ExpTime Tableaux with Global Caching for the Description Logic SHOQ

Linh Anh Nguyen\textsuperscript{1,2} and Joanna Golińska-Pilarek\textsuperscript{3}

\textsuperscript{1} Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

\textsuperscript{2} Faculty of Information Technology, VNU University of Engineering and Technology
144 Xuan Thuy, Hanoi, Vietnam

\textsuperscript{3} Institute of Philosophy, University of Warsaw
Krakowskie Przedmieście 3, 00-927 Warsaw, Poland
j.golinska@uw.edu.pl

July, 2013 (last revised: July, 2014)

Abstract. We give the first ExpTime (complexity-optimal) tableau decision procedure for checking satisfiability of a knowledge base in the description logic SHOQ, which extends the basic description logic ALC with transitive roles, hierarchies of roles, nominals and quantified number restrictions. The complexity is measured using unary representation for numbers. Our procedure is based on global caching and integer linear feasibility checking.

Keywords: automated reasoning, description logics, global state caching, integer linear feasibility

1 Introduction

Description logics (DLs) are formal languages suitable for representing terminological knowledge. They are of particular importance in providing a logical formalism for ontologies and the Semantic Web. DLs represent the domain of interest in terms of concepts, individuals, and roles. A concept is interpreted as a set of individuals, while a role is interpreted as a binary relation among individuals. A knowledge base in a DL consists of axioms about roles (grouped into an RBox), terminology axioms (grouped into a TBox), and assertions about individuals (grouped into an ABox). A DL is usually specified by: i) a set of constructors that allow building complex concepts and complex roles from concept names, role names and individual names, ii) allowed forms of role axioms. The basic DL ALC allows basic concept constructors listed in Table 1 but does not allow role constructors nor role axioms. The most common additional features for extending ALC are also listed in Table 1 together with syntax and examples: $I$ is a role constructor, $Q$ and $O$ are concept constructors, while $H$ and $S$ are allowed forms of role axioms. The name of a DL is usually formed by the names of its additional features, as in the cases of $SH$, $SHI$, $SHIQ$, $SHIO$ and $SHOQ$. $SROIQ$ \cite{12} is a further expressive DL used as the logical base for the Web Ontology Language OWL 2 DL.

Automated reasoning in DLs is useful, for example, in engineering and querying ontologies. One of basic reasoning problems in DLs is to check satisfiability of a knowledge base in a considered DL. Most of other reasoning problems in DLs are reducible to this one. In this paper, we study the problem of checking satisfiability of a knowledge base in the DL SHOQ, which extends the basic DL ALC with transitive roles ($S$), hierarchies of roles ($H$), nominals ($O$) and quantified number restrictions ($Q$). It is known that this problem in SHOQ is ExpTime-complete \cite{30} (even when numbers are coded in binary). Nominals, interpreted as singleton sets, are a useful notion to express identity and uniqueness. However, when interacting with inverse roles ($I$) and quantified number restrictions in the DL SHOIQ, they cause the complexity of the above mentioned problem to jump up to NExpTime-complete \cite{29} (while that problem in any of the DLs SHOQ, SHIO, SHIQ is ExpTime-complete \cite{30,11,29}).

In \cite{13} Horrocks and Sattler gave a tableau algorithm for deciding the DL SHOQ(D), which is the extension of SHOQ with concrete datatypes. Later, Pan and Horrocks \cite{26} extended
Concept constructors of ALC

| Constructor       | Syntax | Example     |
|-------------------|--------|-------------|
| complement       | ¬C     | ¬Male       |
| intersection      | C ∩ D  | Human ∩ Male |
| union             | C ⊔ D  | Doctor ⊔ Lawyer |
| existential restriction | ∀r.C | ∀hasChild.Male |
| universal restriction | ∃r.C | ∃hasChild.Female |

**Table 1.** Concept constructors for ALC and some additional constructors/features of other DLs.

| Constructor/Feature | Syntax | Example     |
|---------------------|--------|-------------|
| inverse roles (I)   | r⁻     | hasChild⁻ (i.e., hasParent) |
| quantified number restrictions (Q) | ≥n R.C | ≥3 hasChild.Male |
| ≤n R.C | ≤2 hasParent.⊤ |
| nominals (O)        | {a}    | {John}      |
| hierarchies of roles (H) | R ⊑ S | hasChild ⊑ hasDescendant |
| transitive roles (S) | R ◦ R ⊑ R | hasDescendant ◦ hasDescendant ⊑ hasDescendant |

the method of [13] to give a tableau algorithm for deciding the DL $SHOQ(D_n)$, which is the extension of $SHOQ$ with $n$-ary datatype predicates. These algorithms use backtracking to deal with disjunction ($∪$) and “or”-branching (e.g., the “choice”-rule) and use a straightforward way for dealing with quantified number restrictions. They have a non-optimal complexity ($N^2\text{ExpTime}$) when numbers are coded in unary.

In [2] Faddoul and Haarslev gave an algebraic tableau reasoning algorithm for $SHOQ$, which combines the tableau method with linear integer programming. The aim was to increase efficiency of handling quantified number restrictions. However, their algorithm still uses backtracking to deal with disjunction and “or”-branching and has a non-optimal complexity (“double exponential” [2]).

This paper is a revised and extended version of our workshop paper [20]. In this work we present the first tableau method with an $\text{ExpTime}$ (optimal) complexity for checking satisfiability of a knowledge base in the DL $SHOQ$ when numbers are coded in unary.

Our method is based on global caching and integer linear feasibility checking.

The idea of global caching comes from Pratt’s work [27] on PDL. It was formally formulated for tableaux in some DLs in [6,8] and has been applied to several modal and description logics [5,7,21,22,23,24,25,1] to obtain tableau decision procedures with an optimal complexity. A variant of global caching, called global state caching, was used to obtain cut-free optimal tableau decision procedures for several modal logics with converse and DLs with inverse roles [9,10,16,17,19].

Integer linear programming was exploited for tableaux in [3,2] to increase efficiency of reasoning with quantified number restrictions. However, the first work that applied integer linear feasibility checking to tableaux was [18,19]. In [18] Nguyen gave the first $\text{ExpTime}$ (optimal) tableau decision procedure for checking satisfiability of a knowledge base in the DL $SHIQ$ when numbers are coded in unary. His procedure is based on global state caching and integer linear feasibility checking. In the current paper, we apply his method of integer linear feasibility checking to $SHOQ$. The adaptation requires special techniques due to the following reasons: i) we use global caching for $SHOQ$, while Nguyen’s work [13] uses global state caching for $SHIQ$ (for dealing with inverse roles); ii) we have to deal with the interaction between number restrictions and nominals. Our method substantially differs from Farsiniamarj’s method of exploiting integer programming for tableaux [3]. Our technique for dealing with both nominals and quantified number restrictions is also essentially different from the one by Faddoul and Haarslev [2].

---

4 When the algorithms are improved by using “anywhere blocking”, the complexity will be $N\text{ExpTime}$ and still non-optimal.

