \text{true}
\]
From state \(\langle \leq 2p_2.B, 1, 0 \rangle\) the automaton proceeds to the second successor, \(x \cdot 2\), and checks it similarly to \(x \cdot 1\):

\[
\delta_T(\langle \leq 2p_2.B, 1, 0 \rangle, \langle A, B, p_1, p_{2\text{Self}} \rangle) = ((2, \neg p_2) \vee (2, \neg B)) \wedge (0, \langle \leq 2p_2.B, 2, 0 \rangle)) \vee ((2, p_2) \wedge (2, B) \wedge (0, \langle \leq 2p_2.B, 2, 1 \rangle))
\]

Again, since \(x \cdot 2\) is not a \((p_2, B)\)-neighbor of \(x\), the automaton chooses the first disjunct and switches to the state \(\langle \leq 2p_2.B, 2, 0 \rangle\). Now to satisfy the first disjunct it can choose as above state \(\neg p_2\) and verify its satisfaction:

\[
\delta_T(\neg p_2, \langle A, B, p_1 \rangle) = \text{true}
\]

Now that the automaton has navigated all children, it proceeds with \(x\) itself. Since \(x\) is indeed its own \((p_2, B)\)-neighbor, it chooses the second disjunct; both counters are increased, and the automaton switches to \(\langle \leq 2p_2.B, 3, 1 \rangle\). It also verifies, using the atomic transitions, that both \(p_{2\text{Self}}\) and \(B\) are in the label of \(x\):

\[
\delta_T(\langle \leq 2p_2.B, 2, 0 \rangle, \langle A, B, p_1, p_{2\text{Self}} \rangle) = ((0, \neg p_{2\text{Self}}) \vee (0, \neg B)) \wedge (0, \langle \leq 2p_2.B, 3, 0 \rangle)) \vee ((0, p_{2\text{Self}}) \wedge (0, B) \wedge (0, \langle \leq 2p_2.B, 3, 1 \rangle))
\]

\[
\delta_T(p_{2\text{Self}}, \langle A, B, p_1, p_{2\text{Self}} \rangle) = \text{true}
\]

\[
\delta_T(B, \langle A, B, p_1, p_{2\text{Self}} \rangle) = \text{true}
\]

Finally, from \(\langle \leq 2p_2.B, 3, 1 \rangle\), the automaton uses a transition from item IV(2)3 in Table 5 to verify whether the parent of \(x\) is also its \((p_2, B)\)-neighbor:

\[
\delta_T(\langle \leq 2p_2.B, 3, 1 \rangle, \langle A, B, p_1, p_{2\text{Self}} \rangle) = ((0, \neg p_2) \vee (-1, \neg B)) \wedge (0, \langle \leq 2p_2.B, 4, 1 \rangle)) \vee ((0, p_2) \wedge (-1, B) \wedge (0, \langle a \leq 2p_2.B, 4, 2 \rangle))
\]

Since it is not, the automaton chooses the first disjunct and increases the first counter switching to \(\langle \leq 2p_2.B, 4, 1 \rangle\). It also verifies that \(x\) is not a \(p_2\)-successor of its parent by moving to the state \(\neg p_2\); it then has a transition (as in item IV3 of Table 5) which ensures that \(x\) satisfies \(\leq 2p_2.B\), if among the four potential neighbors of \(x\) we have only one \((p_2, B)\)-neighbor:

\[
\delta_T(\neg p_2, \langle A, B, p_1, p_{2\text{Self}} \rangle) = \text{true}
\]

\[
\delta_T(\langle \leq 2p_2.B, 4, 1 \rangle, \langle A, B, p_1, p_{2\text{Self}} \rangle) = \text{true}
\]

Example 4.12 (Continued). For the same interpretation and tree representation as in the previous example, let us now verify that \(x\) satisfies the concept \(\geq 3(p_1 \cup p_2).A\).

The automaton again starts by setting the counters to zero:

\[
\delta_T(\geq 3(p_1 \cup p_2).A, \langle A, B, p_1, p_{2\text{Self}} \rangle) = (0, \langle \geq 3(p_1 \cup p_2).A, 0, 0 \rangle)
\]

It moves to the first successor \(x \cdot 1\) and verifies that it is indeed a \((p_1 \cup p_2, A)\)-neighbor of \(x\), and increases both counters by one:

\[
\delta_T(\geq 3(p_1 \cup p_2).A, 0, 0, \langle A, B, p_1, p_{2\text{Self}} \rangle) = ((1, \neg p_1 \cap \neg p_2) \vee (1, \neg A)) \wedge (0, \langle \geq 3(p_1 \cup p_2).A, 1, 0 \rangle) \]

\[
\vee ((1, p_1 \cup p_2) \wedge (1, A) \wedge (0, \langle \geq 3(p_1 \cup p_2).A, 1, 1 \rangle))
\]

Here, the right disjunct is chosen for satisfaction. To check that \(x \cdot 1\) is a \((p_1 \cup p_2, A)\)-neighbor of \(x\), the automaton needs to decompose the simple role \((p_1 \cup p_2)\) and choose one of its disjuncts:

\[
\delta_T(p_1 \cup p_2, \langle A, p_1 \rangle) = (0, p_1) \vee (0, p_2)
\]

\[
\delta_T(p_1, \langle A, p_1 \rangle) = \text{true}
\]

\[
\delta_T(A, \langle A, p_1 \rangle) = \text{true}
\]

In this case, let it choose \(p_1\). From \(\langle \geq 3(p_1 \cup p_2).A, 1, 1 \rangle\) the automaton proceeds similarly and verifies that \(x \cdot 2\) is also a \((p_1 \cup p_2, A)\)-neighbor of \(x\):
As above, the right disjunct is chosen for satisfaction. From \(\geq 3(p_1 \cup p_2).A, 2, 2\) the automaton verifies that \(x\) is its own \((p_1 \cup p_2, A)\)-neighbor using item IV(2)2 in Table 5:

\[
\begin{align*}
\delta_T((\geq 3(p_1 \cup p_2).A, 2, 2), \{A, B, p_1, p_2\}) &= ((0, \neg(p_1 \cup p_2) \cup (0, \neg A)) \wedge (0, \geq 3(p_1 \cup p_2).A, 3, 2)) \\
&\quad \lor ((0, (p_1 \cup p_2) \cup (0, A)) \wedge (0, \geq 3(p_1 \cup p_2).A, 3, 3))
\end{align*}
\]

That is, the right disjunct is chosen and the automaton verifies that \(x\) satisfies \((p_1 \cup p_2)\) using role decomposition and atomic checks:

\[
\begin{align*}
\delta_T((p_1 \cup p_2) \cup (0, p_1) \cup (0, p_2)) &= (0, p_1) \lor (0, p_2) \\
\delta_T((p_2) \cup (0, p_1) \cup (0, p_2)) &= true \\
\delta_T((A, B, p_1, p_2)) &= true
\end{align*}
\]

Finally, it checks the number restriction \(\geq 3(p_1 \cup p_2).A, 3, 3\) using a transition of item IV3 in Table 5:

\[
\begin{align*}
\delta_T((\geq 3(p_1 \cup p_2).A, 3, 3), \{A, B, p_1, p_2\}) &= true
\end{align*}
\]

This suffices to establish that \(x\) satisfies \(\geq 3(p_1 \cup p_2).A\). Note that in this case, there is no need to navigate the parent of \(x\) (item IV(2)3 in Table 5). ∎

Example 4.13. Finally, to illustrate the way the automaton \(A_T\) verifies the satisfaction of universal restrictions by ABox individuals, we consider the partial interpretation depicted on the right hand side of Fig. 4. Its tree representation, assuming \(b_K = 3\) and \(I_K = \{a, b, c\}\), is given on the right hand side.

To verify whether the concept \(\forall p.\neg A\) is satisfied at the node 1 interpreting \(a\) (that is, when the automaton is in node 1 and state \(\forall p.\neg A\)), it moves to the state \(\leq 0p.A\) using the corresponding transition in item II of Table 4. Then, it proceeds similarly to the examples above.

\[
\begin{align*}
\delta_T(\leq 0p.A, \{a\}) &= (0, \leq 0p.A, 0, 0)
\end{align*}
\]

Then it successively checks that none of 1-1, 1-2, and 1-3 is a \((p, A)\)-neighbor of 1. It always chooses the first disjunct of the corresponding transition and increases only the first counter:

\[
\begin{align*}
\delta_T((\leq 0p.A, 0, 0), \{a\}) &= (((1, \neg p) \lor (1, \neg A)) \wedge (0, \leq 0p.A, 1, 0)) \lor ((1, p) \wedge (1, A) \wedge (0, \leq 0p.A, 1, 1)) \\
\delta_T((\neg A, \{p, B\}) &= true \\
\delta_T((\leq 0p.A, 1, 0), \{a\}) &= (((2, \neg p) \lor (2, \neg A)) \wedge (0, \leq 0p.A, 2, 0)) \lor ((2, p) \wedge (2, A) \wedge (0, \leq 0p.A, 2, 1)) \\
\delta_T((\neg p, \{s, A, B\}) &= true \\
\delta_T((\leq 0p.A, 2, 0), \{a\}) &= (((3, \neg p) \lor (3, \neg A)) \wedge (0, \leq 0p.A, 3, 0)) \lor ((3, p) \wedge (3, A) \wedge (0, \leq 0p.A, 3, 1)) \\
\delta_T((\neg p, \{\}) &= true
\end{align*}
\]
Since $I(1) \cap I_C = \{a\}$, from $(\leq 0.p.A, 3, 0)$, the automaton uses a transition as in item IV(2)4 of Table 5 to move to the root and to the state $(a, \leq 0.p.A, 0, 0)$, resetting the first counter to 0:

$$\delta_T((a, \leq 0.p.A, 3, 0), [a]) = (-1, (a, \leq 0.p.A, 0, 0))$$

From here, it verifies whether each of the nodes 1 to 3 is a $(p, A)$-neighbor of the node labeled $a$, using transitions as in item IV(2)5 of Table 5. For node 1, the transition is as follows:

$$\delta_T((a, \leq 0.p.A, 0, 0), [r, p_{12}, p_{13}]) = (((((0, \sim p_{11}) \lor (1, \sim a)) \land (0, \sim p_{21}) \lor (2, \sim a)) \land (0, \sim p_{31}) \lor (3, \sim a)) \lor (1, \sim A) \lor ((1, \sim a) \land (1, \sim b) \lor (1, \sim c)) \land (0, (a, \leq 0.p.A, 1, 0)) \lor (((p_{11}) \land (1, a)) \lor ((0, p_{21}) \lor (2, a)) \lor ((0, p_{31}) \lor (3, a) \land (1, A) \lor ((1, a) \lor (1, b) \lor (1, c)) \land (0, (a, \leq 0.p.A, 1, 1)))$$

$A$ is not in the label of 1, and this is enough to ensure that 1 is not a $(p, A)$-neighbor of any node. Hence the automaton needs to choose the first big disjunct, and satisfy it by moving to 1 in state $\sim A$. It increases only the first counter to stay at the root in state $(a, \leq 0.p.A, 1, 0)$, from which it verifies node 2:

$$\delta_T(-A, [a]) = true$$

$$\delta_T((a, \leq 0.p.A, 1, 0), [r, p_{12}, p_{13}]) = (((((0, \sim p_{12}) \lor (1, \sim a)) \land (0, \sim p_{22}) \lor (2, \sim a)) \land (0, \sim p_{32}) \lor (3, \sim a)) \lor (2, \sim A) \lor ((2, \sim a) \land (2, \sim b) \lor (2, \sim c)) \land (0, (a, \leq 0.p.A, 2, 0)) \lor (((p_{12}) \land (1, a)) \lor ((0, p_{22}) \lor (2, a)) \lor ((0, p_{32}) \lor (3, a)) \land (2, A) \lor ((2, a) \lor (2, b) \lor (2, c)) \land (0, (a, \leq 0.p.A, 2, 1)))$$

Again, as 2 is not a $(p, A)$-neighbor of any node, the automaton must choose to move to 2 in state $\sim A$ and to increase only the first counter:

$$\delta_T(-A, [b, B]) = true$$

$$\delta_T((a, \leq 0.p.A, 2, 0), [r, p_{12}, p_{13}]) = (((((0, \sim p_{13}) \lor (1, \sim a)) \land (0, \sim p_{23}) \lor (2, \sim a)) \land (0, \sim p_{33}) \lor (3, \sim a)) \lor (3, \sim A) \lor ((3, \sim a) \land (3, \sim b) \lor (3, \sim c)) \land (0, (a, \leq 0.p.A, 3, 0)) \lor (((p_{13}) \land (1, a)) \lor ((0, p_{23}) \lor (2, a)) \lor ((0, p_{33}) \lor (3, a)) \land (3, A) \lor ((3, a) \lor (3, b) \lor (3, c)) \land (0, (a, \leq 0.p.A, 3, 1)))$$

Now the automaton can not choose the first disjunct: it cannot satisfy the first part as $p_{13}$ is in the label of the root and $a$ is in the label of node 1. It cannot move to node 3 in state $\sim A$ either, as its label contains $A$, and it cannot satisfy the third part as $c$ is in the label of node 3. If it chooses the second disjunct, it needs to move to state $p_{13}$, to state $a$ in node 1, and to states $A$ and $c$ in node 3. But then it would have to increase both counters; by item IV3 in Table 5, all transitions from state $(a, \leq 0.p.A, 3, 1)$ are false:

$$\delta_T(p_{13}, [r, p_{12}, p_{13}]) = true$$

$$\delta_T(a, [a]) = true$$

$$\delta_T(A, [c, A]) = true$$

$$\delta_T((a, \leq 0.p.A, 3, 1), [r, p_{12}, p_{13}]) = false$$

This shows that the number restriction $\leq 0.p.A$, and hence $\forall p.\sim A$, is not satisfied at node 1, as node 3 is indeed a $(p, A)$-neighbor of 1. □

As we have mentioned, infinite paths in the runs of $A_T$ only arise from the transitions that move from states of the form $\forall R^*C$ and $\exists R^*C$ to states of the form $\forall R.\forall R^*C$ and $\exists R.\exists R^*C$, and then move to all or to some successors of the current node in the preceding state, that is, to states $\forall R^*C$ and $\exists R^*C$, respectively. Hence, in every infinite path of a run, a state of either (i) the form $\forall R^*C$ or (ii) the form $\exists R^*C$ occurs infinitely often. Case (i) amounts to the existence of an infinite sequence of $R$-neighbors in the represented interpretation, over which the satisfaction of the concept $\forall R^*C$ is correctly ensured. Paths of case (ii) are also caused by infinite sequences of $R$-neighbors in interpretations, but do not correctly ensure the satisfaction of the concept $\exists R^*C$. To satisfy $\exists R^*C$ at some node $x$, we need to ensure that on every infinite sequence of $R$-neighbors starting from $x$, an instance of $C$ is eventually reached. Hence we disallow paths where $\exists R^*C$ occurs infinitely often, requiring that the disjunct $C$ is eventually chosen and the satisfaction of $C$ is not indefinitely postponed; this is achieved by the acceptance condition (cf. [35]).
Summing up, the core property of the automaton $A_T$ is that it correctly verifies the satisfaction of the concepts in the closure of $C_T$: an accepting run of $A_T$ over an interpretation tree $T$ can have a node labeled $(x,C)$ iff $x$ is an instance of $C$ in the interpretation $I_T$. To formalize this argument, recall that an $(x,s)$-run is just like a (full) run, but the root must be labeled with a given pair of an arbitrary node $x$ of the input tree and a state $s$ (see Definition 2.8). We now show:

**Lemma 4.14.** Let $T = (T, I)$ be an interpretation tree for $K$ and let $x \in \Delta^T$. Furthermore, let $C \in Q(A_T)$ be a concept in $C_l(C_T)$. Then there is an accepting $(x,C)$-run of $A_T$ over $T$ iff $x \in C^T$. 

**Proof (Sketch).** We give only a short sketch here; a more detailed proof can be found in Appendix A. We first show similar concepts correctly, according to the semantics of the different constructors. The only interesting cases are (i) $C = \forall$ and (ii) $C = \exists$. For (i) and (ii), we rely on the following auxiliary claims, where we use the notions of potential neighbors and $(S,D)$-neighbors defined above:

(C1) Let $x \in \Delta^T$ and let $S$ be a simple role in $C_l(C_T)$. Then there is an accepting $(x,i,S)$-run of $A_T$ over $T$ iff $x \in S^{\Delta^T}$. 

(C2) Let $x \in \Delta^T$ and let $S_{\text{set}} \in Q_{\text{set}}$. Then there is an accepting $(x,S)$-run of $A_T$ over $T$ iff $(x,s) \in S^{\Delta^T}$. 

(C3) Let $i, j \in \Delta^T$ and $S_{ij} \in Q_{\text{role}}$. Then there is an accepting $(i,S_{ij})$-run of $A_T$ over $T$ iff $(i,j) \in S^{\Delta^T}$. 

Each of these claims is shown by a straightforward structural induction on the role $S$. Then, using (C1)-(C3), one can show Lemma 4.14 by structural induction on $C$. The induction is rather tedious, due to the large number of constructors of $\forall \exists Q$. Most cases are straightforward, since the transitions in the left column of item II in Table 4 decompose complex concepts correctly, where we use the notions of potential neighbors and $(S,D)$-neighbors defined above:

(C4) Let $x \in \Delta^T$ and let $(\geq n.S.D,i,j) \in Q_{\text{num}}$ (resp., $(\leq n.S.D,i,j) \in Q_{\text{num}}$). Then there is an accepting $(x,(\geq n.S.D,i,j))$-run (resp., $(x,(\leq n.S.D,i,j))$-run) of $A_T$ over $T$ iff there are at least (resp., at most) $n - j$ many $(S,D)$-neighbors of $x$ among its potential neighbors beyond the $i$-th one. 

(C5) Let $(a,\geq n.S.D,i,j) \in Q_{\text{num}}$ (resp., $(a,\leq n.S.D,i,j) \in Q_{\text{num}}$). Then there is an accepting $(a,(\geq n.S.D,i,j))$-run of $A_T$ over $T$ iff there are at least (resp., at most) $n - j$ many $(S,D)$-neighbors of $a^{\Delta^T}$ among its potential neighbors beyond the $(b^i + i)$-th. 

The interesting feature of (iii) and (iv) is that a run can repeatedly generate successors labeled with $(x', C)$. As discussed above, this is handled in the usual way by the termination condition. If $C = \forall D$, $S_{\text{set}}$ is satisfied, the generation of nodes $(x', C)$ may be infinitely repeated, but the resulting branch is accepting as $\text{Inf}(\tau) = \forall D.S_{\text{set}}$, and $F_T$ is satisfied. For $\exists D$, this will happen only finitely often: as $x \in (3R^D)^{\Delta^T}$, $D$ must be eventually reached on some finite path, and some $x' \in D^{\Delta^T}$ will be encountered; we can then add $(x,D)$ to the run. \hfill $\Box$

Now we can easily show that $L(A_T)$ accepts exactly the interpretation trees that represent a model of $\mathcal{T}$. 

**Lemma 4.15.** Let $T$ be an interpretation tree for $K$, then $T \in L(A_T)$ iff $I_T \models \mathcal{T}$. 

**Proof (Sketch).** Since $T$ is an interpretation tree for $K$, $I_T$ is a $b_K$-canonical interpretation. By the transitions in item I of Table 4, $A_T$ first moves to each node $i \in \text{Roots}(I_T)$ in state $C_T$. Then, by Lemma 4.14, it will succeed in completely decomposing $C_T$ at $i$ iff $i \in C^T$. Hence, $A_T$ has an accepting run on $T$ (i.e., $T \in L(A_T)$) iff $\text{Roots}(I_T) \subseteq C^T$, and it follows from Proposition 3.8 that $T \in L(A_T)$ iff $I_T \models \mathcal{T}$. \hfill $\Box$

4.2.4. **Automaton $A_K$ verifying $KB$ satisfaction**

Finally, by intersecting the automata defined above, we obtain the desired automaton $A_K$.