5 This corrects the claim of [20] that the complexity is measured using binary representation for numbers.
The rest of this paper is structured as follows. In Section 2 we recall notation and semantics of SHOQ as well as the integer feasibility problem for DLs. In Section 3 we present our tableau decision procedure for SHOQ together with examples for illustrating our tableau method. We conclude this work in Section 5. Proofs for our results are given in the Appendix.

2 Preliminaries

2.1 Notation and Semantics of SHOQ

Our language uses a finite set $C$ of concept names, a finite set $R$ of role names, and a finite set $I$ of individual names. We use letters like $A$ and $B$ for concept names, $r$ and $s$ for role names, and $a$ and $b$ for individual names. We also refer to $A$ and $B$ as atomic concepts, to $r$ and $s$ as roles, and to $a$ and $b$ as (named) individuals.

An (SHOQ) TBox $R$ is a finite set of role axioms of the form $r \subseteq s$ or $r \circ r \subseteq r$. For example, $\text{link} \sqsubseteq \text{path}$ and $\text{path} \circ \text{path} \subseteq \text{path}$ are such role axioms.

By $\text{ext}(R)$ we denote the least extension of $R$ such that:
- $r \subseteq r \in \text{ext}(R)$ for any role $r$
- if $r \subseteq r' \in \text{ext}(R)$ and $r' \subseteq r'' \in \text{ext}(R)$ then $r \subseteq r'' \in \text{ext}(R)$.

We write $r \subseteq_R s$ to denote $r \subseteq s \in \text{ext}(R)$, and $\text{trans}_R(r)$ to denote $(r \circ r \subseteq r) \in \text{ext}(R)$. If $r \subseteq_R s$ then $r$ is a subrole of $s$ (w.r.t. $R$). If $\text{trans}_R(s)$ then $s$ is a transitive role (w.r.t. $R$). A role is simple (w.r.t. $R$) if it is neither transitive nor has any transitive subrole (w.r.t. $R$).

Concepts in SHOQ are formed using the following BNF grammar, where $n$ is a nonnegative integer and $s$ is a simple role:

$$C, D ::= \top | \bot | A | \neg C | C \sqcap D | C \sqcup D | \exists r.C | \forall r.C | \{a\} | \geq n.s.C | \leq n.s.C$$

A concept stands for a set of individuals. The concept $\top$ stands for the set of all individuals (in the considered domain). The concept $\bot$ stands for the empty set. The constructors $\neg$, $\sqcap$ and $\sqcup$ stand for the set operators: complement, intersection and union. For the remaining forms, we just give some illustrative examples: $\exists \text{hasChild.Male}, \forall \text{hasChild.Female}, \geq 2 \text{hasChild.Teacher}$, $\leq 5 \text{hasChild.T}$.

We use letters like $C$ and $D$ to denote arbitrary concepts.

A TBox is a finite set of axioms of the form $C \sqsubseteq D$ or $C \equiv D$.

An ABox is a finite set of assertions of the form $a : C$, $r(a,b)$ or $a \neq b$. An eABox (extended ABox) is a finite set of assertions of the form $a : C$, $r(a,b)$, $\neg r(a,b)$, $a \equiv b$ or $a \neq b$.

An axiom $C \subseteq D$ means $C$ is a subconcept of $D$, while $C \equiv D$ means $C$ and $D$ are equivalent concepts. An assertion $a : C$ means $a$ is an instance of concept $C$, $r(a,b)$ means the pair $\langle a, b \rangle$ is an instance of role $r$, and $a \neq b$ means $a$ and $b$ are distinct individuals.

A knowledge base in SHOQ is a tuple $\langle R, T, A \rangle$, where $R$ is an RBox, $T$ is a TBox and $A$ is an ABox.

We say that a role $s$ is numeric w.r.t. a knowledge base $KB = \langle R, T, A \rangle$ if:
- it is simple w.r.t. $R$ and occurs in a concept $\geq n.s.C$ or $\leq n.s.C$ in $KB$, or
- $s \subseteq_R r$ and $r$ is numeric w.r.t. $KB$.

We will simply call such an $s$ a numeric role when $KB$ is clear from the context.

A formula is defined to be either a concept or an eABox assertion. We use letters like $\varphi$, $\psi$ and $\xi$ to denote formulas. Let $\text{null} : C$ stand for $C$. We use $\alpha$ to denote either an individual or null. Thus, $\alpha : C$ is a formula of the form $a : C$ or $\text{null} : C$ (which means $C$).

An interpretation $I = \langle \Delta^I, \cdot^I \rangle$ consists of a non-empty set $\Delta^I$, called the domain of $I$, and a function $\cdot^I$, called the interpretation function of $I$, that maps each concept name $A$ to a subset $A^I$ of $\Delta^I$, each role name $r$ to a binary relation $r^I$ on $\Delta^I$, and each individual name $a$ to an
element $a^I \in \Delta^I$. The interpretation function is extended to complex concepts as follows, where $\#Z$ denotes the cardinality of a set $Z$:

$$
\top^I = \Delta^I \quad \bot^I = \emptyset \quad (\neg C)^I = \Delta^I - C^I
$$

$$
(C \cap D)^I = C^I \cap D^I \quad (C \cup D)^I = C^I \cup D^I \quad \{a\}^I = \{a^I\}
$$

$$
(\exists r.C)^I = \{x \in \Delta^I \mid \exists y (\langle x, y \rangle \in r^I \text{ and } y \in C^I)\}
$$

$$
(\forall r.C)^I = \{x \in \Delta^I \mid \forall y (\langle x, y \rangle \in r^I \implies y \in C^I)\}
$$

$$
(\exists n s.C)^I = \{x \in \Delta^I \mid \#\{y \mid \langle x, y \rangle \in s^I \text{ and } y \in C^I\} \geq n\}
$$

$$
(\leq n s.C)^I = \{x \in \Delta^I \mid \#\{y \mid \langle x, y \rangle \in s^I \text{ and } y \in C^I\} \leq n\}
$$

For a set $\Gamma$ of concepts, define $I^\Gamma = \{x \in \Delta^I \mid x \in C^I \text{ for all } C \in \Gamma\}$.

The relational composition of binary relations $R_1$ and $R_2$ is denoted by $R_1 \circ R_2$.

An interpretation $I$ is a model of an RBox $\mathcal{R}$ if for every axiom $r \subseteq s$ (resp. $r \circ r \subseteq r$) of $\mathcal{R}$, we have that $r^I \subseteq s^I$ (resp. $r^I \circ r^I \subseteq r^I$). Note that if $I$ is a model of $\mathcal{R}$ then it is also a model of $\text{ext}(\mathcal{R})$.

An interpretation $I$ is a model of a TBox $\mathcal{T}$ if for every axiom $C \subseteq D$ (resp. $C \models D$) of $\mathcal{T}$, we have that $C^I \subseteq D^I$ (resp. $C^I = D^I$).

Given an interpretation $I$, define:

- $I \models a : \mathcal{C}$ iff $a^I \in C^I$
- $I \models r(a, b)$ iff $\langle a^I, b^I \rangle \in r^I$
- $I \models \neg r(a, b)$ iff $\langle a^I, b^I \rangle \notin r^I$
- $I \models a = b$ iff $a^I = b^I$
- $I \models a \neq b$ iff $a^I \neq b^I$.

If $I \models \varphi$ then we say that $I$ satisfies $\varphi$. An interpretation $I$ is a model of an eABox $\mathcal{A}$ if it satisfies all the assertions of $\mathcal{A}$. In that case, we also say that $I$ satisfies $\mathcal{A}$.

An interpretation $I$ is a model of a knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ if $I$ is a model of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}$.

A knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is satisfiable if it has a model.

An interpretation $I$ satisfies a concept $C$ (resp. a set $X$ of concepts) if $C^I \neq \emptyset$ (resp. $X^I \neq \emptyset$).

It validates a concept $C$ if $C^I = \Delta^I$. A set $X$ of concepts is satisfiable w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if there exists a model of $\mathcal{R}$ and $\mathcal{T}$ that satisfies $X$. We say that an eABox $\mathcal{A}$ is satisfiable w.r.t. an RBox $\mathcal{R}$ and a TBox $\mathcal{T}$ if there exists an interpretation $I$ that is a model of $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{T}$.

In that case, we also call $I$ a model of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$.

In this paper, we assume that concepts and ABox assertions are represented in negation normal form (NNF), where $\neg$ occurs only directly before atomic concepts. We use $\overline{C}$ to denote the NNF of $\neg C$, and for $\varphi = (a : \mathcal{C})$, we use $\overline{\varphi}$ to denote $a : \overline{\mathcal{C}}$. For simplicity, we treat axioms of a TBox $\mathcal{T}$ as concepts representing global assumptions: an axiom $C \subseteq D$ is treated as $\overline{C} \cup \overline{D}$, while an axiom $C \models D$ is treated as $(\overline{C} \cup D) \cap (\overline{D} \cup C)^\triangledown$. That is, we assume that $\mathcal{T}$ consists of concepts in NNF. A concept $C \in \mathcal{T}$ can be thought of as an axiom $\top \subseteq C$. Thus, an interpretation $I$ is a model of $\mathcal{T}$ iff $I$ validates every concept $C \in \mathcal{T}$.

2.2 An Integer Feasibility Problem for Description Logics

For dealing with number restrictions in $\mathcal{SHOQ}$, we consider the following integer feasibility problem, which was introduced in [18]:

$$
\sum_{j=1}^{m} a_{i,j} \cdot x_j \geq b_i, \quad \text{for } 1 \leq i \leq l;
$$

$$
x_j \geq 0, \quad \text{for } 1 \leq j \leq m;
$$

Every formula can be transformed to an equivalent formula in NNF in polynomial time.

As this way of handling the TBox is not efficient in practice, the absorption technique like the one discussed in [17] can be used to improve the performance of reasoning.
where each $a_{i,j}$ is either 0 or 1, each $x_j$ is a variable standing for a natural number, each $\vartriangleright_i$ is either $\leq$ or $\geq$, each $b_i$ is a natural number encoded by using no more than $n$ bits (i.e., $b_i \leq 2^n$).

We call this an IFDL($l,m,n$)-problem (a problem of Linear Integer Feasibility for Description Logics with size specified by $l,m,n$). The problem is feasible if it has a solution (i.e., values for the variables $x_j$, $1 \leq j \leq l$, that are natural numbers satisfying the constraints), and is infeasible otherwise. By solving an IFDL($l,m,n$)-problem we mean checking its feasibility.

It is known from linear programming that, if the variables $x_j$ are not required to be natural numbers but can be real numbers then the above feasibility problem can be solved in polynomial time in $l,m$ and $n$. The general integer linear optimization problem is known to be NP-hard.

To solve an integer feasibility problem, we propose to use the decomposition technique and the “branch and bound” method [14]. One can first analyze dependencies between the variables and the constraints to decompose the problem into smaller independent subproblems, then solve the subproblems that are trivial, and after that apply the “branch and bound” method [14] to the remaining subproblems.

The above mentioned approach may not guarantee that a given IFDL($l,m,n$)-problem is solved in exponential time in $n$. We recall below an estimation of the upper bound for the complexity for some specific cases, using another approach.

**Lemma 2.1 ([18]).** Every IFDL($l,m,n$)-problem such that $l \leq n$, $m$ is (at most) exponential in $n$, and $b_i \leq n$ for all $1 \leq i \leq l$ can be solved in (at most) exponential time in $n$.

**Proof.** Consider the following nondeterministic procedure:

1. initialize $c_{i,j} := 0$ for each $1 \leq i \leq l$ and $1 \leq j \leq m$ such that $a_{i,j} = 1$
2. for each $i$ from 1 to $l$ do
   for each $k$ from 1 to $b_i$ do
     choose some $j$ among $1,\ldots,m$ such that $a_{i,j} = 1$ and set $c_{i,j} := c_{i,j} + 1$
3. if the set of constraints $\{x_j \vartriangleright_i c_{i,j} \mid 1 \leq i \leq l, 1 \leq j \leq m, a_{i,j} = 1\}$ is feasible then return “yes”, else return “no”.

   Observe that the considered IFDL($l,m,n$)-problem is feasible iff there exists a run of the above procedure that returns “yes”. Since $b_i \leq n$ for all $1 \leq i \leq l$, there are no more than $m^{l-n}$ possible runs of the above procedure. All the steps of the procedure can be executed in time $O(l \cdot m \cdot n)$. Since $l \leq n$ and $m$ is (at most) exponential in $n$, we conclude that the considered IFDL($l,m,n$)-problem can deterministically be solved in (at most) exponential time in $n$. \qed

The following lemma is more general than the above lemma.

**Lemma 2.2 ([18]).** Every IFDL($l,m,n$)-problem satisfying the following properties can be solved in (at most) exponential time in $n$:

- $l \leq n$, $m$ is (at most) exponential in $n$,
- and
  - either $b_i \leq n$ for all $1 \leq i \leq l$ such that $\vartriangleright_i$ is $\leq$,
  - or $b_i \leq n$ for all $1 \leq i \leq l$ such that $\vartriangleright_i$ is $\geq$.

**Proof.** Suppose $l \leq n$, $m$ is (at most) exponential in $n$, and $b_i \leq n$ for all $1 \leq i \leq l$ such that $\vartriangleright_i$ is $\leq$. The other case is similar and omitted. Consider the following nondeterministic procedure:

1. let $J = \{j \mid 1 \leq j \leq m$ and there exists $1 \leq i \leq l$ such that $\vartriangleright_i$ is $\leq$ and $a_{i,j} = 1\}$
2. for each $1 \leq i \leq l$ and $1 \leq j \leq m$ such that $\vartriangleright_i$ is $\leq$ and $a_{i,j} = 1$, set $c_{i,j} := 0$
3. for each $i$ from 1 to $l$ such that $\vartriangleright_i$ is $\leq$, do
   for each $k$ from 1 to $b_i$ do
     choose some $j$ among $1,\ldots,m$ such that $a_{i,j} = 1$ and set $c_{i,j} := c_{i,j} + 1$

8. [http://en.wikipedia.org/wiki/Integer_programming](http://en.wikipedia.org/wiki/Integer_programming)
4. for each \( j \in J \) do
\[
d_j := \min \{ a_{i,j} \mid 1 \leq i \leq l, \exists a_i \text{ is } \leq \text{ and } a_{i,j} = 1 \}
\]
5. if the set of constraints \( \{ \sum_{j=1}^{m} a_{i,j} \cdot x_j \geq b_i \mid 1 \leq i \leq l, \exists \text{ is } \geq \} \cup \{ x_j = d_j \mid j \in J \} \) is feasible then return “yes”, else return “no”.