**Definition 4.16.** Given $K$, let $A_K$ be a 2ATA such that $L(A_K) = L(A_T) \cap L(A_A) \cap L(A_T)$, as in Proposition 2.10. \hfill $\Box$

4.3. **Soundness and completeness**

The following proposition states soundness and completeness of $A_K$ with respect to canonical models of $K$.

**Proposition 4.17.** For a given normalized $\forall \exists Q KB K$, $L(A_K) = \{T \mid T$ is an interpretation tree and $I_T \models K\}$. 

**Proof.** $(\subseteq)$. By definition, $T \in L(A_K)$ implies $T \in L(A_T) \cap L(A_A) \cap L(A_T)$. As $T \in L(A_T)$, by Lemmas 4.5 and 4.7 $T$ is an interpretation tree for $K$ and $I_T$ is a canonical interpretation for $K$. Furthermore, by Lemma 4.9, $T \in L(A_A)$ implies $I_T \models A$, and by Lemma 4.15, $T \in L(A_T)$ implies $I_T \models I$. Consequently, $I_T$ is a canonical model of $K$. 

**Proof.** $(\supseteq)$. By definition, $T \in L(A_K)$ implies $T \in L(A_T) \cap L(A_A) \cap L(A_T)$. As $T \in L(A_T)$, by Lemmas 4.5 and 4.7 $T$ is an interpretation tree for $K$ and $I_T$ is a canonical interpretation for $K$. Furthermore, by Lemma 4.9, $T \in L(A_A)$ implies $I_T \models A$, and by Lemma 4.15, $T \in L(A_T)$ implies $I_T \models I$. Consequently, $I_T$ is a canonical model of $K$. 

\hfill $\Box$
Theorem 4.18. A normalized $\mathcal{ZIQ}$ KB $\mathcal{K}$ is satisfiable iff $\mathcal{L}(\mathcal{A}_\mathcal{K}) \neq \emptyset$.

Proof. If $\mathcal{L}(\mathcal{A}_\mathcal{K}) \neq \emptyset$, then by Proposition 4.17 $\mathcal{K}$ has some model, hence $\mathcal{K}$ is satisfiable. Conversely, if $\mathcal{K}$ is satisfiable, then by Theorem 3.10 it has some canonical model $\mathcal{I}$, and as $\mathcal{I} = \mathcal{I}_L$ (by Lemma 4.5), by Proposition 4.17 $\mathcal{I}_L \in \mathcal{L}(\mathcal{A}_\mathcal{K})$, hence $\mathcal{L}(\mathcal{A}_\mathcal{K}) \neq \emptyset$. □

Thus, checking satisfiability of a normalized $\mathcal{ZIQ}$ KB $\mathcal{K}$ reduces to testing the automaton $\mathcal{A}_\mathcal{K}$ for emptiness.

4.4. Complexity

By $\|\mathcal{K}\|$ we denote the size of a (string) representation of $\mathcal{K}$; we assume here unary number encoding, i.e., the numbers $n$ in number restrictions are encoded by a string of length $\Theta(n)$. Recall that $\mathcal{C}_\mathcal{K}$ and $\mathcal{R}_\mathcal{K}$ denote the atomic concepts and roles, respectively, that occur in $\mathcal{K}$, and $\mathcal{I}_\mathcal{K}$ denotes the ABox individuals; $b_\mathcal{K}$ denotes $\max(k_T, |I_\mathcal{K}|)$ where $k_T = |O(C_T)| \cdot \max(|n| \geq n.S.C \in Cl(C_T) \cup \{0\})$. Furthermore, let $n'_{\max} = \max(|n| \geq n.S.C \in Cl(C_T) \cup \{0\})$. Note that $|\mathcal{C}_\mathcal{K}|$, $|\mathcal{R}_\mathcal{K}|$, $|I_\mathcal{K}|$, and $|Cl(C_T)|$ are linear in $\|\mathcal{K}\|$. Under unary number encoding, this holds also for $n'_{\max}$, and $b_\mathcal{K}$ is quadratic in $\|\mathcal{K}\|$. We thus obtain:

Lemma 4.19. For $\mathcal{A}_\mathcal{K}$, we have $|\Sigma(\mathcal{A}_\mathcal{K})| \leq 2^{O(\|\mathcal{K}\|^5)}$, $|Q(\mathcal{A}_\mathcal{K})| \leq O(\|\mathcal{K}\|^5)$, and $\text{ind}(\mathcal{A}_\mathcal{K}) = 3$.

Proof. Recall that $\Sigma(\mathcal{A}_\mathcal{K}) = \Sigma_k$ and $Q(\mathcal{A}_\mathcal{K}) = Q_k$. The result is a consequence of the following simple estimates:

- $|\Sigma(\mathcal{A}_\mathcal{K})| = 2^{M(\mathcal{K})}$, where $M(\mathcal{K}) = |\mathcal{C}_\mathcal{K}| + |\mathcal{R}_\mathcal{K}| + |I_\mathcal{K}| + |P| + |PS| + 1$, and we have $|P| = |\mathcal{R}_\mathcal{K}| \cdot |b_\mathcal{K}|^2$ and $|P| = |\mathcal{R}_\mathcal{K}|$. Clearly, $M(\mathcal{K}) = O(\|\mathcal{K}\|^5)$.
- $|Q_k| \leq |Q_A| + |Q_A| + |Q_T| + 1$, where the following bounds hold:
  
  \[
  |Q_A| \leq 1 + 2|I_\mathcal{K}| + |\mathcal{C}_\mathcal{K}| + |\mathcal{R}_\mathcal{K}| \cdot |I_\mathcal{K}|^2
  \]
  
  \[
  |Q_T| \leq 2 + (|I_\mathcal{K}| + 1)
  \]
  
  \[
  |Q_A| \leq |C_{\text{ext}}| + |Q_{\text{set}}| + |Q_{A_{\text{role}}}| + |Q_{\text{num}}| + |Q_{A_{\text{num}}}|\]
  
  \[
  |Q_{\text{excl}}| \leq |Cl(C_T)| + 2|I_\mathcal{K}|
  \]
  
  \[
  |Q_{\text{set}}| \leq |Cl(C_T)| + |I_\mathcal{K}| \cdot |Cl(C_T)|
  \]
  
  \[
  |Q_{A_{\text{role}}}| \leq |Cl(C_T)| \cdot |b_\mathcal{K}|^2
  \]
  
  \[
  |Q_{A_{\text{num}}}| \leq |I_\mathcal{K}| \cdot |Cl(C_T)| \cdot |b_\mathcal{K} \cdot (n'_{\max} + 1)
  \]

  Hence, it is easy to see that $|Q_k| = O(\|\mathcal{K}\|^5)$ (cf. $|Q_{A_{\text{num}}}|$).

- $\text{ind}(\mathcal{A}) = \max(\text{ind}(\mathcal{A}_X), \text{ind}(\mathcal{A}_A), \text{ind}(\mathcal{A}_T)) = \max(2, 2, 3) = 3$. □

Thus, by Theorems 2.9 and 4.18, we get an optimal upper bound for KB satisfiability.

Corollary 4.20. Deciding whether a given KB in $\mathcal{ZIQ}$ is satisfiable is in ExpTime under unary number encoding.

This is worst-case optimal, since a matching hardness result holds already for much weaker DLs, e.g., $\mathcal{ALC}$ [2].

5. Query answering via automata

We now turn to entailment of P2RPQs in KBs. As follows from Theorem 3.10, to decide whether $\mathcal{K} \models q$ for a normalized P2RPQ $q$ and a normalized KB $\mathcal{K}$ in this DL, it is sufficient to decide whether $\mathcal{K}$ has a canonical model in which $q$ has no match. We show how to do this using tree automata. Specifically, we build an automaton $\mathcal{A}_{\mathcal{K} \models q}$ that accepts all trees that represent a canonical model of $\mathcal{K}$ in which $q$ has no match; hence, deciding query entailment reduces to checking $\mathcal{L}(\mathcal{A}_{\mathcal{K} \models q}) = \emptyset$. Roughly speaking, $\mathcal{A}_{\mathcal{K} \models q}$ is obtained by intersecting two automata: $\mathcal{A}_\mathcal{K}$ from Section 4.2 (which accepts the trees representing a canonical model of $\mathcal{K}$), and $\mathcal{A}_{\neg q}$, which is constructed in this section and accepts the trees representing
an interpretation that admits no match for \( q \). We first construct an automaton \( A_q \) that accepts a tree \( \mathcal{T} \) iff \( q \) has a match in the interpretation \( \mathcal{I}_q \); we then show how to obtain from \( A_q \) the desired \( A_{\neg q} \). Fig. 5 gives a general overview of the query answering technique; each of the steps will be discussed in detail below. More information about the construction of the automata in Fig. 5 is summarized in Tables 6 and 8.

All automata we have constructed in Section 4 run over \( b_{\mathcal{K}} \)-ary trees, where \( b_{\mathcal{K}} = \max(k_{\mathcal{T}}, |I_\mathcal{K}|) \) for \( k_{\mathcal{T}} = br(C_T) \). By Theorem 3.10, we know that to decide query entailment it suffices to consider the \( k_{\mathcal{T}, \neg q} \)-canonical models of \( \mathcal{K} \), for \( k_{\mathcal{T}, \neg q} = br((C_T) \cup \mathcal{D}_q) \) as in Definition 3.9. To be able to use the same automata constructions for query answering, in this section we assume that \( k_{\mathcal{T}, \neg q} \leq b_{\mathcal{K}} \). Note that this is trivially true for queries in which complex concepts are disallowed. It also holds if the numbers \( n \) in number restrictions \( \geq n.S.C \) and \( \leq n.S.C \) occurring in \( q \) are not greater than those in the query (and in the absence of number restrictions, if \( \mathcal{K} \) contains some existential or universal concept whenever \( q \) does). For an arbitrary \( q \) and \( \mathcal{K} \), we can enforce this condition by simply adding \( C \subseteq C \) to the TBox of \( \mathcal{K} \) for every concept \( C \) that occurs in \( q \) and contains a number restriction, or a universal or existential concept.

5.1. Representing query matches

Prior to defining \( A_q \), we extend the tree representation of interpretations to include also query matches. In what follows, we assume a given normalized P2RPQ \( q = \exists \nu.\varphi(\nu) \). We assume w.l.o.g. that in \( q \) each atom \( C(\nu) \) has been equivalently replaced with \( id(C)(\nu, \nu) \), so that all atoms in \( At(q) \) are of the form \( R(\nu, \nu') \), where \( R \) is in NNF. Let \( V_q = \{ \nu_1, \ldots, \nu_{\ell} \} \) be the variables in \( \overline{\nu} \). We denote by \( C_q, R_q, \) and \( I_q \) the sets of atomic concepts, role names, and individuals, respectively, that occur in \( q \), and define \( \overline{R}_q = R_q \cup \{ p^\sim \mid p \in R_q \} \).

We need the following notion, combining an interpretation \( \mathcal{I} \) with a possible match \( \pi \) for \( q \).

**Definition 5.1.** An extended (canonical) interpretation is a pair \( \mathcal{I} = (\mathcal{I}, \pi) \) consisting of a (canonical) interpretation \( \mathcal{I} \) and a total function \( \pi : V_q \cup I_q \rightarrow \Delta^\mathcal{I} \) such that \( \pi(a) = a^\mathcal{I} \) for each \( a \in I_q \). If \( \mathcal{I}, \pi \models q \), we write \( \mathcal{I} \models q \).

**Example 5.2.** Recall the interpretation \( \mathcal{I}_q \) given in Example 4.3, and the mapping \( \pi(\nu_1) = Zeus^{\mathcal{I}_q}, \pi(\nu_2) = Alcmen{\mathcal{I}_q}^{\mathcal{I}_q} \) and \( \pi(\nu_3) = Heracles^{\mathcal{I}_q} \) for the query \( q_q \) in Example 2.5; the pair \( (\mathcal{I}_q, \pi) \) is an extended interpretation.

Extended canonical interpretations are represented by extended interpretation trees that are labeled using an alphabet \( \Sigma_{\mathcal{K}, q} \). The latter enriches \( \Sigma_{\mathcal{K}} \) with the variables in \( V_q \) and for each \( \nu \in V_q \) allows us to include the symbol \( \nu \) in the label of the node \( \pi(\nu) \). We construct below an automaton \( A_q \) that accepts a \( \Sigma_{\mathcal{K}, q} \)-labeled tree iff it represents an extended interpretation \( (\mathcal{I}, \pi) \) such that \( \pi \) is actually a match for \( \mathcal{I} \) and \( q \), i.e., \( \mathcal{I}, \pi \models q \).

**Definition 5.3.** Let \( \Sigma_{\mathcal{K}, q} = \{ \sigma \cup \sigma' \mid \sigma \in \Sigma_{\mathcal{K}} \) and \( \sigma' \in 2^{V_q} \). An extended interpretation tree is a \( \Sigma_{\mathcal{K}, q} \)-labeled \( b_{\mathcal{K}} \)-ary tree \( T = (T, L) \) such that:

1. Its \( \Sigma_{\mathcal{K}} \)-restriction \( T|_{\Sigma_{\mathcal{K}}} = (T, L|_{\Sigma_{\mathcal{K}}} \), where \( L|_{\Sigma_{\mathcal{K}}} (x) = L(x) \cap \Sigma_{\mathcal{K}} \) for each \( x \in T \), is an interpretation tree. Abusing notation, we use \( L \) to denote \( L|_{\Sigma_{\mathcal{K}}} \).
2. For each \( \nu \in V_q \) there is exactly one \( x \in T \) with \( \nu \in L(x) \), called the candidate match for \( \nu \); furthermore, \( x \) is such that \( x \in \Delta^T \) (i.e., it is connected to an individual node via the roles in \( \overline{R}_K \)).

We let \( \pi_T : V_q \cup I_q \rightarrow T \) be the function that maps each \( \nu \in V_q \) to its candidate match, and each \( a \in I_q \) to the (unique) node \( i \in T \) with \( a \in L(i) \).
We associate extended interpretation trees with extended interpretations and vice-versa as follows.

**Definition 5.4.** The extended interpretation represented by an extended interpretation tree $T$ is $J_T = (I_T, \pi_T)$. The tree representation $T_J$ of an extended canonical interpretation $J = (I, \pi)$ is the extended interpretation tree $T_J = (T, L)$ whose $\Sigma_K$-restriction is $T_I$ and such that, for each $v \in V_q$ and each $x \in T$, $v \in L(x)$ iff $x = \pi(v)$. $\square$

Analogously to Lemma 4.5, we have:

**Lemma 5.5.** If $T$ is an extended interpretation tree, then $J_T$ is an extended canonical interpretation.

**Proof.** If $T$ is an extended interpretation tree, then by condition (e1) in Definition 5.3, $T|\Sigma_K$ is an interpretation tree and by Lemma 4.5, $T_I$ is an extended interpretation. By condition (t2) in Definition 4.1, for each individual $a \in I_K$ there is exactly one $i \in I|\Sigma_K$ such that $a \in I(i)$. Furthermore, $i \in \Delta^T$ and $a^{\Delta^T} = i$. Similarly, for each $v \in V_q$, by condition (e2) in Definition 5.3, there is exactly one $x \in T|\Sigma_K$ such that $x \in \Delta^T$ and $v \in L(x)$. Therefore the mapping $\pi_T$ is well-defined and it defines a total function from $V_q \cup I_q$ to $\Delta^T$, with $\pi(a) = a^{\Delta^T}$. This proves that $J_T$ is an extended canonical interpretation. $\square$

**Example 5.6.** The tree representation of the extended interpretation $(I_g, \pi)$ (see Examples 5.2 and 4.3) is shown in Fig. 6. It extends the tree representation of Fig. 2 with the variables $V_q = \{v_1, v_2, v_3\}$. $\square$

### 5.2. Constructing the automaton that checks query matches

Now we construct the automaton $A_q$ that accepts a $\Sigma_{K,q}$-labeled tree $T$ iff (i) $T$ is an extended interpretation tree and (ii) the map $\pi_T$ represents a match for the query $q$ in the associated interpretation $I_T$. We define $A_q$ as the intersection of automata $A_T$ and $A_r$ for (i) and (ii), respectively; they are summarized in Table 6. All these automata are 2ATAs that run over $b_K$-ary trees. As seen later, their size (in terms of states) is polynomial in $|K|$.

As for $A_T$, we can easily define first a 2ATA $A_V$ that verifies whether a given tree satisfies condition (e2) of Definition 5.3. We then obtain the desired $A_T$ by intersecting $A_V$ with an adaptation $A^\prime_T$ of $A_T$ from Section 4.2, such that its alphabet is $\Sigma_{K,q}$ and it accepts all trees whose $\Sigma_K$-restrictions are interpretation trees.

**Definition 5.7.** The 2ATA $A_V = (b_K, \Sigma_{K,q}, Q_V, \delta_V, s_0^V, F_V)$ is defined as follows:

- $Q_V = \{s^V \} \cup V_q \cup \bigcup_{v \in V_q} \{\neg v, v^+, v^-\} \cup R_K$. Intuitively, the state $v$ is used to check that the label of the current node contains $v$, and the state $\neg v$ that it doesn't. The state $v^+$ is used to check that the node labeled $v$ is in the tree rooted at the current node, and the state $v^-$ that it is not.
• $F_V = (\emptyset, \{ v^- \mid v \in V_q \}, Q_V)$.
• The transition function $\delta_V : Q_V \times \Sigma_{K,q} \to B([b_K] \times Q_V)$ contains three groups of transitions:
  1. For each $\sigma \in \Sigma_{K,q}$ with $r \in \sigma$, a transition from the initial state:

$$\delta_V(s^V_0, \sigma) = \bigwedge_{v \in V_q} \left( (0, v^-) \vee \bigvee_{1 \leq j \leq b_K} \left( \bigvee_{a \in I_K} (j, a) \wedge \bigwedge_{1 \leq j' \leq b_K, j' \neq j} (j, v^+ \wedge 1) \right) \right)$$

  2. For each $v \in V_q$ and each $\sigma \in \Sigma_{K,q}$, transitions to the subtrees:

$$\delta_V(v^+, \sigma) = \left( (0, v) \wedge \bigvee_{1 \leq i \leq b_K} (i, v^-) \right) \vee \left( (0, \neg v) \wedge \bigvee_{1 \leq i \leq b_K} (i, v^+) \right)$$

$$\delta_V(v^-, \sigma) = (0, \neg v) \wedge \bigwedge_{1 \leq i \leq b_K} (i, v^-)$$

  3. For each $\sigma \in \Sigma_{K,q}$ and each $v \in V_q$, transitions that check the labeling with $v$:

$$\delta_V(v, \sigma) = \begin{cases} \text{true}, & \text{if } v \in \sigma \\ \text{false}, & \text{if } v \notin \sigma \end{cases}$$

$$\delta_V(\neg v, \sigma) = \begin{cases} \text{true}, & \text{if } v \notin \sigma \\ \text{false}, & \text{if } v \in \sigma \end{cases}$$

  4. For each $\sigma \in \Sigma_{K,q}$ and each $P \in R_K$, transitions that check the labeling with $P$:

$$\delta_V(P, \sigma) = \begin{cases} \text{true}, & \text{if } P \in \sigma \\ \text{false}, & \text{if } P \notin \sigma \end{cases}$$

The automaton $A_T$ in Definition 4.6 is modified to $A_T' = (b_K, \Sigma_{K,q}, Q_T, \delta_T', s^T_0, F_T)$, by changing $\Sigma_K$ to $\Sigma_{K,q}$ and setting, for each $\sigma \in \Sigma_{K,q}$ and each state $q' \in Q_T$, $\delta_T'(\sigma, q') = \delta_T(\sigma', q')$ whenever $\sigma' = \sigma \setminus V_q$. Then, $A_T$ is an automaton that accepts the intersection of $A_V$ and $A_T'$, as in Lemma 2.10. □

Lemma 5.8. $\mathcal{L}(A_T) = \{ T \mid T \text{ is an extended interpretation tree for } K \}$.

Proof. $\mathcal{L}(A_T)$ is the intersection of $\mathcal{L}(A_T')$ and $\mathcal{L}(A_V)$. By construction, for every $\Sigma_{K,q}$-labeled $b_K$-ary tree $T = (T, L)$, we have $T \in \mathcal{L}(A_T')$ iff $A_T \models \Sigma_{K,q} \in \mathcal{L}(A_T)$. Hence $T \in \mathcal{L}(A_T')$ iff it satisfies (e1) in Definition 5.3. It is then enough to prove that there is an accepting run of $A_V$ over $T$ iff $T$ satisfies (e2) in Definition 5.3. To this aim, we first observe that, by item 3 in the definition of $\delta_V$, the following hold for every $x \in T$ and $v \in V_q$:

• There is an accepting $(x, v)$-run of $A_V$ over $T$ iff $v \in L(x)$.
• There is an accepting $(x, \neg v)$-run of $A_V$ over $T$ iff $v \notin L(x)$.

Similarly, by item 4, for every $x \in T$ and $P \in R_K$:

• There is an accepting $(x, P)$-run of $A_V$ over $T$ iff $P \in L(x)$.

Then we can use the second item in the definition of $\delta_V$ to show the following:

• There is an accepting $(x, v^-)$-run of $A_V$ over $T$ iff $v \notin L(x \cdot w)$ for every $x \cdot w \in T$ with $w \in \mathbb{N}^*$. This is an easy consequence of the fact that the transition function ensures that there exists an accepting $(x, v^-)$-run iff there exists an accepting $(x, \neg v)$-run and, for each child $i$ of $x$ there exists an accepting $(x_i, v^-)$-run.
• Assume $x \in \Delta_T^x$. There is an accepting $(x, v^+)$-run of $A_V$ over $T$ iff there is exactly one $x \cdot w \in T$ with $w \in \mathbb{N}^*$ such that $v \in L(x \cdot w)$, and for this $x \cdot w$, we have $x \cdot w \in \Delta_T^x$. This holds because the transition function ensures that there exists an accepting $(x, v^+)$-run if either: (i) there is an accepting $(x, v)$-run, and for every child $x_i$ of $x$, there exists an accepting $(x_i, v^-)$-run; or (ii) there is an accepting $(x, \neg v)$-run, there is one child $x_i$ of $x$ such that there exists an accepting $(x_i, v^-)$-run and an accepting $(x_i, P)$-run for some $P \in R_K$, and for all remaining children $x_j$, $j \neq i$, there exists an accepting $(x_j, v^-)$-run. Note that $x \in \Delta_T^x$ implies that $x_i \in \Delta_T^{x_i}$ for some $P \in R_K$. Thus, $x \in \Delta_T^x$ implies that $x_i \in \Delta_T^{x_i}$.

Item 1 in the definition of $\delta_V$ ensures that, in every run starting at $e$ and $s^V_0$, for each $v \in V_q$, (after checking that $v \notin L(e)$) the automaton moves to one individual node in state $v^+$, and to all remaining level one nodes in state $v^-$ ($A_V$ does not check that the labels $a \in I_K$ correctly identify a unique individual node, but this is verified by $A_T'$ so we can assume it is the case). From this and the items shown above, it follows that there exists an accepting $(e, s^V_0)$-run iff for each $v \in V_q$, there exists some individual node $i$ and exactly one $i \cdot w \in T$ with $w \in \mathbb{N}^*$ such that $v \in L(i \cdot w)$, and for this $i \cdot w$, we
have \(i \cdot w \in \Delta^\pi\). Moreover, \(v \notin L(x)\) for every \(x \in T, x \neq i \cdot x_v\). This shows that there exists an accepting \((e, z_0^\pi)\)-run iff \((e2)\) holds. □

To define the 2ATA \(A_\pi\) we use \(q\)-concepts. They are like regular concepts, but may use the individuals and variables in \(q\) as atomic concepts.

**Definition 5.9.** A \(q\)-concept \(C\) is defined as a regular \(\mathcal{ZI\mathcal{Q}}\) concept, but allows also the elements of \(V_q \cup I_q\) in place of atomic concepts. The semantics of a \(q\)-concept \(C\) in a extended interpretation \(\mathcal{J} = (I, \pi)\) is as follows:

- if \(C = A\) for some \(A \in C\), then \(C^{\mathcal{J}} = A^\pi\);
- if \(C = v\) for some \(v \in V_q \cup I_q\), then \(C^{\mathcal{J}} = \{\pi(v)\}\).

This inductively extends to complex \(q\)-concepts as in **Definition 2.3** (i.e., like for a regular interpretation). For each atom \(\alpha = R(v, v')\) in \(q\), we define the \(q\)-concept \(C_\alpha = \exists p_u^\ast \cdot (v \in \exists R.v')\), where \(p_u\) is the role introduced in step 2 of the normalization of \(K\), or if this \(p_u\) is not present in \(K\), then \(p_u = \bigcup_{p \in \mathcal{R}_K} P\). □

Intuitively, \(C_\alpha\) expresses that somewhere (reachable by any chain of roles) there is an object labeled \(v\), which is related via \(R\) to an object labeled \(v'\). Since the interpretations represented in extended interpretation trees are \(K\)-connected, and every node is reachable from a root, we have that \(I, \pi \models \alpha\) iff \(C_\alpha\) is satisfied at some root. Hence, we can check whether \(\pi\) is a match for \(q\) by checking the satisfaction of the \(q\)-concepts for its atoms at the roots.

**Example 5.10.** Recall the query \(q_g\) from **Example 2.5**, with atoms \(\alpha_1 = \text{HasParent}^* \circ \text{HasParent}^{-*}(v_1, v_2), \alpha_2 = \text{HasParent}^-(v_1, v_3), \alpha_3 = \text{HasParent}^-(v_2, v_3), \alpha_4 = \text{male}(v_1), \alpha_5 = \text{female}(v_2), \alpha_6 = \neg \text{deity}(v_1), \) and \(\alpha_7 = \neg \text{deity}(v_2)\). In order to have only binary atoms, we consider \(\alpha'_2 = \text{id(male)}(v_1, v_2), \alpha'_3 = \text{id(female)}(v_2, v_2), \alpha'_6 = \text{id(deity)}(v_1, v_1), \) and \(\alpha'_7 = \text{id}(\neg \text{deity})(v_2, v_2)\). The respective \(q\)-concepts are:

\[
C_{\alpha_1} = \exists p_u^\ast \cdot (v_1 \in \exists \text{HasParent}^* \circ \text{HasParent}^{-*}.v_2)
\]
\[
C_{\alpha_2} = \exists p_u^\ast \cdot (v_1 \in \exists \text{HasParent}^-.v_3)
\]
\[
C_{\alpha_3} = \exists p_u^\ast \cdot (v_2 \in \exists \text{HasParent}^-.v_3)
\]
\[
C_{\alpha'_2} = \exists p_u^\ast \cdot (v_1 \in \exists \text{id(male)}.v_1)
\]
\[
C_{\alpha'_3} = \exists p_u^\ast \cdot (v_2 \in \exists \text{id(female)}.v_2)
\]
\[
C_{\alpha'_6} = \exists p_u^\ast \cdot (v_1 \in \exists \text{id(deity)}.v_1)
\]
\[
C_{\alpha'_7} = \exists p_u^\ast \cdot (v_2 \in \exists \text{id}(\neg \text{deity)}.v_2)
\]

Consider the extended interpretation \(\mathcal{J} = (I_g, \pi)\) represented in Fig. 6, which has roots \(\{1, \ldots, 5\}\). Since the root label is \(L_R = \{\text{Father}_1, \text{Father}_3, \text{Father}_4, \text{Father}_8, \text{Father}_15, \text{HasMother}_21, \text{HasParent}_21, \text{HasParent}_34, \text{HasParent}_45, \text{HasParent}_51, \text{HasParent}_23\}\), we have that:

\[
(v_1 \in \exists \text{HasParent}^* \circ \text{HasParent}^{-*}.v_2)^\mathcal{J} = \{1\}
\]
\[
(v_1 \in \exists \text{id(male)}.v_1)^\mathcal{J} = \{1\}
\]
\[
(v_2 \in \exists \text{HasParent}^-.v_3)^\mathcal{J} = \{3\}
\]
\[
(v_2 \in \exists \text{id(female)}.v_2)^\mathcal{J} = \{3\}
\]
\[
(v_2 \in \exists \text{id}(\neg \text{deity}).v_2)^\mathcal{J} = \{3\}
\]

Hence \(C_{\alpha_2} = C_{\alpha'_2} = C_{\alpha_3} = C_{\alpha'_3} = C_{\alpha'_6} = C_{\alpha'_7} = \Delta^\mathcal{J}\), and in particular we have \(i \in C_{\alpha_2} = C_{\alpha'_2} = C_{\alpha_3} = C_{\alpha'_3} = C_{\alpha'_6} = C_{\alpha'_7}\) for every root \(i\) of \(I_g\). In contrast, \(C_{\alpha'_6} = \emptyset\), so \(i \notin C_{\alpha'_6}\) for every root \(i\). □

As discussed above, the following holds.

**Lemma 5.11.** For every extended canonical interpretation \(\mathcal{J} = (I, \pi)\) and atom \(\alpha = R(v, v')\) occurring in \(q\), we have \(I, \pi \models \alpha\) iff there is some root \(i \in \Delta^\mathcal{J}\) such that \(i \in C_\alpha^\mathcal{J}\).

**Proof.** (Only If) Assume \(I, \pi \models R(v, v')\). Then \((\pi(v), \pi(v')) \in R^\mathcal{J}\), and by the semantics of \(q\)-concepts, \(\pi(v) \in (\exists R.v')^\mathcal{J}\). As \(v^\mathcal{J} = (\pi(v))\) by definition, we get \(\pi(v) \in (v' \in \exists R.v')^\mathcal{J}\). Since \(I\) is a canonical interpretation, there is some \(i \in \text{Roots}(I)\) such that \(\pi(v)\) is \(K\)-connected to \(i\), that is, \((i, \pi(v)) \in (p_u^\ast)^\mathcal{J}\). By the semantics of \(q\)-concepts, this implies \(i \in (\exists p_u^\ast \cdot (v' \in \exists R.v'))^\mathcal{J}\). This, together with \(v^\mathcal{J} = (\pi(v))\) and \(\pi(v) \in (v' \in \exists R.v')^\mathcal{J}\) gives \(i \in (\exists p_u^\ast \cdot (v' \in \exists R.v'))^\mathcal{J}\) as desired.

(If) Assume \(i \in (\exists p_u^\ast \cdot (v' \in \exists R.v'))^\mathcal{J}\) for some \(i\). Then \((v' \in \exists R.v')^\mathcal{J} \neq \emptyset\). As \((v' \in \exists R.v')^\mathcal{J} \subseteq v^\mathcal{J} \) and \(v^\mathcal{J} = (\pi(v))\) by definition, we have \(\pi(v) \in (v' \in \exists R.v')^\mathcal{J}\). Hence \(\pi(v) \in (\exists R.v')^\mathcal{J}\), which by \((v')^\mathcal{J} = [\pi(v')]\) gives \((\pi(v), \pi(v')) \in R^\mathcal{J}\), so \((\pi(v), \pi(v')) \in R^\mathcal{J}\) and \(I, \pi \models R(v, v')\). □
Lemma 5.14. The proof is analogous to that of Lemma 4.14. The only difference is that in the base case and in the case of atomic negation, such that

Finally, we define \( A \) accepting run

Proof. By this lemma, we only need to verify that each of \( C_q \), \( \ldots \), \( C_q \) holds at some root for query atoms \( \alpha_1, \ldots, \alpha_q \) that make the query \( q \) true. The satisfaction of the concepts \( C_q \) is verified by an automaton \( A_q \) that decomposes them via transitions analogous to those of \( A_T \). Modulo the initial transition from the root node, \( A_q \) and \( A_T \) are very similar. We recall that \( q = \exists \varphi(\vec{v}) \), and that all atoms in \( \varphi \) are of the form \( R(v, v') \) with \( R \) a \( \exists I \mathcal{Q} \) role in NNF.

Definition 5.12. Let \( C_l = \bigcup_{\alpha \in \mathcal{A}(q)} C(\alpha) \). We define the 2ATA \( \mathcal{A}_q = (b_K, \Sigma_{K,q}, Q_q, \delta_q, \pi_q, F_q) \) as follows:

1. \( Q_q \) is like \( Q_T \) in \( A_T \), but defined using \( C_l \) and \( I_q \) instead of \( C(\mathcal{T}) \) and \( I_K \), and treating the symbols in \( V_q \) analogously to the concept names in \( C_l \), see Table 7.
2. \( F_q = (\emptyset, \{\forall R.C \mid \forall R.C \in C_l\}, Q_q) \), analogously to \( A_T \).
3. The transitions from the initial state are defined for each label \( \sigma \) containing \( r \) (identifying the root node) as

where \( B_\varphi \) results from \( \varphi(\vec{v}) \) by replacing each atom \( \alpha \) with \( (0, C_q) \).

When \( A_q \) is at the root node and in a state \( C_q \) for some \( \alpha \in \mathcal{A}(q) \), it checks that the concept \( C_q \) is satisfied at some individual node, via the following transitions, for each \( \alpha \in q \) and each \( \sigma \) containing \( r \):

To further check that \( C_q \) is satisfied, \( A_q \) has transitions similar to those of \( A_T \), viz. for each \( \sigma \in \Sigma_K \):
1. transitions that recursively decompose complex concepts and non-simple roles in \( C_l \), and that handle all simple roles in \( C_l \) and the states in \( Q_{\text{Self}} \) and \( Q_{\alpha, \text{role}} \), as in group (II) of \( \delta_T \);
2. transitions \( \delta_q(s, \sigma) \) for each \( s \in C_q \) of the form \( \exists n.S.C \), and for each \( s \in Q_{\text{Self}} \cup Q_{\alpha, \text{num}} \) to verify the satisfaction of the number restrictions, as in group (IV) of \( \delta_T \);
3. transitions \( \delta_q(s, \sigma) \) for each \( s \in C_q \cup I_q \cup \{p_j \mid p \in R_q, i, j \in \{1, \ldots, b_K\}\} \), as in group (III) of \( \delta_T \), and for each \( s \in V_q \cup I_q \), as in group 3 of \( \delta_T \), to check symbol occurrences in node labels.

Finally, we define \( A_q \) as a 2ATA accepting the intersection of \( A_T \) and \( A_T \). □

Given an extended interpretation tree, \( A_q \) correctly checks the satisfaction of complex concepts in \( C_l \).

Lemma 5.13. (Cf. Lemma 4.14.) Let \( T = (T, L) \) be an extended interpretation tree and let \( x \in \Delta_{I_T} \). Furthermore, let \( C \in C_l \). Then there is an accepting \((x, C)\)-run of \( A_q \) over \( T \) iff \( x \in C_{I_T} \).

Proof (Sketch). The proof is analogous to that of Lemma 4.14. The only difference is that in the base case and in the case of atomic negation, \( C \) may also be a possibly negated variable or individual \( t \in I_q \cup V_q \). In this case the last group of atomic transitions ensures that an accepting \((x, t)\)-run (resp., an accepting \((x, \neg t)\)-run) exists iff \( t \in L(x) \) (resp., \( t \notin L(x) \)). □

Hence a run of \( A_q \) on an extended interpretation tree that visits a state \( C_q \) correctly verifies the existence of a match for the atom \( \alpha \). By this and Lemma 5.11, the initial transition from the root is sufficient to verify whether this holds for a set of atoms that makes \( q \) true:

Lemma 5.14. An extended interpretation tree \( T = (T, L) \) is accepted by \( A_q \) iff there exists some \( B \subseteq \mathcal{A}(q) \) such that

(a) by assigning true to the atoms in \( B \) and false to those in \( \mathcal{A}(q) \setminus B \), \( \varphi \) evaluates to true, and
(b) for every \( \alpha \in B \), there is some root \( i \in \Delta_{I_T} \) such that \( i \in C_{\alpha_{I_T}} \).

Proof. The result follows straightforwardly from the definition of \( \delta_q \) and Lemma 5.13. Indeed, suppose that there is an accepting run \((T_f, r)\) of \( A_q \) over \( T \). Since \( \delta_q(s_0, \sigma) = B_\varphi \) is the only given transition from state \( s_0 \), there is a set \( B \subseteq \mathcal{A}(q) \) such that \( B \) satisfies condition (a), and the root of \( T_f \) has a child labeled \( (e, C_q) \) for each \( \alpha \in B \). By the transitions of the
form $\delta_T(C_0, a) = \bigvee_{i \in \mathcal{L}_C} \big(\delta_T(i, C_0) \land \bigwedge_{b \in \mathcal{L}_C} \delta_T(i, b)\big)$, each such child with label $(\varepsilon, C_0)$ must in turn have a child $(i, a)$ and a child $(i, C_0)$. Since the run is accepting, it must be the case that $a \in L(i)$, which implies that $i$ is a root of $T_T$, and that, by Lemma 5.13, $i \in C_0^{\mathcal{J}_T}$, so $B$ satisfies condition (b) as well. For the converse, assume a set $B$ as in the claim. Then we know that for each $a \in B$ there is a root $i_a$ of $T_T$ such that $i_a \in C_0^{\mathcal{J}_T}$. As $i_a$ is a root of $T_T$, there exists some $a \in L(i_a)$ and an accepting $(i_a, a)$-run $T_{i,a}$ of $A_T$ on $T$. We can obtain an accepting run of $A_T$ on $T$ by taking a root labeled $(\varepsilon, s_0)$, creating a child $y_0$ labeled $(\varepsilon, C_0)$ for each $a \in B$, and adding to each $y_0$ the whole $T_{i,a}$ as subruns (that is, adding descendants and labels as in the respective runs). This shows that an accepting run of $A_T$ over $T$ exists.

The following is a simple corollary of Lemmas 5.11 and 5.14:

**Corollary 5.15.** Let $T$ be an extended interpretation tree. Then $T \in \mathcal{L}(A_T)$ if and only if $\mathcal{J}_T \models q$.

Now we can easily show that $A_T$ accepts the trees representing interpretations where $q$ has a match:

**Proposition 5.16.** $\mathcal{L}(A_T) = \{T \mid T$ is an extended interpretation tree and $\mathcal{J}_T \models q\}$

**Proof (Sketch).** $(\subseteq)$ Assume $T$ is a $\Sigma_{K,q}$-labeled $b_K$-ary tree that $T \in \mathcal{L}(A_T)$. Then $T \in \mathcal{L}(A_T)$ by Lemma 5.8. $T$ is an extended interpretation tree. Furthermore, $T \in \mathcal{L}(A_T)$, so by Corollary 5.15 $\mathcal{J}_T \models q$.