Observe that the considered IFDL(\( l, m, n \))-problem is feasible iff there exists a run of the above procedure that returns “yes”. Under the assumptions of the lemma, there are no more than \( m^{l\cdot n} \) possible runs of the above procedure. All the steps of the procedure can be executed in time \( O(l \cdot m \cdot n) \). Since \( l \leq n \) and \( m \) is (at most) exponential in \( n \), we conclude that the considered IFDL(\( l, m, n \))-problem can deterministically be solved in (at most) exponential time in \( n \).

\[\square\]

3 The Traditional Tableau Method and Its Problems

The problem we study is to check whether a given knowledge base \( KB = \langle R, \mathcal{T}, A \rangle \) in \( SHOQ \) is satisfiable. The traditional tableau method for this task is as follows. We start from the ABox \( A \) and try to modify it to obtain a model of \( KB \). At each moment, we have an ABox, which is like a graph. At the beginning, each (named) individual occurring in \( A \) is a node labeled by the set \( Label(a) = \{ C \mid a : C \in \mathcal{A} \} \cup \mathcal{T} \), and each assertion \( r(a, b) \in \mathcal{A} \) forms an edge from \( a \) to \( b \) that is labeled by \( r \). The concepts in \( Label(a) \) are treated as requirements to be realized for \( a \). As \( \mathcal{T} \) consists of the global assumptions that should be satisfied for all individuals, the concepts from \( \mathcal{T} \) are included in \( Label(a) \). For example, an axiom \( \top \subseteq \text{Human} \) is encoded in NNF as \( \text{Human} \), and such a global assumption states that all individuals in the domain should be human beings.

To see how the requirements for nodes can be realized, let us consider several cases:

- If \( C \cap D \in Label(v) \) then to realize the requirement \( C \cap D \) for \( v \) we add both \( C \) and \( D \) to \( Label(v) \). To see the intuition of this, assume that \( John \) is an individual and \( \text{Male} \cap \text{Happy} \in Label(John) \). In this case, \( John \) is required to satisfy the property \( \text{Male} \cap \text{Happy} \), and to realize this, we add both the requirements \( \text{Male} \) and \( \text{Happy} \) to \( Label(John) \).
- If \( C \cup D \in Label(v) \) then to realize the requirement \( C \cup D \) for \( v \) we add either \( C \) or \( D \) to \( Label(v) \). That is, we make an “or”-branching, which is dealt with by backtracking (since at each moment we consider only one ABox). If the current “or”-branch leads to inconsistency, we will backtrack to the nearest “or”-branching point and try another “or”-branch. To see the intuition of this, assume that \( \text{Doctor} \cup \text{Lawyer} \in Label(John) \). In this case, \( John \) is required to satisfy the property \( \text{Doctor} \cup \text{Lawyer} \), which states that he is either a doctor or a lawyer, and to realize this requirement, we make a choice: either add the requirement \( \text{Doctor} \) or add the requirement \( \text{Lawyer} \) to \( Label(John) \).
- If \( \exists r.C \in Label(v) \) then to realize the requirement \( \exists r.C \) for \( v \) we connect \( v \) to a new node \( w \) with \( Label(w) = \{ C \} \cup \mathcal{T} \) via an edge labeled by \( r \). Once again, \( \mathcal{T} \) is included in \( Label(w) \) because it consists of the global assumptions that should be realized for all individuals. (Instead of creating a new node, one may use an existing node for \( w \) as in the approach with global caching, but this should be done appropriately, e.g., as in our tableau method discussed in the next section. Alternatively, one can use a blocking technique as in [13,26].)

To see the intuition of the expansion, assume that \( \exists \text{hasChild}.\text{Female} \in Label(John) \). In this case, \( John \) should satisfy the requirement \( \exists \text{hasChild}.\text{Female} \), which states that he has a female child (a daughter). To realize this, we connect the node \( John \) of the graph to a new node \( w \) with \( Label(w) = \{ \text{Female} \} \cup \mathcal{T} \) via an edge labeled by \( \text{hasChild} \). From this, it can be seen that the graph contains not only named individuals occurring in \( \mathcal{A} \), but it may also contain nodes like \( w \), which are called unnamed individuals.
- If \( \forall r.C \in Label(v) \) then to realize the requirement \( \forall r.C \) for \( v \), for every node \( w \) such that there is an edge with the label \( r \) from \( v \) to \( w \), we add \( C \) to \( Label(w) \). To see the intuition of this, assume that \( \forall \text{hasChild}.\text{Happy} \in Label(John) \) and there are edges with the label
hasChild from the node John to the nodes Mary and w (i.e., Mary and w are children of John). In this case, John should satisfy the requirement $\neg$hasChild.Happy, which states that all the children of John should be happy. To realize this, we add the requirement Happy to both Label(Mary) and Label(w).

- If $\{a\} \in$ Label(v) then v and a should denote the same individual (this is the semantics of nominals), and to realize the requirement $\{a\}$ for v we merge the nodes v and a together in an appropriate way.

- If $\geq n \ r.C \in$ Label(v) then to realize the requirement $\geq n \ r.C$ for v we connect v to n new nodes $w_1, \ldots, w_n$ with $\text{Label}(w_i) = \{C\} \cup T$ via an edge labeled by r for all $1 \leq i \leq n$, and keep the constraints $w_i \neq w_j$ for all $1 \leq i \neq j \leq n$. (Once again, an appropriate caching or blocking technique can be used to reduce the number of created nodes.) To see the intuition of the expansion, assume that $\geq 2 \ hasChild.Female \in$ Label(John). In this case, John should satisfy the requirement $\geq 2 \ hasChild.Female$, which states that he has at least two female children (daughters). To realize this, we connect the node John of the graph to new nodes $w_1$ and $w_2$ with $\text{Label}(w_1) = \text{Label}(w_2) = \{Female\} \cup T$ via an edge labeled by hasChild and keep the constraint $w_1 \neq w_2$.

- If $\leq n \ r.C \in$ Label(v) and there are pairwise different nodes $w_1, \ldots, w_{n+1}$ such that v is connected to $w_i$ via an edge labeled by r and C $\in$ Label($w_i$) for all $1 \leq i \leq n + 1$, then:
  - if there exist different i and j among $1, \ldots, n$ such that the constraint $w_i \neq w_j$ is absent then we merge $w_i$ and $w_j$ together in an appropriate way,
  - otherwise, the current ABox is inconsistent and we do backtracking.