$(\supseteq)$ Assume $T$ is an extended interpretation tree such that $\mathcal{J}_T \models q$. Then $T \in \mathcal{L}(A_T)$ by Lemma 5.8. As $\mathcal{J}_T \models q$, by Corollary 5.15 we have $T \in \mathcal{L}(A_T)$. Hence $T \in \mathcal{L}(A_T) \cap \mathcal{L}(A_T) = \mathcal{L}(A_T)$.

5.3. Deciding query entailment

Our algorithm for deciding $K \models q$ roughly works as follows (cf. Fig. 5). The automaton $A_T$ accepts a tree over the alphabet $\Sigma_{K,q}$, if it represents an extended canonical interpretation ($I$, $\pi$) for $K$ in which $\pi$ is a match for $q$. We project the query variables $V_q$ from $A_T$’s alphabet and obtain an automaton that accepts the same trees, but restricted to $\Sigma_{K,q}$; they correspond to the interpretation trees for $K$ in which $q$ has a match, no matter where it is. The next step is to complement this automaton, such that the resulting automaton $A_{\neg q}$ accepts an interpretation tree exactly when there is no match for $q$ in it. Finally, we intersect this automaton with the automaton $A_K$ to obtain an automaton $A_{K\neg q}$ that accepts the trees that represent a canonical model of $K$ in which $q$ has no match. By Theorem 3.10, $K \models q$ if and only if such a model exists. Hence, deciding $K \models q$ reduces to testing the automaton $A_{K\neg q}$ for emptiness. For easier reference, we summarize in Table 8 the characteristics of the different automata which the query entailment algorithm comprises (recall Fig. 5).

We now show in detail how $A_{K\neg q}$ can be obtained and analyze its size. We start by transforming $A_T$ into a 1NTA whose language we can project to the alphabet $\Sigma_{K,q}$. Note that the transformation to 1NTA causes an exponential blow-up in the number of states (Proposition 2.12), but after it the projection can be easily done (Proposition 2.14). Then we complement the resulting automaton in order to construct a 1NTA that accepts exactly the set of trees such that, if they represent an interpretation for $K$, it is an interpretation where $q$ has no match. The complementation also causes yet another exponential blow-up in the states, and it makes the index as large as the original state set (Proposition 2.13). More precisely, we have:

**Lemma 5.17.** Given $K$ and $q$, it is possible to construct a 1NTA $A_{\neg q}$ such that

1. If $T$ is an interpretation tree for $K$, then $T \in \mathcal{L}(A_{\neg q})$ if and only if $\mathcal{J}_T \models \neg q$.
2. $|\mathcal{L}(A_{\neg q})| \leq 2^{q^{O(|q|)}}$ and $\text{ind}(A_{\neg q}) \leq 2^{0(|q|)}$ for $n_q = |\mathcal{L}(A_{\neg q})|$ and some constant $c$, i.e., $A_{\neg q}$ has double exponentially many states and index single exponential in the number of states of $Q(A_T)$.

**Proof.** Let $A_0 = A_T$ and $n_q = |\mathcal{L}(A_T)|$. By Proposition 2.12, we can construct from $A_0$ a 1NTA $A_1$ with $\mathcal{L}(A_1) = \mathcal{L}(A_0)$ such that $|\mathcal{L}(A_1)| \leq 2^{0(n_q^0)}$ for some constant $c_0$ and $\text{ind}(A_1) = O(\text{ind}(A_0)) = O(1)$. By Lemma 2.14, we can transform $A_1$ into a
Lemma 5.18. There exists some $\Sigma_K$-projection of $L(A_1)$ such that $|Q(A_2)| \leq |Q(A_1)| \leq 2^{O(|Q(A_0)|^c)}$ and $\text{ind}(A_2) \leq \text{ind}(A_1) = O(1)$. By Lemma 2.13, we can construct from $A_2$ an automaton $A_3 = A_{\neg q}$ accepting the complement of $L(A_2)$ such that $|Q(A_3)| \leq 2^{O(|Q(A_2)|)}$ and $\text{ind}(A_3) = O(\log |Q(A_2)|)$, where $f(A_2) = \text{ind}(A_2) - |Q(A_2)| \cdot \log |Q(A_2)| \leq 2^{O(|Q(A_0)|^c)}$, for some constant $c$. It follows that $|Q(A_{\neg q})| \leq 2^{O(n_2^c)}$ and $\text{ind}(A_{\neg q}) \leq 2^{O(n_2^c)}$ for some constant $c$. This shows the second item.

For the first item, we know from Proposition 5.16 that $L(A_{\neg q}) = L(A_1) = |T| \mid T$ is an extended interpretation tree and $\mathcal{F}(\pi = q)$. By construction, $A_2$ accepts the $\Sigma_K$-projection of $L(A_1)$, i.e., $L(A_2) = |T|_{\Sigma_K} \mid T$ is an extended interpretation tree and $\mathcal{F}(\pi = q)$. Recall that, by definition, $T_{\Sigma_K}$ is an interpretation tree. Moreover, for an arbitrary interpretation tree $T$ such that there exists some $\pi : V_q \cup I_q \to \Delta^2$ with $T_{\Sigma_K} \mathcal{F}(T_{\Sigma_K} \pi)$, we have that $\mathcal{F}(\pi = q)$ is an extended canonical interpretation and there is an extended interpretation tree $T$ such that $\mathcal{F}(\pi = q)$ (obtained by adding in $T$ the variable $v$ to the label of the node $\pi(v)$, for all query variables $v$). This implies that $T$ is accepted by $A_2$. Hence, we can equivalently describe $L(A_2)$ as $\{T \mid T$ is an interpretation tree and there exists some $\pi : V_q \cup I_q \to \Delta^2$ with $T_{\Sigma_K} \mathcal{F}(T_{\Sigma_K} \pi)\}$. Then the complement $A_3 = A_{\neg q}$ of $A_2$ accepts an interpretation tree $T$ iff there exists no $\pi : V_q \cup I_q \to \Delta^2$ with $T_{\Sigma_K} \mathcal{F}(T_{\Sigma_K} \pi)$, i.e., $\mathcal{F}(\pi = q)$. This shows the first item. $\square$

We can now transform $A_K$ into a 1NTA and interject it with $A_{\neg q}$, to obtain the automaton $A_{K,\neg q}$. It accepts exactly the canonical models of $K$ for which there is no match for $q$, as desired.

Lemma 5.18. Given $K$ and $q$, it is possible to construct via $A_K$ and $A_{\neg q}$ a 1NTA $A_{K,\neg q}$ such that:

1. $L(A_{K,\neg q}) = \{T \mid T$ is an interpretation tree such that $T_{\Sigma_K} \mathcal{F}(\pi = q)$ and $T_{\Sigma_K} \mathcal{F}(\pi = \neg q)\}$,
2. $|Q(A_{K,\neg q})| \leq 2^{O(|Q(A_K,\neg q)|^{c/2})}$ and $\text{ind}(A_{K,\neg q}) \leq 2^{O(n_2^c)}$ for some constant $c$ where $n_K = |Q(A_K)|$ and $n_q = |Q(A_0)|$, i.e., $A_{K,\neg q}$ has double exponentially many states in the number of states of $A_K$ and $A_{\neg q}$, and index single exponential in the number of states of $A_q$.

Proof. By Proposition 2.12 we can construct from $A_K$ a 1NTA $A_1$, with $|Q(A_1)| \leq 2^{O(|Q(A_K)|^{c/2})}$ for some constant $c_1$ and $\text{ind}(A_1) = O(\text{ind}(A_K)) = O(1)$, such that $L(A_K) = L(A_1)$. We then construct a 1NTA $A_3 = A_{K,\neg q}$ as the intersection of $A_1$ and $A_2 = A_q$, which by Lemma 2.15 has

\[ \text{ind}(A_3) = O(f(A_1, A_2)), \]

\[ |Q(A_3)| \leq 2^{O\left(f(A_1, A_2)^c\right)} \cdot f(A_1, A_2) \cdot |Q(A_1)| \cdot |Q(A_2)|, \]

where $f(A_1, A_2) = \text{ind}(A_1) + \text{ind}(A_2) + 1$. Since by Lemma 5.17, $|Q(A_2)| \leq 2^{O(n_2^c)}$ and $\text{ind}(A_2) \leq 2^{O(n_2^c)}$ for some constant $c_2$, we have that

\[ \text{ind}(A_3) \leq 2^{O(n_2^c)} + O(1), \]

\[ |Q(A_3)| \leq 2^{O\left(f(A_1, A_2)^c\right)} \cdot f(A_1, A_2) \cdot 2^{O|Q(A_K)|^{c_1}} \cdot 2^{O(n_2^c)}, \]

where $f(A_1, A_2) = \text{ind}(A_1) + \text{ind}(A_2) + 1 \leq 2^{O(n_2^c)} + O(1) + 1$. It follows from this that $|Q(A_{K,\neg q})| \leq 2^{O(n_K + n_q q^c)}$ and $\text{ind}(A_{K,\neg q}) \leq 2^{O(n_2^c)}$ for some constant $c$. This shows the second item. Now we show the first item, i.e., that $L(A_{K,\neg q}) = L(A_K) \cap L(A_{\neg q})$ is the set of trees representing models of $K$ where $q$ has no match. (\(\subseteq\)) Suppose $T \in L(A_{K,\neg q})$. Then $T \in L(A_K)$, which by Proposition 4.17 means that $T$ is an interpretation tree and $T_{\Sigma_K} \mathcal{F}(\pi = K)$. We also have $T \in L(A_{\neg q})$, which by Lemma 5.17 means that $T_{\Sigma_K} \mathcal{F}(\pi = \neg q)$. (\(\supseteq\)) Conversely, suppose $T$ is an interpretation tree such that $T_{\Sigma_K} \mathcal{F}(\pi = K) \text{ and } T_{\Sigma_K} \mathcal{F}(\pi = \neg q)$. Then by Proposition 4.17 $T \in L(A_K)$, and by Lemma 5.17 $T \in L(A_{\neg q})$. This implies that $T \in L(A_K) \cap L(A_{\neg q}) = L(A_{K,\neg q})$. $\square$

Therefore we can decide whether $K \models q$ by testing $A_{K,\neg q}$ for emptiness.

Theorem 5.19. For every normalized P2RPQ $q$ over a normalized KB $K$ in ACCQITbreg, it holds that $K \models q$ iff $L(A_{K,\neg q}) = \emptyset$.

Proof. It follows from Lemma 5.18 that if $L(A_{K,\neg q}) \neq \emptyset$, there exists some interpretation $I$ such that $I \models K$ and $I \models q$, so $K \models q$. For the converse, if $K \models q$, by Theorem 3.10 and our assumption that $b_K > k_T q$ (see the beginning of Section 5), there exists some $b_K$-canonical interpretation $I$ such that $I \models K$ and $I \models q$. Since by Lemma 4.5 $I = I_{F} T$, we can apply Lemma 5.18 to conclude that $T_{I} \in L(A_{K,\neg q})$ hence $L(A_{K,\neg q}) \neq \emptyset$. $\square$

5.4. Complexity

We now show that the reduction of query entailment to automata emptiness in Theorem 5.19 gives a tight upper complexity bound for the problem. By $|q|$ we denote the size of a (string) representation of a query $q$, and by $\|K,q\| = \|K\| + |q|$ the combined size of a KB $K$ and $q$ (assuming unary number encoding).
It is not difficult to show that \( A_q \) has polynomially many states in \( \| K, q \| \), and a short acceptance condition (i.e., its index is a small constant).

**Lemma 5.20.** \( |Q(A_q)| = O(\| K, q \|^c) \), for some constant \( c \), and \( \text{ind}(A_q) = 3 \).

**Proof.** Recall that under unary number encoding in \( K \) (cf. Section 4.4), \( b_K \) is quadratic in \( \| K \| \) (and hence in \( \| K, q \| \)). Likewise for \( q \) we have that \( |C_lq|, |V_q|, \) and \( n_{\text{max}}^q = \max(n, n.S.C \in C_lq) \cup \{0\} \) are linear in \( \| q \| \) (and in \( \| K, q \| \)). As \( A_q = A_r \cap A_T = A_r \cap (A_V \cap A_{\lambda}^2) \), it follows from Proposition 2.10 that \( |Q(A_q)| \leq |Q_r| + |Q_V| + |Q_T| + 2 \), where:

\[
\begin{align*}
|Q_T| &\leq 2 + 2(|I_K| + 1) \\
|Q_r| &\leq 1 + 4|V_q| \\
|Q_V| &\leq 1 + |C_r^l| + |Q_q.,A.,role| + |Q_q.,num| + |Q_q.,A.,num| \\
|C_r^l| &\leq |C_lq| + 2(|I_K| + |V_q|) \\
|Q_q.,set| &\leq |C_lq| \\
|Q_q.,A.,role| &\leq |C_lq| \cdot |b_K|^2 \\
|Q_q.,num| &\leq |C_lq| \cdot (b_K + 1) \cdot (n_{\text{max}}^q + 1) \\
|Q_q.,A.,num| &\leq |I_K| \cdot |C_lq| \cdot b_K \cdot n_{\text{max}}^q
\end{align*}
\]

Hence, \( |Q(A_q)| = O(\| K, q \|^3) \). Moreover, by Proposition 2.10, \( \text{ind}(A_q) = \max(\text{ind}(A_r), \text{ind}(A_T), \text{ind}(A_{\lambda}^2)) = 3 \).  

We next establish that the number of states of \( A_{K,q} \) is double exponential in \( \| K, q \| \).

**Lemma 5.21.** \( |Q(A_{K,q})| = 2^{O(|K,q|^3)} \) and \( \text{ind}(A_{K,q}) = 2^{O(|K,q|^3)} \) for some constant \( c \). Furthermore, \( A_{K,q} \) can be constructed in time double exponential in \( \| K, q \| \).

**Proof.** The first part follows from Lemmas 4.19, 5.18, and 5.20. The second part holds since in all the automata constructions given in Section 5.3, the time required to construct an automaton is polynomial in its size plus the size of the input.  

Using Proposition 2.16, we obtain the main result of this section.

**Theorem 5.22.** Given a P2RPQ \( q \) over a KB \( K \in \text{ALC QI}_{\text{reg}} \) and a P2RPQ \( q \) over \( K \), deciding whether \( K \models q \) is in 2ExpTime under unary number encoding.

This bound is worst case optimal. Indeed, it was shown in [46] that answering CQs over KBs in \( \text{ALCQI} \) (i.e., P2RPQs built only with conjunction \( \land \), where no regular role expressions, but only concept and role names are allowed) is 2ExpTime-hard, and the same lower bound was established in [47] for \( \text{SH} \), a DL with transitive roles but lacking inverse roles. It is not hard to see that the proof for \( \text{SH} \) in [47] can be adapted to show hardness already for \( \text{ALC} \), for two restricted classes of P2RPQs that generalize CQs: (i) positive queries (PQs), which allow conjunction and disjunction, but atoms contain only concept and role names and no regular role expressions, and (ii) conjunctive RQPs (CRQs), which do not allow for either disjunctions between atoms or inverse roles in the regular expressions (this latter lower bound was recently observed in [58]). Hence we obtain the following result.

**Theorem 5.23.** Let \( L_K \subseteq ZIQ \) be a DL, \( L_q \subseteq P2RPQs \) a query language, \( q \) in \( L_q \), and \( K \in L_K \). Then deciding \( K \models q \) is 2ExpTime-complete under unary number encoding if any of the following holds:

1. \( L_q \subseteq \text{CQs} \), and either \( \text{ALCQI} \subseteq L_K \) or \( \text{SH} \subseteq L_K \).
2. \( \text{ALC} \subseteq L_K \), and either \( L_q \subseteq \text{CRQs} \) or \( L_q \subseteq \text{PQs} \).

### 6. Complex role inclusion axioms

The DL \( SRIQ \) was introduced in [5] as an extension of \( RIQ \) [59], which in turn extends the well-known DL \( SHIQ \) [60] underlying OWL-Lite. \( SRIQ \) has gained considerable attention in the last years as the nominal-free fragment of the DL \( SROIQ \) underlying the new OWL 2 standard [3]. In this section, we show that our algorithm can also be utilized for query answering in \( SRIQ \) by means of a suitable reduction to the logic \( ZIQ \).
The most prominent feature of \( SRIQ \) are complex role inclusion axioms of the form \( R_1 \circ \cdots \circ R_n \sqsubseteq R \) subject to some regularity restrictions. The latter, which are necessary to guarantee decidability of reasoning, make it also possible to simulate such axioms with regular expressions. \( SRIQ \) also allows one to explicitly state certain properties of roles, including (ir)reflexivity, symmetry, and disjointness, which can be simulated in \( ZIQ \) using BRIAs and CIAs. To recall \( SRIQ \) KBs, we follow [5] and [61]. Let \( \bar{R} = R \cup \{ R^\sim | R \in R \} \).

Definition 6.1 \( (SRIQ \text{ knowledge base}) \). A SRIQ role inclusion axiom (SRIA) is an expression of the form \( R_1 \circ \cdots \circ R_n \sqsubseteq R \), \( n \geq 1 \), where \( R \) and all \( R_i \), \( 1 \leq i \leq n \), are from \( \bar{R} \), and different from \( T \) if \( n > 1 \).

A set \( \mathcal{R} \) of SRIAs is regular, if there exists a strict partial order \( < \) on \( \bar{R} \) such that

(i) for every \( R, R' \in \mathcal{R} \), \( \text{inv}(R) < R' \) iff \( R < R' \), and
(ii) every SRIA in \( \mathcal{R} \) is of one of the forms (a) \( R \circ R \sqsubseteq R \), (b) \( \text{inv}(R) \sqsubseteq R \), (c) \( w \sqsubseteq R \), (d) \( w \circ R \sqsubseteq R \), or (e) \( R \circ w \sqsubseteq R \), where \( w = R_1 \circ \cdots \circ R_n \) and \( R_i < R \) for each \( 1 \leq i \leq n \).

For a given set \( \mathcal{R} \) of SRIAs, the relation \( \sqsubseteq_{\mathcal{R}} \) is the smallest relation such that

(i) \( R \sqsubseteq_{\mathcal{R}} R \) for every \( R \in \mathcal{R} \) such that either \( R \) or \( \text{inv}(R) \) occurs in \( \mathcal{R} \), and
(ii) if \( w_1 \circ R \circ w_2 \sqsubseteq_{\mathcal{R}} R \), where \( w_1 = R_1 \circ \cdots \circ R_{n_1} \), \( n_1 \geq 0 \) for \( i = 1, 2 \) and \( \mathcal{R} \) contains either \( w \sqsubseteq R' \) or \( \text{inv}(w) \sqsubseteq R' \), where \( w = R_1 \circ \cdots \circ R_n \), \( n \geq 1 \) and \( \text{inv}(w) = \text{inv}(R_1) \circ \cdots \circ \text{inv}(R_n) \), then \( w_1 \circ w \circ w_2 \sqsubseteq_{\mathcal{R}} R \) (i.e., \( \sqsubseteq_{\mathcal{R}} \) is closed with respect to unfolding roles on the left hand side with SRIAs).

A role is simple in \( \mathcal{R} \), if no roles \( R_1, \ldots, R_n, n \geq 2 \), exist such that \( R_1 \circ \cdots \circ R_n \sqsubseteq_{\mathcal{R}} R \).

An assertion about roles is an expression of the form \( \text{Sym}(R) \), \( \text{Ref}(R) \), or \( \text{inv}(R) \), or \( \text{Dis}(R, R') \), for roles \( R, R' \in \bar{R} \). It is simple w.r.t. to a set \( R \) of SRIAs, if all roles occurring in it are simple in \( R \) or it is of the form \( \text{Sym}(R) \).