Inconsistency may occur, for example, in the following cases:

- when $\bot \in$ Label(v) for some v; or
- when $\{A, \neg A\} \subseteq$ Label(v) for some A and v; or
- when a and b were merged together, but the assertion $a \neq b$ is a kept constraint; or
- when the current ABox contains an edge with the label r from a to b, but $(\neg r(a, b)) \in A$.

As mentioned before, when the current ABox is inconsistent, backtracking occurs, and if there is no “or”-branching point to come back, the process terminates with the result “KB is unsatisfiable”.

The above discussion only gives a sketch on how the traditional tableau method works. We did not discuss how role axioms can be dealt with and how a blocking technique can be applied to guarantee termination. Furthermore, merging nodes causes merging edges, and hence an edge may be labeled by a set of roles. In general, a tableau algorithm is usually designed so that, if it does not terminate with the result “KB is unsatisfiable”, then KB is satisfiable and we can directly construct a model of KB from the resulting (clash-free and completed) ABox. We refer the reader to [13,20] for details.

The traditional tableau method for SHOQ has the advantage of being intuitive, but it has two disadvantages that make the complexity non-optimal (N2ExpTime or NExpTime, depending on the applied blocking technique, in comparison with the optimal complexity ExpTime) and the reasoning process not scalable w.r.t. number restrictions:

- An ABox is like an “and”-structure (i.e., all of its assertions must hold together) and the search space for the traditional tableau method is an “or”-tree of “and”-structures. Recall that “or”-branchings are caused, amongst others, by the rule for realizing requirements of the form $C \cup D$. The problem is that two nodes in ABoxes in different “or”-branches may have the same label and the same “neighborhood”, and both of them are expanded with no reuse, which causes a kind of redundant computation [8].

- Reconsider the traditional tableau rule for realizing a requirement of the form $\geq n \ r.C$. If n is big, for example, 1000 or 1000000, then the rule creates many new nodes. In the DL literature, this is called “pay-as-you-go”, but this payment is unnecessarily too high when n is big and it causes the reasoning process not scalable w.r.t. number restrictions.
4 ExpTime Tableaux for \( \text{SHOQ} \)

In this section, we first define the data structures and outline the framework of our tableau method. We then describe our techniques for dealing with nominals. After that, we specify the used tableau rules and state properties of the resulting tableau decision procedure.

4.1 Data Structures and the Tableau Framework

Recall that the search space for the traditional tableau method for \( \text{SHOQ} \) \cite{NGUYEN201826} is an “or”-tree of “and”-structures, and this causes the complexity of the reasoning process to become non-optimal (even in the case without number restrictions). The idea for overcoming this problem is to use global caching \cite{NGUYEN201826,NGUYEN201827}. With global caching, the search space is like a single “and-or” graph. For checking satisfiability of a concept w.r.t. an RBox and a TBox \cite{NGUYEN201828}, each node of the graph is a simple node like an individual (in an ABox). For checking satisfiability of a knowledge base \cite{NGUYEN201829,NGUYEN201830}, each node of the graph is either a complex node like an eBox, or a simple node like an individual. More precisely, the label of a complex node is a set of eBox assertions, while the label of a simple node is a set of concepts. The information about whether a node \( v \) is complex or simple is kept by \( \text{SType}(v) \) (the subtype of \( v \)).

At the beginning, the graph has only one node, called the root, which is a complex node. Then, in the first stage, complex nodes are expanded only by so called static (tableau) rules that do not create new (unnamed) individuals. This creates a layer of complex nodes. When no static tableau rules are applicable to a complex node \( v \), if \( \text{Label}(v) \) contains a requirement of the form \( a : \exists r.C \) then to realize this requirement we can connect \( v \) to a simple node \( w \) with \( \text{Label}(w) = \{ C \} \cup \mathcal{T} \) via an edge \( e \). This edge is related to \( a \) and \( r \). To keep this information we store \( \pi_I(e) = a \) (the letter \( \pi \) stands for “projection” and the letter \( I \) stands for “individual”) and \( \pi_R(e) = \{ r \} \) (the letter \( R \) stands for “roles”; as mentioned earlier, due to merging nodes, an edge may be labeled by more than one role, and hence we use a set of roles).

A transitional (tableau) rule is a rule that realizes a requirement of the form \( a : \exists r.C, a : (\geq n r.C), \exists r.C \text{ or } \geq n r.C \) for a node \( v \) by connecting \( v \) to a new node or a number of new nodes or by using some existing nodes. If no static rules are applicable to a node \( v \) then \( v \) is called a state, otherwise it is called a non-state. This information is kept by \( \text{Type}(v) \) (the type of \( v \)). Transitional tableau rules are applied only to states. A non-state is like an “or”-node in an “and-or” graph, but a state is a structure more sophisticated than an “and”-node in an “and-or” graph (due to feasibility checking of the set of integer linear constraints related to the state)\(^9\).

Consider a state \( v \) (i.e., a state that is a simple node) with \( \exists r.C \in \text{Label}(v) \). To realize this requirement for \( v \), we can connect \( v \) to a new simple node \( w \) with \( \text{Label}(w) = \{ C \} \cup \mathcal{T} \) by an edge \( e \). For such an edge \( e \), let \( \pi_I(e) = \text{null} \) (i.e., no named individual is related to \( e \)).

Consider a state \( v \). To realize requirements of the form \( a : (\geq n r.C), a : (\leq n r.C), \geq n r.C \text{ or } \leq n r.C \) for \( v \), we may have to connect \( v \) to some simple nodes \( w_i \) by edges \( e_i \), respectively, and check feasibility of a certain set of integer linear constraints. The set of integer linear constraints for \( v \) is kept by \( \text{ILConstraints}(v) \). For such mentioned edges \( e_i \), let \( \pi_T(e_i) = \text{testingClosedness} \) (the letter \( T \) stands for “type”). For other edges \( e \), which are created for realizing a requirement of the form \( a : \exists r.C \text{ or } \exists r.C, \) let \( \pi_T(e) = \text{testingClosedness} \).

We have explained the attributes \( \pi_T(e), \pi_R(e) \) and \( \pi_I(e) \) that should be kept for an edge \( e \) outgoing from a state. Summing up, we have the following formal definition:

**Definition 4.1.** Let \( \text{EdgeLabels} = \{ \text{testingClosedness}, \text{checkingFeasibility} \} \times \mathcal{P}(\mathbb{R}) \times (\mathbb{I} \cup \{ \text{null} \}) \). For \( e \in \text{EdgeLabels} \), let \( e = (\pi_T(e), \pi_R(e), \pi_I(e)) \). Thus, \( \pi_T(e) \) is called the type of the edge label \( e, \pi_R(e) \) is a set of roles, and \( \pi_I(e) \) is either an individual or null. (Each edge is specified by the source, the target and the label.) \( \square \)

\(^9\) In tableaux for simpler DLs like \( \text{ALC} \) \cite{NGUYEN201828} or \( \text{SHIT} \) \cite{NGUYEN201830}, a state is simply an “and”-node.
We have explained the attributes \( Label(v) \), \( Type(v) \), \( SType(v) \) and \( ILConstraints(v) \) for a node \( v \). We need three more attributes for \( v \), which are described and justified below.

1. To realize the requirement \( C \cup D \in Label(v) \) for a simple node \( v \), we expand \( v \) by a static rule that connects \( v \) to two simple nodes \( w_1 \) and \( w_2 \) such that \( Label(w_1) = Label(v) \cup \{C\} \setminus \{C \cup D\} \) and \( Label(w_2) = Label(v) \cup \{D\} \setminus \{C \cup D\} \). The requirement \( C \cup D \) is put to the sets \( RFmls(w_1) \) and \( RFmls(w_2) \) to record that it has been realized for \( w_1 \) and \( w_2 \), respectively. In general, the attribute \( RFmls(w) \) for a node \( w \) keeps the set of the requirements that have been realized for \( w \) by using static rules. It is called the set of reduced formulas of \( w \).

2. Suppose \( v \) is a complex node and either \( a \neq b \) or \( a : \{b\} \) belongs to \( Label(v) \). Then, to realize that requirement for \( v \), we merge the individual \( b \) to the individual \( a \) in an appropriate way and record this fact by keeping \( IndRepl(v)(b) = a \). The attribute \( IndRepl(v) \) is called the partial mapping specifying replacements of individuals for the node \( v \).

3. The last attribute needed for a node \( v \) is called the status of \( v \) and denoted by \( Status(v) \). Possible statuses of nodes are: unexpanded, partially-expanded, fully-expanded, closed, open, blocked, and closed w.r.t. a set of complex states. Informally, closed means “unsatisfiable w.r.t. \( R \) and \( T \)”, open means “satisfiable w.r.t. \( R \) and \( T \)”, and closed-wrt(\( U \)) means “unsatisfiable w.r.t. \( R \), \( T \) and any node from \( U \)”.

We arrive at the following formal definition.

**Definition 4.2.** A tableau is a rooted graph \( G = (V,E,v) \), where \( V \) is a set of nodes, \( E \subseteq V \times V \) is a set of edges, \( v \in V \) is the root, each node \( v \in V \) has a number of attributes, and each edge \( \langle v,w \rangle \) may have a number of labels from \( EdgeLabels \). The attributes of a tableau node \( v \) are:

- \( Type(v) \in \{\text{state, non-state}\} \).
- \( SType(v) \in \{\text{complex, simple}\} \) is called the subtype of \( v \).
- \( Status(v) \in \{\text{unexpanded, p-expanded, f-expanded, closed, open, blocked}\} \cup \{\text{closed-wrt}(U) \mid U \subseteq V \text{ and } Type(u) = \text{state} \land SType(u) = \text{complex} \text{ for all } u \in U\} \), where p-expanded and f-expanded mean “partially expanded” and “fully expanded”, respectively. \( Status(v) \) may be p-expanded only when \( Type(v) = \text{state} \). If \( Status(v) = \text{closed-wrt}(U) \) then we say that the node \( v \) is closed w.r.t. any node from \( U \).
- \( Label(v) \) is a finite set of formulas called the label of \( v \).
- \( RFmls(v) \) is a finite set of formulas called the set of reduced formulas of \( v \).
- \( IndRepl(v) : I \rightarrow I \) is a partial mapping specifying replacements of individuals. It is available only when \( v \) is a complex node. If \( IndRepl(v)(a) = b \) then at the node \( v \) we have \( a \neq b \) and \( b \) is the representative of its equivalence class.
- \( ILConstraints(v) \) is a set of integer linear constraints. It is available only when \( Type(v) = \text{state} \). The constraints use variables \( x_{w,e} \) indexed by a pair \( \langle w,e \rangle \) such that \( \langle v,w \rangle \in E \), \( e \in ELabels(v,w) \) and \( \pi_T(e) = \text{checkingFeasibility} \). Such a variable specifies how many copies of the successor \( w \) using the edge label \( e \) will be created for \( v \). \( \square \)

If \( \langle v,w \rangle \in E \) then we call \( v \) a predecessor of \( w \) and \( w \) a successor of \( v \). An edge outgoing from a node \( v \) has labels iff \( Type(v) = \text{state} \). When defined, the set of labels of an edge \( \langle v,w \rangle \) is denoted by \( ELabels(v,w) \). If \( e \in ELabels(v,w) \) then \( \pi_I(e) = \text{null} \) iff \( SType(v) = \text{simple} \).

Formally, a node \( v \) is called a state if \( Type(v) = \text{state} \), and a non-state otherwise. It is called a complex node if \( SType(v) = \text{complex} \), and a simple node otherwise. The root \( v \) is a complex non-state.

A node may have status blocked only when it is a simple node with the label containing a nominal \( \{a\} \). The status blocked can be updated only to closed or closed-wrt(\( \ldots \)). We write \( closed-wrt(\ldots) \) to mean closed-wrt(\( U \)) for some \( U \). By \( Status(v) \neq closed-wrt(\{u, \ldots\}) \) we denote that \( Status(v) \) is not of the form closed-wrt(\( U \)) with \( u \in U \).

---

An edge \( \langle v,w \rangle \) may have a number of labels from \( EdgeLabels \) because of global caching, which we will briefly discuss later.
The graph $G$ consists of two layers: the layer of complex nodes and the layer of simple nodes. There are no edges from simple nodes to complex nodes. The edges from complex nodes to simple nodes are exactly the edges outgoing from complex states. That is, if $(v, w)$ is an edge from a complex node $v$ to a simple node $w$ then $Type(v) = state$, if $Type(v) = state$ and $(v, w) \in E$ then $SType(w) = simple$. Each complex node of $G$ is like an eABox (more formally, its label is an eABox), which can be treated as a graph whose vertices are named individuals. On the other hand, a simple node of $G$ stands for an unnamed individual. If $e$ is a label of an edge from a complex state $v$ to a simple node $w$ then the triple $(v, e, w)$ can be treated as an edge from the named individual $\pi_I(e)$ (an inner node in the graph representing $v$) to the unnamed individual corresponding to $w$, and that edge is via the roles from $\pi_R(e)$.

We will use also assertions of the form $a:(\leq n.s.C)$ and $a:(\geq n.s.C)$, where $s$ is a numeric role. The difference between $a:(\leq n.s.C)$ and $a:(\leq n.s.C)$ is that, for checking $a:(\leq n.s.C)$, we do not have to pay attention to assertions of the form $s(a, b)$ or $r(a, b)$ with $r$ being a subrole of $s$. The aim for $a:(\geq n.s.C)$ is similar. We use $a:(\leq n.s.C)$ and $a:(\geq n.s.C)$ only as syntactic representations of some expressions, and do not provide semantics for them. We define

$$FullLabel(v) = Label(v) \cup RFmls(v) - \{\text{formulas of the form } a:(\leq n.s.C) \text{ or } a:(\geq n.s.C)\}.$$ 

We apply global caching: if $v_1, v_2 \in V$, $Label(v_1) = Label(v_2)$ and $(SType(v_1) = SType(v_2) = simple \text{ or } (SType(v_1) = SType(v_2) = complex \text{ and } Type(v_1) = Type(v_2)))$ then $v_1 = v_2$. Due to global caching, an edge outgoing from a state may have a number of labels as the result of merging edges. Creation of a new node or a new edge is done by Procedure ConToSucc (connect to a successor) given on page 10. This procedure creates a connection from a node $v$ given as the first parameter to a node $w$ with $Type(w)$, $SType(w)$, $Label(w)$, $RFmls(w)$, $IndRepl(w)$, $ELabels(v, w)$ specified by the remaining parameters.