A SRIQ TBox is a finite set \( \mathcal{R} = \mathcal{R}^t \cup \mathcal{R}^d \) of SRIAs \( \mathcal{R}^i \) and assertions about roles \( \mathcal{R}^d \) such that \( \mathcal{R}^i \) is regular and each assertion in \( \mathcal{R}^d \) is simple w.r.t. to \( \mathcal{R}^i \). To define SRIQ TBoxes and ABoxes, we assume a given SRIQ RBox containing the set \( \mathcal{R}^i \) of SRIAs. Then SRIQ concepts \( C, C' \) obey the following syntax:

\[
C, C' \rightarrow A | \neg C | C \cap C' | C \cup C' | \forall R.C | \exists R.C | \geq n.S.C | \leq n.S.C | \exists S.self,
\]

where \( A \in \mathcal{C}, R, S \in \mathcal{R} \), and \( S \) is simple in \( R \). A SRIQ concept inclusion axiom (SCIA) is an expression \( C \sqsubseteq C' \) for arbitrary SRIQ concepts \( C, C' \); a SRIQ TBox is a set of SCIs. A SRIQ assertion is an expression \( C(a), R(a, b), \neg S(a, b), \) or \( a \neq b \), where \( C \) is a SRIQ concept, \( S, R \) are SRIQ roles, \( S \) is simple in \( R \), and \( a, b \in I \); a SRIQ ABox is a set of SRIQ assertions. Finally, a SRIQ knowledge base is a triple \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \) where \( \mathcal{A} \) is a non-empty ABox, \( \mathcal{T} \) is a TBox, and \( \mathcal{R} \) is a SRIQ RBox as above. \( \square \)

The semantics of SRIQ TBoxes and ABoxes is defined analogously to that of ZIQ. An interpretation \( I \) satisfies an assertion about roles \( \text{Sym}(R) \), \( \text{Ref}(R) \), or \( \text{inv}(R) \), if \( R^2 \) is symmetric, reflexive, or irreflexive, respectively; \( I \) satisfies \( \text{Dis}(R, R') \), if the relations \( R^i \) and \( R'^j \) are disjoint, i.e., \( R^i \cap R'^j = \emptyset \); \( I \) satisfies a SRIA \( R_1 \circ \cdots \circ R_n \sqsubseteq R \) if \( R^i_1 \circ \cdots \circ R^i_n \subseteq R^i \) (where again the symbol \( \circ \) denotes composition of binary roles). An interpretation \( I \) satisfies (or is a model of) an RBox \( \mathcal{R} \), if it satisfies all SRIAs and all assertions about roles in \( \mathcal{R} \), written \( I \models \mathcal{R} \); it satisfies (or is a model of) a SRIQ KB \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \), denoted \( I \models \mathcal{K} \), if \( I \models \mathcal{A} \) and \( I \models \mathcal{T} \) and \( I \models \mathcal{R} \).

Example 6.2. In Fig. 7, we give part of a genealogy KB in SRIQ syntax. The first group in the left column is a TBox \( T^i_1 \) consisting of four SCIs. They are almost identical to (some of) the CIAs of the KB \( K^d \) from Example 2.5, but in \( \text{HasParent} \sqsubseteq \forall \text{HasParent}*.\text{deity} \) the complex \( ZIQ \) role \( \text{HasParent}^* \), which is not allowed in SRIQ, has been replaced by the

\footnote{We use the term assertion about roles instead of role assertions as in [5], since the latter is often used to refer to ABox assertions of the form \( R(a, b) \). In [5] also the assertion \( \text{Trans}(R) \), stating that \( R \) is transitive, is allowed. We omit this as it is expressible with the SRIA \( R \circ R \sqsubseteq R \).}

\footnote{As in Definition 2.2, we only consider w.l.o.g. non-empty ABoxes.}
role hasAncestor. In the second group in the left column we give a set $\mathcal{R}_g^a$ of (two) assertions about roles, and the right column contains a set $\mathcal{R}_h^i$ of SRIAs.

6.1. Reducing $\mathcal{SRIQ}$ to $\mathcal{ZIQ}$

We describe a rewriting that transforms a $\mathcal{SRIQ}$ KB $\mathcal{K}$ into a $\mathcal{ZIQ}$ KB $\Psi(\mathcal{K})$ in a way that will allow us to exploit our automata-based algorithms for reasoning in $\mathcal{SRIQ}$. It builds on the fact that the restriction to regular sets of SRIAs, which is crucial for the decidability of $\mathcal{SRIQ}$, ensures that the implications between roles define a regular language. Hence they can be simulated using the regular expressions over roles present in $\mathcal{ZIQ}$. More precisely:

**Lemma 6.3.** (See [59].) If $\mathcal{R}$ is a regular set of SRIAs, then for each $R \in \mathcal{R}$ occurring in $\mathcal{K}$, the set $L_R(R) = \{R_1 \circ \cdots \circ R_n \mid R_1 \circ \cdots \circ R_n \subseteq \mathcal{R}\}$ is a regular language.

Furthermore, the authors of [59] show how to construct a finite state automaton representing $L_R(R)$, which is equivalent to a regular expression $\rho^R(R)$ over the alphabet $\mathcal{R}$. That is, $L_R(R)$ can be written as a $\mathcal{ZIQ}$ role $\rho^R(R)$. In particular, if a role $S$ is simple in $\mathcal{R}$, the resulting expression is $\rho^R(S) = \bigcup_{S \in \mathcal{S}} S'$, which is a simple $\mathcal{ZIQ}$ role. It is easy to see that $R^I \subseteq (\rho^R(R))^I$ in every interpretation $I$, since $R \subseteq \mathcal{R}$. Hence, by suitably interpreting the added $R_n$, we can construct a $\mathcal{ZIQ}$ role $\rho^R(R)$ such that $(\rho^R(R))^I = \bigcup_{R_1 \circ \cdots \circ R_n \subseteq \mathcal{R}} R^I$ in every interpretation $I$, and hence $(\rho^R(R))^I = R^I$ whenever $I \models \mathcal{R}$. Moreover, $\rho^R(R)$ is simple whenever $R$ is simple in $\mathcal{R}$.

**Corollary 6.4.** Given a regular set $\mathcal{R}$ of SRIAs and $R \in \mathcal{R}$, we can construct a $\mathcal{ZIQ}$ role $\rho^R(R)$ such that $(\rho^R(R))^I = \bigcup_{R_1 \circ \cdots \circ R_n \subseteq \mathcal{R}} R^I$ in every interpretation $I$, and hence $(\rho^R(R))^I = R^I$ whenever $I \models \mathcal{R}$. Moreover, $\rho^R(R)$ is simple whenever $R$ is simple in $\mathcal{R}$.

**Example 6.5.** Given the SRIAs $\mathcal{R}_h^i$ in Fig. 7, we can construct the following $\mathcal{ZIQ}$ roles for the role names in $\mathcal{R}_h^i$: $\rho^R(\text{HasMother}) = \text{HasMother}$, $\rho^R(\text{HasFather}) = \text{HasFather}$, $\rho^R(\text{HasParent}) = \text{HasMother} \cup \text{HasFather} \cup \text{HasParent}$, and $\rho^R(\text{hasAncestor}) = (\text{HasMother} \cup \text{HasFather} \cup \text{HasParent} \cup \text{hasAncestor})^+$. □

In what follows, we assume a fixed regular set $\mathcal{R}$ of SRIAs, and for each $R \in \mathcal{R}$, $\rho^R(R)$ is an arbitrary but fixed regular expression as above. The rewriting $\Psi$ of a $\mathcal{SRIQ}$ KB exploits Lemma 6.3 above, and has the following steps:

1. **TBox rewriting.** In every concept occurring in the TBox, we replace the role $R$ by the regular expression $\rho^R(R)$. We show below that this ensures that the interpretation of concepts in the rewritten TBox respects the restrictions that arise from the SRIAs.

2. **ABox rewriting.** Similarly as above, we replace in every concept assertion of the form $\Psi(a)$ each role $R$ occurring in $\mathcal{C}$ by $\rho^R(R)$. We furthermore remove the negated role membership assertions $\neg \Psi(a, b)$, which are not allowed in $\mathcal{ZIQ}$, and simulate them using a fresh role name $P_{-\Psi}$ for each role $\Psi$. We replace $\neg \Psi(a, b)$ by $P_{-\Psi}(a, b)$, and add BRIAs which ensure that $P_{-\Psi}$ is interpreted as a role that is disjoint from $\Psi$.

3. **RBox rewriting.** SRIAs are dropped, and assertions about roles are simulated using BRIAs and CIAs.

More formally, the rewriting $\Psi(\mathcal{K})$ is as follows.

**Definition 6.6.** Consider a $\mathcal{SRIQ}$ KB $\langle \mathcal{T}, \mathcal{R} \cup \mathcal{R}^a, \mathcal{A} \rangle$, where $\mathcal{R}^a$ is a set of assertions about roles.

(1) For a $\mathcal{SRIQ}$ concept $C$, we denote by $\Psi^R(C)$ the $\mathcal{ZIQ}$ concept that results from replacing every occurrence of a role $R$ by $\rho^R(R)$. Then $\Psi^R(\mathcal{T})$ is the $\mathcal{ZIQ}$ TBox $\{\Psi^R(C) \subseteq \rho^R(D) \mid C \subseteq D \in \mathcal{T}\}$.

(2) $\Psi^R(\mathcal{A})$ is the $\mathcal{ZIQ}$ ABox obtained by replacing in $\mathcal{A}$ (i) each assertion $\Psi(a)$ by $\Psi^R(C)(a)$, and (ii) each assertion $\neg \Psi(a, b)$ by $P_{-\Psi}(a, b)$, for a fresh role name $P_{-\Psi}$. $\mathcal{T}_{\mathcal{A}}^R$ is the TBox containing $P_{-\Psi} \cap \rho^R(S) \subseteq B$ for each $\neg \Psi(a, b)$ in $\mathcal{A}$ (note that $\rho^R(S)$ is a simple role).

(3) $\Psi^R(\mathcal{R}^a)$ is the $\mathcal{ZIQ}$ TBox

$$\Psi^R(\mathcal{R}^a) = \{\text{Inv}(\rho^R(R)) \subseteq \mathcal{R} \mid \text{Sym}(R) \in \mathcal{R}^a\} \cup \{\top \subseteq \exists R. \text{Self} \mid \text{Ref}(R) \in \mathcal{R}^a\} \cup \{\rho^R(R) \cap \rho^R(R') \subseteq B \mid \text{Dis}(R, R') \in \mathcal{R}^a\},$$


where the notation Inv(R) (defined for simple roles in Section 3.2) is extended to arbitrary $\mathcal{ZIQ}$ roles as follows: $\text{Inv}(R \cup R') = \text{Inv}(R) \cup \text{Inv}(R')$, $\text{Inv}(R \circ R') = \text{Inv}(R') \circ \text{Inv}(R)$, $\text{Inv}(\rho(R)) = \text{Inv}(\rho(R))^*$, and $\text{Inv}(\text{id}(C)) = \text{id}(C)$.

(4) Finally, $\Psi(\mathcal{K})$ is the $\mathcal{ZIQ}$ KB $\langle \Psi^R(\mathcal{T}) \cup \mathcal{T}_{\mathcal{A}}^R \cup \Psi^R(\mathcal{R}^a), \Psi^R(\mathcal{A}) \rangle$. □

The rewriting of ABoxes and TBoxes preserves the semantics in the models of $\mathcal{R}$. The final step, rewriting assertions about roles, has the stronger property of preserving equivalence in all models. Hence, by suitably interpreting the added
role names $P_{=5}$, all models of $\Psi (K)$ become models of $K$. The converse holds only in a slightly weaker form, as the SRIAs of $K$ need not be satisfied in every model $I$ of $\Psi (K)$. However, each such $I$ can be transformed into a model of the SRIAs by adding all implied pairs of individuals to the extension of the roles.

**Proposition 6.7.** Let $K = (T, R \cup R^0, A)$ be a SRIQ KB as above. Then:

1. For every interpretation $I$, if $I \models K$, then there is an interpretation $I'$ such that $I' \models \Psi (K)$, and $I'$ coincides with $I$ on all concepts and roles over the signature of $K$.

2. Let $I$ be an interpretation such that $I \models \Psi (K)$, and let $I'$ be the interpretation that has $R^I = (\rho R^0 (R))^I$ for each role $R$ in $R$. Then $I' \models K$.

**Proof.** For the first item, we start by proving the following three claims, for an arbitrary interpretation $I$:

(i) $I \models R$ implies $C^I = (\Psi^R (C))^I$ for each SRIQ concept $C$; hence, if $I \models R$ and $I \models T$ then $I \models \Psi^R (T)$.

(ii) If $I \models R$ and $I \models A$, then there is an interpretation $I'$ such that $I' \models \Psi^R (A)$, and $I'$ coincides with $I$ on all concepts and roles over the signature of $K$.

(iii) If $I \models R$ and $I \models R^0$, then $I \models \Psi^R (R^0)$.

Assume $I \models R$. To show item (i), we start by showing that $C^I = (\Psi^R (C))^I$ by structural induction on $C$. The claim is trivial for atomic concepts and for concepts of the form $\neg D$, $D \land D'$, and $D \lor D'$, since $\Psi^R (C)$ coincides with $\neg \Psi^R (D)$, $\Psi^R (D) \land \Psi^R (D')$, and $\Psi^R (D) \lor \Psi^R (D')$, respectively. For $C$ of the form $\exists R.D$, we have $\Psi^R (C) = \exists \rho R^0 (R) \Psi^R (D)$. Then $(\exists R.D)^I \subseteq (\exists \rho R^0 (R) \Psi^R (D))^I$ follows from the induction hypothesis and $R^I \subseteq (\rho R^0 (R))^I$. For the converse, $(\exists \rho R^0 (R) \Psi^R (D))^I \subseteq (\exists R.D)^I$, we need to use the induction hypothesis and the assumption that $I \models R$ to infer $(\rho R^0 (R))^I \subseteq R^I$. The cases for $C$ of the forms $\geq n R.D$ and $\exists R.S$ are analogous. For $C$ of the form $\forall R.D$ or $\leq n R.D$ the proof is similar, but now $\Psi^R (C) \subseteq C^I$ follows directly from the induction hypothesis and $R^I \subseteq (\rho R^0 (R))^I$, while showing $C^I \subseteq \Psi^R (C)$ requires the assumption that $I \models R$ and hence $(\rho R^0 (R))^I \subseteq R^I$. Once we have shown $C^I = (\Psi^R (C))^I$, it follows that $I \models T$ implies $I \models \Psi^R (T)$, and item (i) holds.

For item (ii), assume $I \models R$ and $I \models A$. We let $I'$ be the interpretation that has $R_{=5}^I = (\Delta^I \times \Delta^I) \setminus S^I$ for each fresh role name $R_{=5}$ in $\Psi^R (A)$, and is identical to $I$ otherwise. As $I$ and $I'$ coincide on all symbols in $K$, $I' \models R$ and $I' \models A$. To show $I' \models \Psi^R (A)$, we only need to show that $I'$ satisfies each assertion from $\Psi^R (A)$. We distinguish three cases:

- The assertion is of the form $\Psi^R (C)(a)$, and it replaced some $C(a) \in A$ in step (2i). As $I' \models R$, $C^I = \Psi^R (C)^I$, and as $I' \models A$ and $C(a) \in A$, $a^I \in C^I$. Hence $a^I \in \Psi^R (C)^I$ and $I'$ satisfies $\Psi^R (C)(a)$.

- If the assertion is of the form $R(a, b)$, then $R(a, b) \in A$ (i.e., it is an assertion from the original ABox). Then $I'$ satisfies $R(a, b)$ because $I' \models A$.

- If the assertion is of the form $\neg S(a, b)$, then it replaced some $S(a, b) \in A$ in step (2ii). Since $I \models A$ and $\neg S(a, b) \in A$, it follows that $(a^I, b^I) \notin S^I$, and hence $(a^I, b^I) \in P_{=5}^I$ by construction of $I'$. Hence $I'$ satisfies $P_{=5}^I(a, b)$.

This proves $I' \models \Psi^R (A)$. To show $I' \models T^A$, consider an arbitrary $P_{=5} \cap R(S)^I \subseteq B$ with $S(a, b) \in A$. Since $I \models R$, by **Corollary 6.4** $(x, y) \in \rho R^0 (S)^I$ implies $(x, y) \in S^I$, and by construction of $I'$, $(x, y) \notin P_{=5}^I$. As $\rho R^0 (S)^I = \rho R^0 (S)^I$, this shows that $(x, y) \in \rho R^0 (S)^I$ implies $(x, y) \notin P_{=5}^I$ for every pair $(x, y) \in \Delta^I \times \Delta^I$, so $I'$ satisfies $P_{=5} \cap R(S)^I$ and $I' \models T^A$ as desired.

For item (iii), we assume $I \models R$ and $I \models R^0$, and show that $I$ satisfies the different forms of inclusions in $\Psi^R (R^0)$:

- $\text{Inv}(\rho R^0 (R)^I) \subseteq R$ with $\text{Sym}(R) \in R^0$. Consider an arbitrary $(x, y) \in \text{Inv}(\rho R^0 (R)^I)$. Then $(y, x) \in \rho R^0 (R)^I$. Since $I \models R$, from **Corollary 6.4** we have $(y, x) \in R^I$. From this and the fact that $I$ satisfies $\text{Sym}(R)$, it follows that $(x, y) \in R^I$ and $I$ satisfies the inclusion as desired.

- $T \subseteq \exists R. \text{Self}$ with $\text{Rel}(R) \in R^0$. Since $I$ satisfies $\text{Rel}(R)$, $\exists R. \text{Self}^I = \Delta^I$ and $I$ satisfies $T \subseteq \exists R. \text{Self}$ as desired.

- $\exists \rho R^0 (R). \text{Self} \subseteq \bot$ with $\text{Inv}(R) \in R^0$. This case is similar to the previous one: since $I$ satisfies $\text{Inv}(R)$, $\exists R. \text{Self}^I = \emptyset$. Since $I \models R$, from **Corollary 6.4** it follows that $\exists \rho R^0 (R). \text{Self}^I = \emptyset$, and $I$ satisfies $\exists \rho R^0 (R). \text{Self} \subseteq \bot$ as desired.

- $P_{=5} \cap \rho R^0 (R)^I \subseteq B$ with $\text{Dis}(R, R') \in R^0$. We consider an arbitrary pair $(x, y) \in \rho R^0 (R)^I$. Since $I \models R$, from **Corollary 6.4** it follows that $(y, x) \in R^I$. Since $I$ satisfies $\text{Dis}(R, R')$, this means $(x, y) \notin R^I$. We can now use again **Corollary 6.4** to infer $(x, y) \notin \rho R^0 (R)^I$. Hence $I$ satisfies $\rho R^0 (R) \cap \rho R^0 (R') \subseteq B$.

This concludes the proof of item (iii). Item (1) in the claim now follows easily: consider an arbitrary $I$ such that $I \models K$, and let $I'$ be as in the proof of item (ii), i.e., $(P_{=5}^I) = (\Delta^I \times \Delta^I) \setminus S^I$ for each fresh role name $P_{=5}$ in $\Psi^R (A)$ and $I'$ is identical to $I$ otherwise. We have shown in item (ii) that $I' \models \Psi^R (A)$ and $I' \models T^A$. Since $I'$ coincides with $I$ on
all concepts and roles over the signature of $\mathcal{K}$ and $\mathcal{I} \models \mathcal{K}$, it follows that $\mathcal{I}' \models \mathcal{R}$, $\mathcal{I}' \models \mathcal{T}$ and $\mathcal{I}' \models \mathcal{R}^a$, and by items (i) and (iii), we can conclude $\mathcal{I}' \models \psi_R(\mathcal{T})$ and $\mathcal{I}' \models \psi_R(\mathcal{R}^a)$. Hence $\mathcal{I}' \models \psi_R(\mathcal{K})$.