We say that a node $v$ may affect the status of the root $v$ if there exists a path consisting of nodes $v_0 = v, v_1, \ldots, v_{n-1}$, $v_n = v$ such that, for every $0 \leq i < n$, $Status(v_i)$ differs from open and closed, and if it is closed-wrt($U$) then $U$ is disjoint from $\{v_0, \ldots, v_i\}$. In that case, if $u \in \{v_1, \ldots, v_n\}$ then we say that $v$ may affect the status of the root $v$ via a path through $u$.

From now on, let $\langle R, T, A \rangle$ be a knowledge base in NNF of the logic $SHOQ$, with $A \neq \emptyset$. In this section we present a tableau calculus $C_{SHOQ}$ for checking satisfiability of $\langle R, T, A \rangle$. A $C_{SHOQ}$-tableau for $\langle R, T, A \rangle$ is a rooted graph $G = \langle V, E, v \rangle$ constructed as follows.

---

11 If $A$ is empty, we can add $a: \top$ to it, where $a$ is a special individual.
Initialization: Set $V := \emptyset$ and $E := \emptyset$. Then, create the root node by executing $\nu := \mathit{ConToSucc}(\text{null, non-state, complex, label}, \emptyset, \text{null, null})$, where $\mathit{label} = \mathcal{A} \cup \{(a : C) \mid C \in \mathcal{T} \text{ and } a \text{ is an individual }\}$ occurring in $\mathcal{A}$ or $\mathcal{T}$.

Rules’ Priorities and Expansion Strategies: The graph is then expanded by the following rules, which will be specified shortly:

(UPS) rules for updating statuses of nodes,
(US) unary static expansion rules,
(DN) a rule for dealing with nominals,
(NUS) a non-unary static expansion rule,
(FS) the forming-state rule,
(TP) a transitional partial-expansion rule,
(TF) a transitional full-expansion rule.

Each of the rules is parametrized by a node $v$. We say that a rule is applicable to $v$ if it can be applied to $v$ to make changes to the graph. The rule (UPS) has a higher priority than (US), which has a higher priority than the remaining rules in the list. If neither (UPS) nor (US) is applicable to any node, then choose a node $v$ with status unexpanded or p-expanded, choose the first rule applicable to $v$ among the rules in the last five items of the above list, and apply it to $v$. Any strategy can be used for choosing $v$, but it is worth to choose $v$ for expansion only when $v$ may affect the status of the root $\nu$ of the graph. Note that the priorities of the rules are specified by the order in the above list, but the rules (UPS) and (US) are checked globally (technically, they are triggered immediately when possible), while the remaining rules are checked for a chosen node.

The construction of the graph ends when the root $\nu$ receives the status closed or open or when no more changes that may affect the status of $\nu$ can be made.” Theorem 4.7 states that the knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff $\mathit{Status}(\nu) \neq \text{closed}$.

4.2 Techniques for Dealing with Nominals

As usual, to deal with assertions of the form $a : \{b\}$ (resp. $a : \neg\{b\}$) we use the predicate $\mathit{\equiv}$ (resp. $\neq$) and the replacement technique. Recall also that we use statuses of the form closed-wrt$(U)$ in order to be able to apply global caching in the presence of nominals. Updating statuses of nodes is defined appropriately. Our other techniques for dealing with nominals are described below.

Suppose $v$ is a simple node with $\mathit{Status}(v) \notin \{\text{closed, open}\}$ and $\{a\} \in \mathit{Label}(v)$, a complex state $u$ is an ancestor of $v$, and $v$ may affect the status of the root $\nu$ via a path through $u$. Let $u_0$ be a predecessor of $u$. The node $u_0$ has only $u$ as a successor and it was expanded by the forming-state rule. There are three cases:

- If, for every $C \in \mathit{Label}(v)$, the formula obtained from $a : C$ by replacing every individual $b$ with $\mathit{IndRepl}(u)(b)$, when $\mathit{IndRepl}(u)(b)$ is defined, belongs to $\mathit{FullLabel}(u)$, then $v$ is “consistent” with $u$.
- If there exists $C \in \mathit{Label}(v)$ such that the formula obtained from $a : \overline{C}$ (where $\overline{C}$ is the negation of $C$ in NNF) by replacing every individual $b$ with $\mathit{IndRepl}(u)(b)$, when $\mathit{IndRepl}(u)(b)$ is defined, belongs to $\mathit{FullLabel}(u)$, then $v$ is “inconsistent” with $u$. In this case, if $\mathit{Status}(v)$ is of the form closed-wrt$(U)$ then we update it to closed-wrt$(U \cup \{u\})$, else we update it to closed-wrt$(\{u\})$.

12 That is, ignoring nodes that are unreachable from $\nu$ via a path without nodes with status closed or open, no more changes can be made to the graph.
In the remaining case, the node \( u \) is “incomplete” w.r.t. \( v \), which means that the expansion of \( u_0 \) was not appropriate. Thus, we delete the edge \( \langle u_0, u \rangle \) and re-expand \( u_0 \) by an appropriate “or”-branching.

For dealing with interaction between number restrictions and nominals, to guarantee that every nominal represents a singleton set and a named individual cannot be cloned we use concepts of the form \( \leq 1 r.\{a\} \) and assertions of the form \( a: \leq 1 r.\{b\} \) that are relevant w.r.t. the TBox \( T \) and the label of the considered node. One can define this relation to be the full one (i.e., so that such formulas are always relevant). However, to increase efficiency we define this notion as follows.

Let \( X \) be a set of formulas. We say that a formula \( \varphi \) occurs positively at the modal depth 0 in \( X \) if there exist \( \psi \in X \) and an occurrence of \( \varphi \) in \( \psi \) that is not in the scope of \( \neg, \exists r, \forall r, \geq r n, \leq r n \) for any \( r \in \mathbb{R} \). (Recall that formulas are in NNF.)