For the second item of Proposition 6.7, take an arbitrary $\mathcal{I}$ and let $\mathcal{I}'$ be as in the claim, i.e., $R^\mathcal{I}' = (R^\mathcal{K}(R))^\mathcal{I}'$ for each role $R \in \mathcal{R}$ occurring in $\mathcal{K}$ and $\mathcal{I}'$ is identical to $\mathcal{I}$ otherwise. Observe that since $R^\mathcal{I}' = (R^\mathcal{K}(R))^\mathcal{I}'$ by construction, $\mathcal{I}' \models \mathcal{R}$ holds. Now we assume $\mathcal{I} \models \psi(\mathcal{K})$ and show that $\mathcal{I}' \models \mathcal{K}$. First, to show that $\mathcal{I}' \models \mathcal{T}$, we observe that since $\mathcal{I}' \models \mathcal{R}$, we have $R^\mathcal{I}' = (R^\mathcal{K}(R))^\mathcal{I}'$ for every role $R$ occurring in $\mathcal{K}$. From this and $R^\mathcal{K} = (R^\mathcal{K}(R))^\mathcal{K}$, it follows that $(R^\mathcal{K}(R))^\mathcal{I}' = (R^\mathcal{K}(R))^\mathcal{K}$. It is then easy to show that for all concepts $C$, $\psi_R(\mathcal{C})^\mathcal{I}' = \psi_R(\mathcal{C})^\mathcal{K}$. Hence $\mathcal{I}' \models \psi_R(\mathcal{T})$ implies $\mathcal{I}' \models \psi_R(\mathcal{K})$, and this together with $\mathcal{I}' \models \mathcal{R}$ and item (i) above implies $\mathcal{I}' \models \mathcal{T}$.

It is only left to show that $\mathcal{I}' \models \mathcal{A}$ and $\mathcal{I}' \models \mathcal{R}^a$. For the former, we again distinguish between the different forms of assertions in $\mathcal{A}$:

- For an assertion of the form $C(a) \in \mathcal{A}$, we again use the fact that $\psi_R(\mathcal{C})^\mathcal{I}' = \psi_R(\mathcal{C})^\mathcal{K}$, and the fact that $\mathcal{I}' \models \mathcal{R}$ implies $C^\mathcal{I}' = \psi_R(C)^\mathcal{I}'$ by item (i) above. It follows from $\mathcal{I} \models \psi_R(\mathcal{A})$ that $a^\mathcal{I} \in \psi_R(C)^\mathcal{I}'$, so $a^\mathcal{I} \in \psi_R(C)^\mathcal{K}$, and $a^\mathcal{I} \in C^\mathcal{I}'$. Hence $\mathcal{I}' \models C(a)$ as desired.

- For an assertion of the form $(R(a, b), a) \in \mathcal{A}$, we have that $R(a, b) \in \psi_R(\mathcal{A})$ and $\mathcal{I} \models \psi_R(\mathcal{A})$, so $\mathcal{I} \models R(a, b)$ and $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$, which implies $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}'$, hence $\mathcal{I}' \models R(a, b)$.

- For an assertion of the form $\lnot a \in \mathcal{A}$, we have that $a \in \psi_R(\mathcal{A})$ and $\mathcal{I} \models \psi_R(\mathcal{A})$, so $\mathcal{I} \models a \in \psi_R(\mathcal{A})$. Moreover, since $\mathcal{I} \models \mathcal{T}^\mathcal{R}$ and $(a^\mathcal{I}, b^\mathcal{I}) \in \mathcal{R}^\mathcal{I}$, it follows that $(a^\mathcal{I}, b^\mathcal{I}) \in \mathcal{R}^\mathcal{I}'$. Hence $\mathcal{I}' \models \lnot a$. This proves $\mathcal{I}' \models \mathcal{A}$. Finally, to show $\mathcal{I}' \models \mathcal{R}^a$, we consider the different forms of assertions about roles in $\mathcal{R}^a$:

- For $\mathrm{Sym}(R) \in \mathcal{R}^a$, we have $\mathrm{Inv}(\rho^\mathcal{K}(R)) \subseteq R \in \psi_R(\mathcal{R}^a)$. Assume $(x, y) \in R^\mathcal{I}'$. Then $(x, y) \in \rho^\mathcal{K}(R)^\mathcal{I}'$, so $(y, x) \in \rho^\mathcal{K}(R)^\mathcal{I}'$, and as $\mathcal{I}$ satisfies $\mathrm{Inv}(\rho^\mathcal{K}(R)) \subseteq R$, we have $(y, x) \in R^\mathcal{I}'$, and thus $(y, x) \in R^\mathcal{I}'$. This shows that $R^\mathcal{I}'$ is symmetric, so $\mathcal{I}'$ satisfies $\mathrm{Sym}(R)$.

- For $\mathrm{Ref}(R) \in \mathcal{R}^a$, we have $\mathcal{T} \models \exists R.\mathrm{Self} \in \psi_R(\mathcal{R}^a)$. Consider an arbitrary $x \in \Delta^\mathcal{I}$. Since $\mathcal{I}$ satisfies $\mathcal{T} \subseteq \exists R.\mathrm{Self}$, $x \in \exists R.\mathrm{Self}^\mathcal{I}$, hence $(x, x) \in R^\mathcal{I}$ and $(x, x) \in R^\mathcal{I}'$. This shows that $R^\mathcal{I}'$ is reflexive and $\mathcal{I}'$ satisfies $\mathrm{Ref}(R)$.

- For $\mathrm{Imp}(R) \in \mathcal{R}^a$, we have $\exists R.\mathrm{Self} \subseteq \mathcal{T} \subseteq \psi_R(\mathcal{R}^a)$. Consider an arbitrary $x \in \Delta^\mathcal{I}$, and assume towards a contradiction that $(x, x) \in R^\mathcal{I}'$. Then $(x, x) \in \rho^\mathcal{K}(R)^\mathcal{I}'$, which implies $x \in \exists R.\mathrm{Self}^\mathcal{I}$. But since $\mathcal{I} \models \psi_R(\mathcal{R}^a)$ and $\exists R.\mathrm{Self} \subseteq \mathcal{T} \subseteq \psi_R(\mathcal{R}^a)$, we have $\exists R.\mathrm{Self}^\mathcal{I} = \emptyset$, which contradicts $x \in \exists R.\mathrm{Self}^\mathcal{I}$. This shows that $R^\mathcal{I}'$ is irreflexive and $\mathcal{I}'$ satisfies $\mathrm{Imp}(R)$.

- For $\mathrm{Dis}(R, R') \in \mathcal{R}^a$, we have $\rho^\mathcal{K}(R) \cap \rho^\mathcal{K}(R') \subseteq \mathcal{B} \in \psi_R(\mathcal{R}^a)$. We consider an arbitrary pair $(x, y) \in R^\mathcal{I}'$. Then $(x, y) \in \rho^\mathcal{K}(R)^\mathcal{I}'$ and $\mathcal{I}$ satisfies $\rho^\mathcal{K}(R) \cap \rho^\mathcal{K}(R') \subseteq \mathcal{B}$, so $(x, y) \in R^\mathcal{I}'$. This shows that $R^\mathcal{I}'$ and $R^\mathcal{I}'$ are disjoint, and $\mathcal{I}'$ satisfies $\mathrm{Dis}(R, R') \in \mathcal{R}^a$.

This proves $\mathcal{I}' \models \mathcal{R}^a$. We have shown that $\mathcal{I}' \models \mathcal{T}$, $\mathcal{I}' \models \mathcal{R}$, and $\mathcal{I}' \models \mathcal{A}$, so $\mathcal{I}' \models \mathcal{K}$. This concludes the proof of the claim. □

As a consequence, we have a reduction from KB satisfiability in $\text{SRIQ}$ to KB satisfiability in $\text{ZIQ}$.

**Corollary 6.8.** For every $\text{SRIQ}$ KB $\mathcal{K}$, $\mathcal{K}$ is satisfiable iff $\psi(\mathcal{K})$ is satisfiable.

In the rewriting above, we have replaced $R$ by $\rho^\mathcal{K}(R)$ for all roles $R$. It would also be possible to do this replacement only for non-simple roles, while leaving simple roles untouched and keeping in the resulting $\text{ZIQ}$ TBox the corresponding inclusions (note that every SRIA with a simple role on the right hand side is syntactically a $\text{ZIQ}$ BRIA). We have also replaced $R$ by $\rho^\mathcal{K}(R)$ everywhere in $\mathcal{T}$ and $\mathcal{A}$. This is not strictly necessary, and an alternative translation from SRIQ to ZIQ can be defined by replacing $R$ by $\rho^\mathcal{K}(R)$ only in concepts of the forms $\exists R.\mathrm{C}$, $\exists R.\mathrm{Self}$, and $\exists n R.\mathrm{C}$ on the left hand side of inclusions, and in concepts of the forms $\forall R.\mathrm{C}$ and $\leq n R.\mathrm{C}$ on the right hand side of inclusions and in ABox assertions. Such a translation would resemble more the one in [61] and similar ones. However, it would require us to rewrite the KB into NNF, and to eliminate complex concepts $C$ that occur inside concepts of the forms $\exists R.\mathrm{C}$, $\forall R.\mathrm{C} \geq n R.\mathrm{C}$, or $\leq n R.\mathrm{C}$ (which can be easily done by introducing a fresh concept name $\mathrm{Ac}$ and adding CIAs $\mathrm{Ac} \subseteq C$, $C \subseteq \mathrm{Ac}$ to the TBox). In such
a translation the negated role assertions in the ABox and the assertions about roles in the RBox can be handled exactly as in Definition 6.6.

Example 6.9. Fig. 8 shows the result of applying the transformation $\Psi$ to the axioms in Fig. 7, using the regular expressions in Example 6.5. Note that the result is a $ZIQ$ TBox containing one CIA and one BRIA. □

6.2. Deciding KB satisfiability

By Corollary 6.8, the automata algorithm in Section 4 can be used to decide the satisfiability of $SRIQ$ knowledge bases; under unary number encoding, the resulting algorithm is worst-case optimal. Let $\rho_{\text{max}}^R$ be from the set $\{\rho_R^R | R \in R_R\}$ such that its length is maximal, i.e., $|\rho_{\text{max}}^R| = \max_{R \in R_R} |\rho_R^R|$. Each step of the rewriting $\Psi$ is clearly polynomial in the size of $A$, $T$, and $\rho_{\text{max}}^R$; however, the size of $\rho_{\text{max}}^R$ can be exponential in the size of $R$ [59].

Theorem 6.10. Under unary number encoding, the satisfiability of a given $SRIQ$ knowledge base $K = \langle T, R, A \rangle$ is decidable in time exponential in the combined size of $T$, $A$, and $\rho_{\text{max}}^R$ and in triple exponential time in the size of $K$.

As shown in [61], KB satisfiability for $SRIQ$ is 2ExpTime-hard. Hence our bound is optimal under the assumption of unary number encoding. Note that the blow-up in complexity w.r.t. $ZIQ$ is due to the size of $\rho_{\text{max}}^R$ and that the algorithm is single exponential whenever $\rho_{\text{max}}^R$ has size polynomial in $R$, e.g., for simple role hierarchies as defined in [59]. This compares well to the $SRIQ$ algorithm given in [5], which, even for such hierarchies may require non-deterministic double exponential time in the size of $K$.

6.3. Query answering in $SRIQ$

We also obtain an algorithm to decide query entailment in $SRIQ$. To this end, we rewrite a P2RPQ $q$ over $K$ into a new query over $\Psi(K)$ in a way such that query entailment is preserved.

Definition 6.11. For every P2RPQ $q$ over a $SRIQ$ knowledge base $K = \langle T, R, A \rangle$, $\Psi^R(q)$ is the P2RPQ obtained from $q$ by replacing each role $R$ with $\rho_R^R(R)$. □

Note that $\Psi^R(q)$ may contain regular expressions even when $q$ does not, i.e., our technique reduces positive (resp., conjunctive) queries over $SRIQ$ to positive (resp., conjunctive) regular path queries over $ZIQ$.

Lemma 6.12. Let $q$ be a P2RPQ over a $SRIQ$ knowledge base $K = \langle T, R, A \rangle$. Then $K \models q$ iff $\Psi(K) \models \Psi^R(q)$.

Proof. For the (Only If) direction, suppose $K \models q$ and consider an interpretation $I$ such that $I = \Psi(K)$. Let $I'$ be the variant of $I$ in which each $R$ is interpreted as $\langle \rho_R^R(R) \rangle^I$. Then, by item (2) of Proposition 6.7, $I' \models K$, hence $I' \models q$. As every match for $q$ in $I'$ is a match for $\Psi^R(q)$ in $I'$, it follows $I \models \Psi(q)$; hence, $\Psi(K) \models \Psi^R(q)$.

For the (If) direction, suppose $\Psi(K) \models \Psi^R(q)$. Consider an interpretation $I$ such that $I \models K$. By item (1) of Proposition 6.7, we know that $I' \models \Psi(K)$ for some $I'$ that only differs from $I$ in the interpretation of the fresh role names $P$-s that do not occur in $K$ or $q$, which means $I' \models \Psi^R(q)$. Hence, it follows $I \models \Psi^R(q)$. Since $I \models R$, Corollary 6.4 implies that $R^I = (\rho_R^R(R))^I$ for each $R \in R_R$, and since $I \models \Psi^R(q)$ it follows $I \models q$. This shows that $K \models q$ as desired. □

Again, the length of the longest regular expression $\rho_{\text{max}}^R$ over all role names $R$ in $R_R$ affects the overall complexity of the algorithm.

Theorem 6.13. Under unary number encoding, query entailment $K \models q$ for a given $SRIQ$ KB $K$ and a P2RPQ $q$ over $K$ is decidable in double exponential time in the combined size of $q$, $C_K$, $I_K$, and $\rho_{\text{max}}^R$, and in triple exponential time in the combined size of $q$ and $K$.

7. Conclusion

In this paper, we have substantially pushed the frontier of decidable query answering over expressive Description Logics (DLs), which is an active area of research driven by the growing interest in deploying DLs to various application areas. Exploiting automata-theoretic results and methods, we have shown that query entailment for a very rich class of queries beyond (union of) conjunctive queries, namely the positive (existential) two-way regular path queries (P2RPQs), is decidable over knowledge bases in the DL $ZIQ$. Making use of this result, we also show decidability of query entailment over knowledge bases in the DL $SRIQ$, which underlies the nominal-free fragment of the OWL 2 ontology standard by the W3C [3].
Our results also yield some novel complexity bounds, namely that the entailment problem of P2RPQs is 2Exp-Time-complete for \(ZIQ\) and in 3ExpTime for \(SRIQ\). Given that conjunctive query entailment is 2ExpTime-hard already for the DLs \(ALCT\) [46] and \(SH\) [47], and that CRPQ/PQ entailment is equally hard already for \(ALC\), our results show that both on the query and on the knowledge base side, one can increase the expressiveness substantially without a further increase in worst-case complexity. In particular, this applies to queries that allow one to navigate the models of a knowledge base in order to connect distant elements of the model, which is for instance desired in applications of semi-structured data and graph databases [32,30,19,27,62].

The automata-based technique we apply is, in a sense, more accessible than other techniques that are based on tableaux [59,63] or resolution-based transformations to disjunctive datalog [64]. It is computational in nature and works directly on models of a knowledge base, processing them with flexible local operations; furthermore, subtasks can be modularly combined. This allows us to accommodate different DL constructors and obtain results for more expressive knowledge bases and queries than had been considered before, which seems to be more difficult using other approaches.

The viability of the automata approach has been confirmed by [33], where along the lines and ideas of this paper, the decidability frontier for entailment of P2RPQs has been orthogonally extended to \(ZOQ\) and \(ZOI\) as well as to \(SROQ\) and \(SROI\). Furthermore, also decidability results for query containment are given there, which are obtained by a reduction to query answering, extending well-known relationships between query containment and conjunctive query answering to the richer setting of P2RPQs. However, the results in [33] use richer, tailored automata models which directly support features such as counting and forest shaped structures. This makes the encoding simpler, but at the same time leaves out essential technical aspects in handling, for instance, number restrictions and ABoxes. The encoding in this paper is from first principles using classical automata on infinite trees; it is more illustrative on the technical issues that have to be resolved.

Unlike [35,44] we have considered qualified number restrictions. To incorporate them into the automata algorithm as simply as possible, we have assumed that numbers are encoded in unary and have used a natural encoding of counters into the states of the automaton. It is still somewhat cumbersome, as the simultaneous presence of inverses, concepts \(\text{Self}\), and arbitrary ABoxes requires to navigate different parts of an interpretation to establish the satisfaction of a number restriction. We conjecture that one can obtain the same complexity bounds even if numbers are encoded in binary; however, this might require a significantly more involved encoding using binary counters.

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 the complexity of this problem. However, they seem to be of more limited use for assessing its data complexity, i.e., the complexity measured in terms of the ABox (data), assuming that the TBox and the query are fixed. Indeed, the set of states of the automaton \(A_q\) depends polynomially on the size of the ABox \(A\), hence the set of states of \(A_{K_q}\) can be double exponential in the size of \(A\) (more specifically, in the number of individuals occurring in \(A\)). This means that the algorithm only gives a double exponential bound for data complexity. It is important to note, however, that this is due to the way the ABox is incorporated into interpretation trees, which uses states that check the relations for each pair of ABox individuals. The automata-theoretic operations used to build \(A_{K_q}\) and the emptiness test on it treat all automata states equally, which does not allow us to distinguish the complexity arising from a specific kind of states. Although this is indeed an intrinsic limitation of our approach, there is no reason to believe that the query entailment problem itself has such a high data complexity. Indeed, we would not be surprised if query entailment in \(ZIQ\) and \(SRIQ\) is coNP-complete like for other expressive DLs [13,16]. However, other approaches may be more promising in order to establish such a result. The automata approach may still be viable, but requires a different handling of ABoxes, decoupling them from the infinite model tree, similarly as in [65].

Another drawback of the approaches based on automata on infinite trees is that they have so far resisted implementation. Although for simpler problems, such as TBox satisfiability some initial proposals for automata-based implementations have been made [66]. Hence, we do not expect the results presented here to lead to practicable algorithms in the near future. It now becomes interesting to look for alternative techniques that are better suited for implementation. We are confident that the tight complexity bounds that we have established will provide a valuable guidance in this direction, and may provide interesting insights to exploit, for instance, tableaux as in [16] or orbits as considered in [67].

Acknowledgments

This work has been partially supported by the FWF projects T515 Recursive Queries over Semantically Enriched Data Repositories and P20480 Reasoning in Hybrid Knowledge Bases, and the EU large-scale Integrating Project Optique Scalable End-user Access to Big Data, grant agreement No. FP7-318338.

We are very grateful to Oliver Carton, Orna Kupferman, and Moshe Vardi for their kind and valuable advise on automata-theoretic questions. We owe particular thanks to Yoad Lustig and Nir Piterman for providing a proof sketch connected to Lemma 2.15. Furthermore, we thank the reviewers for their helpful comments and suggestions to improve this paper.

Appendix A

In this appendix, we provide a proof of the canonical model property of \(ZIQ\) stated in Theorem 3.10, and of the correctness of the automata construction stated in Lemma 4.14.
A.1. Proof of the canonical model property

In what follows, we assume a fixed KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a P2RPQ $q$. Recall that $k_{\mathcal{T},q} = br(\mathcal{T} \cup D_q)$, where $D_q = \{ C | \mathcal{C}(v) \in At(q) \} \cup \{ \exists R.A | R(v, v) \in At(q) \}$ for an arbitrary concept name $A$, and $br(M) = |Cl(M)| \cdot n_{\text{max}}$ where $n_{\text{max}} = \max(|n| \geq nS.C \in Cl(M)) \cup \{0\}$. To simplify the notation, we use $\mathcal{C}(T, q)$ as a shorthand for $Cl(\mathcal{T} \cup D_q)$. To show that $\mathcal{K}$ has a $k_{\mathcal{T},q}$-canonical model, we will follow the lines of similar proofs for the $\mathcal{A}$-Box.

We start by defining concept and role atoms, which are consistent sets of concepts and roles from $\mathcal{C}(T, q)$. In what follows, we denote by $Cl_2(T, q)$ and $Cl_1(T, q)$ the set of concepts and the set of roles in $\mathcal{C}(T, q)$, respectively. A concept atom is a set $At \subseteq Cl_2(T, q)$ of concepts closed under the rules of Table 9, while a role atom of $\mathcal{K}$ is a set $AtR \subseteq Cl_1(T, q)$ of simple roles closed under the rules of Table 10. The set of all concept and the set of all role atoms are respectively denoted by $atC(T, q)$ and $atR(T, q)$.

A pre-model is an interpretation $\mathcal{I}$ in which each object is mapped to a concept atom and each pair of objects to a role atom. Formally, a pre-model of $\mathcal{K}$ is a pair $(\mathcal{I}, \theta)$ where $\mathcal{I} = (\Delta^2, \cdot)$ is an interpretation for $\mathcal{K}$ and $\theta$ is a function that maps each $d \in \Delta^2$ to a concept atom $\theta(d)eatC(T, q)$ and each pair $(d, d') \in \Delta^2 \times \Delta^2$ to a role atom $\theta(d, d') \in atR(T, q)$, such that

1. $C \theta = \theta(A^2)$, for each $a \in I^2$,
2. $A(a) \in A$, implies $A(\theta(a^2), \theta)$ and $p(a, b) \in A$ implies $p(\theta(a^2), \theta)$,
3. for each $d, d' \in \Delta^2$ and $p \in \mathcal{C}(T, q)$, $p(d, d') \in \theta(p(d, d'))$ and $\neg p(d, d') \in \theta(p(d, d'))$,
4. for each $d, d' \in \Delta^2$ and $p \in \mathcal{C}(T, q)$, $p(d, d')$ iff $p(d', d')$ and
5. for each $d \in \Delta^2$
   (a) $\theta(d)$ implies $d \in \Delta^2$ and $\neg \theta(d)$ implies $d \notin \Delta^2$, for each concept name $A \in Cl(T, q)$,
   (b) $\exists S.\theta(\phi)(d)$ implies $S \in \theta(d), \phi$,
   (c) if $\neg nS.C \in \theta(d)$, then there is some $V \subseteq \text{neigh}_{\mathcal{I},\theta}(S, d)$ such that $|V| \geq n$ and $C \theta' d'$ for every $d' \in V$, and
   (d) if $\neg nS.C \in \theta(d)$, then there is some $V \subseteq \text{neigh}_{\mathcal{I},\theta}(S, d)$ such that $|V| \leq n$ and $C \theta' d'$ for every $d' \in \text{neigh}_{\mathcal{I},\theta}(S, d) \setminus V$.

Intuitively, $(\mathcal{I}, \theta)$ is almost a model of $\mathcal{K}$, except that it is not ensured that concepts of the form $\exists R^*.C$ are satisfied. Instead, if $\exists R^*.C$ must hold at some element $d$, we only require that some $R$ neighbor of $d$ satisfies $\exists R^*.C$, and allow that the satisfaction of $C$ may be infinitely postponed. To trace the evaluation of $\exists R^*.C$ concepts and to distinguish pre-models that represent models of $\mathcal{K}$, we introduce adorned pre-models $(\mathcal{I}, \theta, eb)$ that extend pre-models $(\mathcal{I}, \theta)$ with a choice function.

| Table 9 |
| Concept atom $At \subseteq Cl_2(T, q)$. |
| If $A$ is a concept name in $Cl_2(T, q)$, then $A \in At$ and $\neg A \notin At$. |

| Table 10 |
| Role atom $AtR \subseteq Cl_1(T, q)$. |
| If $p$ is a role name in $Cl_1(T, q)$, then $p \in AtR$ and $\neg p \notin AtR$. |

| Role atom $AtR \subseteq Cl_1(T, q)$. |
| If $p$ is a role name in $Cl_1(T, q)$, then $p \in AtR$ and $\neg p \notin AtR$. |
A choice function for a pre-model \((\mathcal{I}, \theta)\) of \(\mathcal{K}\) is a partial function \(ch\) such that:

- for each pair \((d, C \cup C')\) with \(d \in \Delta^\mathcal{I}\) and \(C \cup C' \in \theta(d)\), \(ch(d, C \cup C')\) is a concept in \([C, C'] \cap \theta(d)\);
- for each pair \((d, \geq nS.C)\) with \(d \in \Delta^\mathcal{I}\) and \(\geq nS.C \in \theta(d)\), \(ch(d, \geq nS.C)\) is a subset \(V\) of \(\text{neighbor}_\mathcal{I,\theta}(S, d)\) such that \(|V| \geq n\) and \(C \in \theta(d')\) for every \(d' \in V\); and
- for each pair \((d, \leq nS.C)\) with \(d \in \Delta^\mathcal{I}\) and \(\leq nS.C \in \theta(d)\), \(ch(d, \leq nS.C)\) is a subset \(V\) of \(\text{neighbor}_\mathcal{I,\theta}(S, d)\) such that \(|V| \leq n\) and \(\sim C \in \theta(d')\) for every \(d' \in V\).

For an adorned pre-model \((\mathcal{I}, \theta, ch)\) of \(\mathcal{K}\), the derivation relation \(\vdash \subseteq (\Delta^\mathcal{I} \times \text{Cl}(T, q)) \times (\Delta^\mathcal{I} \times \text{Cl}(T, q))\) is the smallest relation such that for every \(d \in \Delta^\mathcal{I}\):

- \(C \cup C' \in \theta(d)\) implies \((d, C \cup C') \vdash (d, ch(d, C \cup C'))\),
- \(C \cap C' \in \theta(d)\) implies \((d, C \cap C') \vdash (d, C')\),
- \(\geq nS.C \in \theta(d)\) implies \((d, \geq nS.C) \vdash (d', C)\) for every \(d' \in ch(d, \geq nS.C)\),
- \(\leq nS.C \in \theta(d)\) implies \((d, \leq nS.C) \vdash (d', -C)\) for every \(d' \in \text{neighbor}_\mathcal{I,\theta}(S, d)\) \(\setminus ch(d, \leq nS.C)\),
- \(\exists R.C \in \theta(d)\) implies \((d, \exists R.C) \vdash (d, C \cup \exists R.\exists R.C)\), and
- \(\forall R.C \in \theta(d)\) implies \((d, \forall R.C) \vdash (d, C \cap \forall R.\forall R.C)\).

We say that a concept \(\exists R.C\) is regenerated from \(d\) to \(d'\) in \((\mathcal{I}, \theta, ch)\), if there is a sequence \((d_1, C_1), \ldots, (d_k, C_k)\) with \(k > 1\) such that \(d_1 = d, d_k = d', C_1 = C_k = \exists R.C, \exists R.C\) is a subconcept of every \(C_i\) and \((d_i, C_i) \vdash (d_{i+1}, C_{i+1})\) for each \(1 \leq i < k\). We also say that \((\mathcal{I}, \theta, ch)\) is well-founded, if there is no \(\exists R.C \in \text{Cl}(T, q)\) and infinite sequence \(d_1, d_2, \ldots\) such that \(\exists R.C\) is regenerated from \(d_i\) to \(d_{i+1}\) for every \(i \geq 1\). Then one can show:

**Lemma A.1.** For every normalized \(\mathcal{ZIQ} \text{ KB } \mathcal{K}\), the following holds.

1. If \(\mathcal{K} \not\models q\) for some \(q\), then \(\mathcal{K}\) has a well-founded adorned pre-model \((\mathcal{I}, \theta, ch)\) such that \(\mathcal{I} \not\models q\).
2. If \((\mathcal{I}, \theta, ch)\) is a well-founded adorned pre-model of \(\mathcal{K}\), then \(\mathcal{I}\) is a model of \(\mathcal{K}\).

**Proof (Sketch).** The proof is essentially an adaptation of similar proofs in [43,41], which extend the original proof for the \(\mu\)-calculus in [68]. The absence of alternating fixpoints makes our setting simpler, and the presence of the ABox and the additional constructs are not hard to accommodate.

1. If \(\mathcal{K} \not\models q\) for some \(q\), then there exists some \(\mathcal{I}\) such that \(\mathcal{I} \models \mathcal{K}\) and \(\mathcal{I} \not\models q\). The existence of a \(\theta\) such that \((\mathcal{I}, \theta)\) is a pre-model is straightforward: simply set \(\theta(d) = \{C \in \text{Cl}_{\mathcal{C}}(T, q) \mid d \in C^\mathcal{I}\}\) and \(\theta(d, d') = \{S \in \text{Cl}_{\mathcal{C}}(T, q) \mid (d, d') \in S^\mathcal{I}\}\) for every \(d, d' \in \Delta^\mathcal{I}\). The existence of a choice function \(ch\) that makes \((\mathcal{I}, \theta, ch)\) a well-founded adorned pre-model is also proved in the standard way. Roughly speaking, while a choice function trivially exists, to prove well-foundedness one observes that for every formula \(\exists R.C\) such that \(d \in \exists R.C\), there exists in \(\mathcal{I}\) some finite sequence of elements reachable from \(d\) via \(R\) that leads to some \(d' \in C^\mathcal{I}\); selecting such a path for the choice function avoids infinite regeneration of \(\exists R.C\).

2. One can show by structural induction that

\((\dagger)\) if \((\mathcal{I}, \theta, ch)\) is a well-founded adorned pre-model, then \(d \in C^\mathcal{I}\) for every \(C \in \theta(d)\) and \((d, d') \in S^\mathcal{I}\) for every \(S \in \theta(d, d')\).

We remark that Boolean role expressions are handled analogously to concept expressions, while items (4) and (5b) of a pre-model ensure the correct interpretation of inverse roles and Self concepts, respectively. For concepts of the form \(\exists R.C\), we rely on the well-foundedness of \((\mathcal{I}, \theta, ch)\), which ensures that \(\exists R.C\) is not infinitely regenerated and that \(\mathcal{C}\) is eventually satisfied. Once \((\dagger)\) has been shown, it is easy to see that \(\mathcal{I} \models \mathcal{K}\): item (1) of a pre-model ensures the satisfaction of the TBox, and item (2) of the ABox.

Now we are ready to prove **Theorem 3.10**, i.e., that for every normalized \(\mathcal{ZIQ} \text{ KB } \mathcal{K}\) and \(P2RPQ q\) such that \(\mathcal{K} \not\models q\), there exists a \(k_{\mathcal{Z}, \mathcal{Q}}\)-canonical model of \(\mathcal{K}\) that admits no match for \(q\).

**Proof of Theorem 3.10.** Assume \(\mathcal{K} \not\models q\). By item 1 of **Lemma A.1**, there exists some well-founded adorned pre-model \((\mathcal{I}, \theta, ch)\) of \(\mathcal{K}\) such that \(\mathcal{I} \not\models q\). We unravel \((\mathcal{I}, \theta, ch)\) into an adorned pre-model \((\mathcal{I}', \theta', ch')\) that is also well-founded, such that \(\mathcal{I}'\) is a \(k_{\mathcal{Z}, \mathcal{Q}}\)-canonical interpretation and \(\mathcal{I}' \not\models q\). By item 2 of **Lemma A.1**, it follows that \(\mathcal{I}'\) is a \(k_{\mathcal{Z}, \mathcal{Q}}\)-canonical model of \(\mathcal{K}\) with \(\mathcal{I}' \not\models q\), which proves the result.

We will inductively build the domain \(\Delta^\mathcal{I}'\) of \(\mathcal{I}'\) as a tree, and define a mapping \(\tau : \Delta^\mathcal{I}' \rightarrow \Delta^\mathcal{I}\) that keeps track of the correspondence between nodes of the tree and elements of \(\mathcal{I}\), which we use for defining the interpretation of concepts and roles in \(\mathcal{I}\). The functions \(\theta'\) and \(ch'\) are defined simultaneously.

Let \(R(\mathcal{I}) = \{a^\mathcal{I} \mid a \in I_\mathcal{C}\}\). Intuitively, to build the tree \(\Delta^\mathcal{I}'\), we start with the roots \(1, \ldots, |R(\mathcal{I})|\), and then continue building branches rooted at these roots, by inductively adding new levels.

For the base case, we let \(\Delta^\mathcal{I}' = \{1, \ldots, |R(\mathcal{I})|\}\) and let \(\tau : \{1, \ldots, |R(\mathcal{I})|\} \rightarrow R(\mathcal{I})\) be an arbitrary bijection. Then we set, for each \(j, j' \in \Delta^\mathcal{I}'\):
• \( \theta'(j) = \theta(\tau(j)) \),
• \( \theta'(j, j') = \theta(\tau(j), \tau(j')) \), and
• for each \( C \cup C' \in \theta'(j) \), \( \text{ch}'(j, C \cup C') = \text{ch}(\tau(j), C \cup C') \).

Choices for concepts \( \geq nS.C \) and \( \leq nS.C \) are defined in the induction step.

For the induction step, consider an \( x \in \Delta^x \) of maximal length, and let \( (\geq n_1 S_1, C_1, e_1), \ldots, (\geq n_m S_m, C_m, e_m) \) be all pairs of a formula \( \geq n_1 S_1, C_1 \in \theta'(x) \) and an \( e_i \in \text{ch}(\tau(x), \geq n_1 S_1, C_1) \). For each \( 1 \leq i \leq m \), we define:

\[
\phi(e_i) = \begin{cases} y, & \text{if } e_i = \tau(y) \text{ for } y \neq x \text{ or } y = x-1, \\ x, & \text{otherwise.} \end{cases}
\]

Then we set \( \Delta^x := \Delta^x \cup [\phi(e_1), \ldots, \phi(e_m)] \) and \( \tau(x) = e_i \), for each \( x \in \Delta^x \). To extend \( \theta' \), we set for each \( x \cdot i \in \Delta^x' \):

• \( \theta'(x \cdot i) = \theta(\tau(x \cdot i)) \),
• \( \theta'(y, x \cdot i) = \theta(\tau(y), \tau(x \cdot i)) \) if \( y = x \) or \( y = x \cdot i \),
• \( \theta'(x \cdot i, x) = \theta(\tau(x \cdot i), \tau(x)) \), and
• \( \theta'(y, x \cdot i) = \tau_i \), for every other \( y \), where \( \tau_i \) is a type such that \( \neg p \in \tau_0 \) for every \( p \in R \).

We also extend the choice function to concepts of the form \( C \cup C' \) in the new \( \theta'(x \cdot i) \), and to concepts \( \geq nS.C \) and \( \leq nS.C \) in \( \theta'(x) \):

• for each \( x \cdot i \in \Delta^x \) and each \( C \cup C' \in \theta'(x \cdot i) \), \( \text{ch}'(x \cdot i, C \cup C') = \text{ch}(\tau(x \cdot i), C \cup C') \);
• for each \( \geq nS.C \in \theta'(x) \), \( \text{ch}'(x, \geq nS.C) = [\phi(e) \mid e \in \text{ch}(\tau(x), \geq nS.C)] \); and
• for each \( \leq nS.C \in \theta'(x) \), \( \text{ch}'(x, \leq nS.C) = [\phi(e) \mid e \in \text{ch}(\tau(x), \leq nS.C) \cap \{e_1, \ldots, e_n\}] \).

Finally, the interpretation \( \mathcal{I}' \) is defined using the mapping \( \tau' \):

• for each \( A \in \mathcal{C}_K, A^{\mathcal{I}'} = \{x \in \Delta^x \mid \tau(x) \in A^x\} \), and
• for each \( p \in \mathcal{R}_K, p^{\mathcal{I}'} = \{(x, y) \in \Delta^x \times \Delta^y \mid (\tau(x), \tau(y)) \in p^x\} \).

We now verify that the conditions (1)-(5) of Definition 3.7 hold. Clearly, (1) \( \{e\} \cup \Delta^x \) is a tree, and (2) \( \text{Roots}(\mathcal{I}) = \{a^x \mid a \in I_K\} \subseteq N \). Next, \( m \) is bounded by \( k_{\tau, q} \) for every sequence \( \phi(e_1), \ldots, \phi(e_m) \) above. Hence by the induction hypothesis, each \( y \in \Delta^x \) is of the form \( x \cdot i \) with \( i \in \text{Roots}(\mathcal{I}') \) and \( x \in \{1, \ldots, k_{\tau, q}\}^* \), and (3) holds. Also, for each \( x \in \Delta^x \), each new element added to \( \Delta^x \) as a child of \( x \) is of the form \( x \cdot i = \phi(e_i) \) with \( e_i \in \text{ch}(\tau(x), \geq n_1 S_1, C_1) \subseteq \text{neigh}_{\mathcal{I}, \theta}(S_1, x) \), hence \( (x, x \cdot i) \in (S_1)^{\mathcal{I}'} \). Since \( S_1 \) is safe, this implies the existence of some atomic role \( P \) such that \( (x, x \cdot i) \in P^{\mathcal{I}'} \), as required by (4). For (5), it only remains to observe that for every pair \( (x, y) \) such that neither (a) \( x = y \), nor (b) \( y \) is a successor of \( x \), nor (c) \( x \) is the predecessor of \( y \), the construction above ensures that \( \theta'(x, y) = \tau_0 \) is such that \( \neg p \in \theta'(x, y) \) for every \( p \in R \) and hence \( (x, y) \notin p^{\mathcal{I}'} \). Therefore, \( \mathcal{I}' \) is a \( k_{\tau, q} \)-canonical interpretation. Furthermore, as \( (\mathcal{I}, \theta, \text{ch}) \) is an adorned pre-model of \( K \), it is readily checked that \( (\mathcal{I}', \theta', \text{ch}') \) is also an adorned pre-model of \( K \). Finally, if a concept \( \exists R^*C \) is regenerated from \( x \) to \( y \) in \( (\mathcal{I}', \theta', \text{ch}') \), then \( \exists R^*C \) is also regenerated from \( \tau(x) \) to \( \tau(y) \) in \( (\mathcal{I}, \theta, \text{ch}) \). As a consequence, well-foundedness of \( (\mathcal{I}, \theta, \text{ch}) \) implies well-foundedness of \( (\mathcal{I}', \theta', \text{ch}') \).

Finally, we show that \( \mathcal{I}' \nmid q \). Assume towards a contradiction that \( \mathcal{I}', \pi \models q \) for some \( \pi \). Then the following claim implies that \( \pi = \pi \circ \tau \) is a match for \( q \) in \( \mathcal{I} \), contradicting the assumption that \( \mathcal{I} \nmid q \).

Claim. Let \( R(t, t') \) be an atom in \( q \) and let \( x, y \) be nodes in \( \Delta^x \). If \( (x, y) \in R^x \) then \( (\tau(x), \tau(y)) \in R^x \). Similarly, for an atom \( C(t) \) of \( q \) and \( x \in \Delta^x \), \( x \in C^x \) implies \( \tau(x) \in C^x \).

It is only left to prove the claim. To this end we exploit the following two properties, which state that all concepts and roles in the query are preserved during the construction of \( \mathcal{I}' \):

• Let \( (x, y) \in \Delta^x \times \Delta^y \) and let \( S \) be a simple role in \( \text{Cl}(T, q) \). Then \( (x, y) \in S^{\mathcal{I}} \) implies \( (\tau(x), \tau(y)) \in S^x \).
• Let \( x \in \Delta^x \) and let \( C \) be a concept in \( \text{Cl}(T, q) \). Then \( x \in C^{\mathcal{I}} \) implies \( \tau(x) \in C^x \).

The proof of both statements is easy, since the construction of \( \mathcal{I}' \) ensures that each pair of objects \( x, y \) (resp., each object \( x \)) preserves all roles (resp., concepts) in the closure that are satisfied by \( \tau(x), \tau(y) \) (resp., \( \tau(x) \)). Formally, they can be shown by a straightforward induction on the structure of \( S \) and \( C \), respectively. To prove the claim, it then suffices to observe that by construction of \( \mathcal{I}' \), if a sequence of objects in \( \mathcal{I}' \) may participate in the satisfaction of some atom in \( q \), the image of this sequence under \( \tau \) would satisfy the same atom in \( \mathcal{I} \).
A.2. Correctness of the main automata construction

Proof of Lemma 4.14. We first show similar properties for simple roles and for the states in $Q_{\text{Self}}$ and $Q_{\text{A\_role}}$. In the following claims, $T = (T, L)$ is an interpretation tree for $K$, as above.

(C1) Let $x \cdot i \in \Delta_{iT}$ and let $S$ be a simple role in $\text{Cl}(C_T)$. Then there is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$ iff $(x, x \cdot i) \in S_{iT}$.

(C2) Let $x \in \Delta_{iT}$ and let $S_{\text{Self}} \in Q_{\text{Self}}$. Then there is an accepting $(x, S)$-run of $A_T$ over $T$ iff $(x, x) \in S_{iT}$.

(C3) Let $i, j \in \Delta_{iT}$ and $S_{ij} \in Q_{\text{A\_role}}$. Then there is an accepting $(i, j, S_{ij})$-run of $A_T$ over $T$ iff $(i, j) \in S_{ij}$.

Each of these properties is shown by a straightforward structural induction on the role $S$.

(C1) Induction base: Let $S = P \in \vec{R}_K$. (If) Assume $(x, x \cdot i) \in P_{iT}$. By construction of $T_P$, $(x, x \cdot i) \in P_{iT}$ implies that either $P = p \in R$ and $(x, x \cdot i) \in R_{i}^p$ or $P = p^{-}$ for some $p \in R$ and $(x \cdot i, x) \in R_{i}^p$. In the former case, $P = L(x, i), b$ is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$. (Only if) Assume that there is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$. Then the root is labeled $r(\varepsilon) = (x, i, P).$ By construction of $A_T$, $\delta_T(P, L(x, i)) = \text{true}$ and the run with one single node $\varepsilon$ labeled $r(\varepsilon) = (x, i, P)$ is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$.

(Only if) Assume that there is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$. Then the root is labeled $r(\varepsilon) = (x, i, P)$. By construction of $A_T$, $\delta_T(P, L(x, i)) = \text{true}$ and the run with one single node $\varepsilon$ labeled $r(\varepsilon) = (x, i, P)$ is an accepting $(x \cdot i, S)$-run of $A_T$ over $T$. Hence, since condition (R2) in Definition 2.8 holds, it follows that $P = L(x, i).$ If $P = p \in R$, this implies $(x, x \cdot i) \in R_{i}^p$, and if $P = p^{-}$ for some $p \in R$, this implies $(x \cdot i, x) \in R_{i}^p$. Since $x \cdot i \in \Delta_{iT}$, in both cases this implies $(x, x \cdot i) \in P_{iT}$.

(C1) Inductive step:

• First we consider the case of role difference. Recall that all roles in $\text{Cl}(C_T)$ are in $\text{NNF}$, hence we can assume that $S = S' \cap P$ for some atomic role $P$.

(If) Assume $(x, x \cdot i) \in S_{iT}$. Then $(x, x \cdot i) \in S'_{iT}$, and $(x, x \cdot i) \notin P_{iT}$. By definition of $T_P$, $(x, x \cdot i) \notin P_{iT}$ implies $P \notin L(x, i)$. By the inductive hypothesis, $(x, x \cdot i) \in S'_{iT}$ implies that there is an accepting $(x \cdot i, S')$-run $(T_r, r)$ of $A_T$ over $T$. To extend it into an accepting $(x \cdot i, S)$-run, we first introduce the following notation: for $n \in N$, let $n \cdot T_r = \{n \cdot w \mid w \in T_r \}$ (note that $n \cdot T_r$ is a practically a tree, but its root is $n$ rather than $\varepsilon$). Then we build a new run $(T_r', r')$ by taking $T_r' = \{n \cdot i, \cup (2 \cdot r), T_r \}$, and setting $r'((\varepsilon)) = (x \cdot i, S')$, $r'((\varepsilon)) = (x \cdot i, -P)$, and $r'((w)) = r(w)$ for every $w \in T_r$. Condition (R2) in Definition 2.8 holds for all $w - w \in T_r$ by construction. It also holds for $1 \in T_r$, since $P \notin L(x, i)$, as mentioned above, and hence $\delta_T(P, L(x, i)) = \text{true}$ by item (III) in the definition of $\delta_T$. Finally, condition (R2) also holds for $S$ as $\delta_T(S, \sigma) = (0, S') \cup (0, -P')$ (by item (II)). This shows that $(T_r', r')$ is an accepting $(x \cdot i, S)$-run.

(Only if) Assume that there is an accepting $(x \cdot i, S)$-run $(T_r, r)$ of $A_T$ over $T$. Then we have $r(\varepsilon) = (x, i, S)$, and the only transition that can be used to satisfy condition (R2) is $\delta_T(S, \sigma) = (0, S') \cup (0, -P')$. This means that $\sigma$ must have children $j, \ell$ such that $r(j) = (x \cdot i, 0, S')$ and $r(\ell) = (x \cdot i, -P)$. It is not hard to see that if we restrict $(T_r, r)$ to the subtrees rooted at $j$ and $\ell$, we obtain an accepting $(x \cdot i, S)$-run and an accepting $(x \cdot i, -P)$-run of $A_T$ on $T$. By the induction hypothesis, this implies $(x, x \cdot i) \in S'_{iT}$, and $(x, x \cdot i) \notin P_{iT}$, hence $(x, x \cdot i) \in S_{iT}$ as desired.

• The cases of $S = S_1 \cap S_2$ and $S_1 \cup S_2$ are analogous. We only sketch the former.

(If) Assume $(x, x \cdot i) \in S_{iT}$. Then $(x, x \cdot i) \in S'_{iT}$, and $(x, x \cdot i) \in S''_{iT}$. Then by the inductive hypothesis, $(x, x \cdot i) \in S'_{iT}$ implies that there is an accepting $(x \cdot i, S_1)$-run $(T_r, r_1)$ and an accepting $(x \cdot i, S_2)$-run $(T_r, r_2)$ of $A_T$ over $T$. We can then build an accepting $(x \cdot i, S)$-run $(T_r', r')$ by taking $T_r' = \{\varepsilon \cup (1 \cdot T_r), \cup (2 \cdot T_r) \}$, and setting $r'((\varepsilon)) = (x \cdot i, S)$, $r'((j)) = r((w))$ for every $w \in T_r$, $j \in \{1, 2\}$.

(Only if) If there exists an accepting $(x \cdot i, S)$-run $(T_r, r)$ of $A_T$ on $T$, then $\delta_T(S, \sigma) = (0, S_1) \cup (0, S_2) \cup (0, S_3)$ must be satisfied, which means that $\sigma$ must have children $j, \ell$ such that $r(j) = (x \cdot i, 0, S_1)$ and $r(\ell) = (x \cdot i, 0, S_2)$. It is not hard to see that if we restrict $(T_r, r)$ to the subtrees rooted at $j$ and $\ell$, we obtain an accepting $(x \cdot i, S_1)$-run and an accepting $(x \cdot i, S_2)$-run of $A_T$ on $T$. By the induction hypothesis, this implies $(x, x \cdot i) \in S'_{iT}$ and $(x, x \cdot i) \notin P_{iT}$, hence $(x, x \cdot i) \in S_{iT}$ as desired.

(C2) Induction base: First we assume a role name $p \in R$. If $I_C \cap L(x) \neq \emptyset$ then $x$ is an individual node. Otherwise, if $x \in \Delta_{iT}$ and $I_C \cap L(x) = \emptyset$, then it is not a level one node. Accordingly, we distinguish two cases:

• $x$ is not a level one node. Then it suffices to observe that $p_{\text{Self}} \in L(x) \iff (x, x) \in P_{iT}$. Then it can be shown as above that the existence of an accepting $(x, p_{\text{Self}})$-run implies $p_{\text{Self}} \in L(x)$ and conversely, if $p_{\text{Self}} \in L(x)$, then there is an accepting $(x, p_{\text{Self}})$-run (comprising a single node $\varepsilon$ with $r(\varepsilon) = (x, p_{\text{Self}})$).

• $x \in I_C$ is an individual node, i.e., there is some $a \in I_C$ such that $a \in L(x)$. Note that $a_{iT} = x$, and $(x, x) \in P_{iT}$ if $p_{\text{Self}} \in L(x)$. Hence, if $(x, x) \in P_{iT}$, then we can build an accepting $(x, p_{\text{Self}})$-run by taking a root $\varepsilon$ with $r(\varepsilon) = (x, p_{\text{Self}})$, adding to it children 1 and 2 with $r(1) = (x, a)$ and $r(2) = (e, (a, p_{\text{Self}}))$, and then adding nodes 2-1 and 2-2, children of 2, which respectively have $r(2-1) = (x, a)$ and $r(2-2) = (e, p_{\text{Self}})$. Since $a \in L(x)$ and $p_{\text{Self}} \in L(e)$, the transition function is satisfied. Conversely, if there is an accepting $(x, p_{\text{Self}})$-run, then its root must have children $i$ and $j$ with $r(i) = (x, b)$ and $r(j) = (e, (b, p_{\text{Self}}))$ for some $b \in I_C$. Since the run is accepting, this $b$ must be such that $b \in L(x)$. The node $j$ must have children with $r(j' - i') = (y, b)$ and $r(j' - j') = (e, p_{\text{Self}})$. Again, since the run is accepting, we can conclude that $b \in L(y)$, which implies $y = x$ (since $T$ is an interpretation tree, there is be exactly one node with $b$ in its label). Since
the accepting run has a node labeled \((\varepsilon, p_{xx})\), it follows that \(p_{xx} \in L(\varepsilon)\), which implies \((x, x) \in p_{xx}^{I_T}\) by construction of \(I_T\).

For \(S\) of the form \(p^{-}\) with \(p \in \mathcal{R}\), we have that in an accepting \((x, p^{-}_{\text{Self}})\)-run the root has a child \(j\) such that \(r(j) = (x, p_{xj})\), which implies \((x, x) \in p_{xj}^{I_T}\) and hence \((x, x) \in S^{I_T}\) and \(x \in \exists S_1\text{Self}\). This shows the \((I')\) direction. For the \((\text{Only if})\) direction, \(x \in \exists S_1\text{Self}\) implies \((x, x) \in p_{xx}^{I_T}\) and, as shown above, there is an accepting \((x, p_{xx}^{I_T})\)-run, which can be easily extended to an \((x, p_{\text{Self}}^{-})\)-run by adding to it a root labeled \((x, p_{\text{Self}}^{-})\) (note that \(\delta_T(p_{\text{Self}}^{-}, L(x)) = (0, p_{\text{Self}})\) by definition).

(C2) **Inductive step:** This step is practically identical to the one of (C1). In the case of role difference one needs to make a case distinction between a role name \(p\) and its inverse \(p^{-}\), analogously to the induction base.

(C3) **Induction base:** For a role name \(p \in \mathcal{R}\), we observe that, since \(i, j \in \Delta^T\) and by the construction of \(I_T\), \(p_{ij} \in L(\varepsilon)\) iff \((i, j) \in p^{I_T}\). Then the rest is as above: the existence of an accepting \((x, p_{ij})\)-run implies \(p_{ij} \in L(\varepsilon)\) and conversely, if \(p_{ij} \in L(\varepsilon)\), then there is an accepting \((x, p_{ij})\)-run (comprising a single node \(\varepsilon\) with \(r(\varepsilon) = (x, p_{ij})\)). For \(S\) of the form \(p^{-}\) with \(p \in \mathcal{R}\): \((I')\) In an accepting \((x, p_{ij}^\varepsilon)\)-run the root has a child \(k\) such that \(r(k) = (x, p_{jk})\), which implies \(p_{jk} \in L(\varepsilon)\) and hence \((j, i) \in p^{I_T}\) and \((i, j) \in p^{-I_T}\). \((\text{Only if})\) Since \((i, j) \in p^{-I_T}\) implies \((j, i) \in p^{I_T}\), we know we have \(p_{ij} \in L(\varepsilon)\) and, as shown above, there is an accepting \((x, p_{ij})\)-run, which can be easily extended to an \((x, p_{ij}^\varepsilon)\)-run by adding to it a root labeled \((x, p_{ij})\) (note that \(\delta_T(p_{ij}^\varepsilon, L(x)) = (0, p_{ij})\) by definition).

(C3) **Inductive step:** This is again analogous to (C1) and (C2).

Using these claims, one can show the lemma by structural induction on \(C\).

**Induction base:** Let \(C = A \in \mathcal{C}\). By the transitions in item III of Table 4 (which are the only ones given for states \(A \in \mathcal{C}\)) there is an accepting \((A, A)\)-run iff \(A \in L(a)\), and by construction of \(I_T\) we have \(A \in L(a)\) iff \(A \in L(x)\).

**Inductive step:** We must distinguish several cases due to the large number of constructors of \(\mathcal{ZTQ}\).

- \(C\) is of the form \(\neg A\), for a concept name \(A\) (recall that all concepts in \(\mathcal{C}(C_T)\) are in NNF). The proof is analogous to the base case: we have \(A \notin L(x)\) iff \(x \in \neg A^{I_T}\), and by the transitions in item III of Table 4 there is an accepting \((x, \neg A)\)-run iff \(A \notin L(x)\).
- \(C\) is of the form \(D \cap D'\) or \(D \cup D'\). The proof is analogous as for intersection and union of simple roles.
- \(C\) is of the form \(\leq n_\mathcal{S}\) or of the form \(\geq n_\mathcal{S}\).

We rely on the following auxiliary claims. Here, we use the notion of potential neighbors and \((S, D)\)-neighbors defined above. Note that, by definition, the potential neighbors include all successors of a node, also the ‘dummy’ nodes in \(T\) that do not correspond to an object in \(\Delta^T\). For an individual node, its potential neighbors also include all the level 1 nodes, both actual individual nodes and ‘dummy’ nodes. Note that, in contrast, the \((S, D)\)-neighbors are necessarily elements of \(\Delta^T\). Moreover, we order the potential neighbors of a node \(x\) as follows: the first \(b_x\) ones are its successors, in the order they occur in \(T\). If there is no \(a \in I_k\) such that \(a^{T_x} = x\), then the \((b_x + 1)\)-th is the node itself, and the \((b_x + 2)\)-th is its predecessor. Otherwise, the \((b_x + 1)\)-th to \((2b_x)\)-th are the level 1 nodes, in the order they appear in \(T\).

(C4) Let \(x \in \Delta^T\) and let \(\langle \leq n_\mathcal{S}, D, i, j \rangle \in \mathcal{Q}_\text{Num} \) (resp., \(\langle \leq n_\mathcal{S}, D, i, j \rangle \in \mathcal{Q}_\text{Num} \)). Then there is an accepting \((x, \langle \leq n_\mathcal{S}, D, i, j \rangle)\)-run (resp., \((x, \langle \leq n_\mathcal{S}, D, i, j \rangle)\)-run) of \(A_T\) over \(T\) iff there are at least (resp., at most) \(n - j\) many \((S, D)\)-neighbors of \(x\) among its potential neighbors beyond the \(i\)-th one.

(C5) Let \(\langle a, \geq n_\mathcal{S}, D, i, j \rangle \in \mathcal{Q}_\text{Num} \) (resp., \(\langle a, \geq n_\mathcal{S}, D, i, j \rangle \in \mathcal{Q}_\text{Num} \)). Then there is an accepting \((x, \langle a, \geq n_\mathcal{S}, D, i, j \rangle)\)-run of \(A_T\) over \(T\) iff there are at least (resp., at most) \(n - j\) many \((S, D)\)-neighbors of \(a^{T_x}\) among its potential neighbors beyond the \((b_x + i)\)-th.

Both claims are easy to show, since the transitions in item IV2 of Table 5 ensure that all potential neighbors are navigated in the right order. The \(i\) counter is always increased, while the \(j\) counter is increased iff the \(i\)-th potential neighbor is actually an \((S, D)\)-neighbor. The correctness of the latter check follows easily from the way both disjuncts are defined, relying on the Claims (C1)-(C3) for \(S\) and on the inductive hypothesis for \(D\).

For each number restriction \(\geq n_\mathcal{S},\), the automaton will first switch to \(\langle \geq n_\mathcal{S}, C, 0, 0 \rangle\). From there, by Claims (C4) and (C5), it will correctly count its potential neighbors. An inspection of the transitions in item IV3 of Table 5 reveals that the run can be successfully completed iff the number restriction holds at the corresponding node.

- \(C\) is of the form \(\exists S_1\text{Self}\). Recall that \(x \in \exists S_1\text{Self}^{I_T}\) iff \((x, x) \in S^{I_T}\), and that \(\delta_T(\exists S_1\text{Self}, \sigma) = (0, S_1\text{Self})\) is the only transition specified for states of the form \(\exists S_1\text{Self}\). We can apply Claim (C2). For the \((\text{If})\) direction, we have that \(x \in \exists S_1\text{Self}^{I_T}\) implies the existence of an accepting \((x, S_1\text{Self})\)-run, which can be extended into an \((x, \exists S_1\text{Self})\)-run by simply adding a root labeled \((x, \exists S_1\text{Self})\). For the converse, in an accepting \((x, \exists S_1\text{Self})\)-run the root necessarily has a child labeled \((x, S_1\text{Self})\), which can be viewed as the root of an accepting \((x, S_1\text{Self})\)-run.

- \(C\) is of the form \(\exists S, D\) or \(\forall S, D\) for a simple role \(S\). Then the claim follows from the fact that the transition function treats these concepts as equivalent number restrictions.

\((I')\) \(x \in (\exists S, D)^{I_T}\) then \(x \in (\geq 15, D)^{I_T}\) so we can take an accepting \((x, \geq 15, D)\)-run, which we have shown to exist and extend it into an \((x, \exists S, D)\)-run by adding a root with label \((x, \exists S, D)\). Similarly, adding an \((x, \forall S, D)\)-labeled root to an accepting \((x, \leq 5, \neg D)\)-run yields an accepting \((x, \forall S, D)\)-run. \((\text{Only if})\) Conversely, if we remove the root of an accepting \((x, \exists S, D)\)-run we obtain a subtree that is an accepting \((x, \geq 15, D)\)-run. As we have shown, this implies that
References