**Definition 4.3.** We say that a concept \( \leq 1 r.\{a\} \) is relevant w.r.t. a TBox \( T \) and a set \( X \) of concepts if the following conditions hold:

- either of the following conditions holds:
  - some concept \( \geq m s. C \) with \( m \geq 2 \) and \( s \subseteq_R r \) occurs positively at the modal depth 0 in \( X \) – in this case, let \( s_1 = s_2 = s \) and \( C_1 = C_2 = C \),
  - some different concepts \( C_1' \) and \( C_2' \) occur positively at the modal depth 0 in \( X \) and each \( C_i' \) is of the form \( \exists s_i, C_i \) or \( \geq 1 s_i, C_i \) with \( s_i \subseteq_R r \);

- one of the following conditions holds:
  - the nominal \( \{a\} \) occurs positively at the modal depth 0 in \( T \) or \( \{C_1\} \) or \( \{C_2\} \),
  - some concept \( \forall r'. D \) satisfying \( s_1 \subseteq r' \) and \( s_2 \subseteq r' \) occurs positively at the modal depth 0 in \( T \) and the nominal \( \{a\} \) occurs positively at the modal depth 0 in \( \{D\} \),
  - some concept \( \leq n r'. D \) satisfying \( s_1 \subseteq r' \) and \( s_2 \subseteq r' \) occurs positively at the modal depth 0 in \( T \) and either \( \{a\} \) or \( \neg\{a\} \) occurs positively at the modal depth 0 in \( \{D\} \).

A concept \( a: \leq 1 r.\{b\} \) is relevant w.r.t. a TBox \( T \) and a set \( X \) of eABox assertions if the concept \( \leq 1 r.\{b\} \) is relevant w.r.t. \( T \) and the set \( \{C \mid a: C \in X\} \).

Note that every interpretation \( \mathcal{I} \) always validates \( \leq 1 r.\{a\} \) (i.e., \( (\leq 1 r.\{a\})^\mathcal{I} = \Delta^\mathcal{I} \)) and satisfies \( a: \leq 1 r.\{b\} \) (i.e., \( \mathcal{I} \models a: \leq 1 r.\{b\} \)).

### 4.3 Illustrative Examples

Before specifying the tableau rules in detail, we present simple examples to illustrate some ideas (but not all aspects) of our method. Despite that these examples refer to the tableau rules, we choose this place for presenting them because the examples are quite intuitive and the reader can catch the ideas of our method without knowing the detailed rules. He or she can always consult the rules in the next subsection.

**Example 4.4.** Let us construct a \( C_{SHOQ} \)-tableau for \( \langle R, T, A \rangle \), where

\[
A = \{ a: A, a: \exists r. \exists A \cup \{a\}, a: \geq 3 r. \forall r. \neg A, a: \forall r. B, a: \leq 3 r. B, r(a, b), b: \forall r. \neg A, b: (\forall r. (\neg A \cap \neg\{a\}) \cup \neg B) \},
\]

\( R = \emptyset \) and \( T = \emptyset \). An illustration is presented in Figure 1.

At the beginning, the graph has only the root \( \nu \) which is a complex non-state with \( Label(\nu) = A \). Since \( \{a: \forall r. B, r(a, b)\} \subseteq Label(\nu) \), applying a unary static expansion rule to \( \nu \), we connect it to a new complex non-state \( \nu_1 \) with \( Label(\nu_1) = Label(\nu) \cup \{b: B\} \).
Fig. 1. An illustration of the tableau described in Example 4.4. The marked nodes $v_4$–$v_7$ and $v_9$ are states. The nodes $v$, $v_1$–$v_4$ are complex nodes, the remaining are simple nodes. In each node, we display the formulas of its label.

Since $b : (\forall r. (\neg A \sqcap \neg \{a\}) \sqcup \neg B) \in \text{Label}(v_1)$, applying the non-unary static expansion rule to $v_1$, we connect it to new complex non-states $v_2$ and $v_3$ with

$$\text{Label}(v_2) = \text{Label}(v_1) - \{b : (\forall r. (\neg A \sqcap \neg \{a\}) \sqcup \neg B)\} \cup \{b : (\forall r. (\neg A \sqcap \neg \{a\})\}$$

$$\text{Label}(v_3) = \text{Label}(v_1) - \{b : (\forall r. (\neg A \sqcap \neg \{a\}) \sqcup \neg B)\} \cup \{b : \neg B\}.$$ 

Since both $b : B$ and $b : \neg B$ belong to $\text{Label}(v_3)$, the node $v_3$ receives the status closed. Applying the forming-state rule to $v_2$, we connect it to a new complex state $v_4$ with

$$\text{Label}(v_4) = \text{Label}(v_2) \cup \{a : \geq 1 r. \exists r. (A \sqcup \{a\})\}, a : \geq 2 r. \forall r. \neg A, a : \leq 2 r. B\}.$$ 

The assertion $a : \geq 1 r. \exists r. (A \sqcup \{a\}) \in \text{Label}(v_4)$ is due to $a : \exists r. \exists r. (A \sqcup \{a\}) \in \text{Label}(v_2)$ and the fact that the negation of $b : \exists r. (A \sqcup \{a\})$ in NNF belongs to $\text{Label}(v_2)$ (notice that $r(a, b) \in \text{Label}(v_2)$).

The assertion $a : \geq 2 r. \forall r. \neg A \in \text{Label}(v_4)$ is due to $a : \geq 3 r. \forall r. \neg A \in \text{Label}(v_2)$ and the fact that $\{r(a, b), b : \forall r. \neg A\} \subset \text{Label}(v_2)$. Similarly, the assertion $a : \leq 2 r. B \in \text{Label}(v_4)$ is due to $a : \leq 3 r. B \in \text{Label}(v_2)$ and the fact $\{r(a, b), b : B\} \subset \text{Label}(v_2)$.

As $r$ is a numeric role, applying the transitional partial-expansion rule\[13\] to $v_4$, we just change the status of $v_4$ to p-expanded. After that, applying the transitional full-expansion

\[13\] which is used for making transitions via non-numeric roles
rule to \(v_4\), we connect it to new simple non-states \(v_5, v_6, v_7\), using the edge label \(e = \langle \text{checkingFeasibility}, \{r\}, a \rangle\), such that \(\text{Label}(v_5) = \{3r.(A \cup \{a\}), B\}, \text{Label}(v_6) = \{\forall r.\neg A, B\}\), \(\text{Label}(v_7) = \{3r.(A \cup \{a\}), \forall r.\neg A, B\}\). The creation of \(v_5\) is caused by \(a : \geq 1 r.\exists r.(A \cup \{a\}) \in \text{Label}(v_4)\), while the creation of \(v_6\) is caused by \(a : \geq 1 r.\forall r.\neg A\). The node \(v_7\) results from merging \(v_5\) and \(v_6\). Furthermore, \(\text{ILConstraints}(v_4)\) consists of \(x_{v_i,e} \geq 0\), for \(5 \leq i \leq 7\), and
\[
\begin{align*}
x_{v_5,e} + x_{v_7,e} & \geq 1 \\
x_{v_6,e} + x_{v_7,e} & \geq 2 \\
x_{v_5,e} + x_{v_6,e} + x_{v_7,e} & \leq 2.
\end{align*}
\]

Applying the forming-state rule to \(v_5\), the type of this node is changed from \text{non-state} to \text{state}. Next, applying the transitional partial-expansion rule to \(v_5\), its status is changed to \text{p-expanded}. Then, applying the transitional full-expansion rule to \(v_5\), we connect \(v_5\) to a new simple non-state \(v_8\) with \(\text{Label}(v_8) = \{A \cup \{a\}\}\) using the edge label \(e' = \langle \text{checkingFeasibility}, \{r\}, \text{null} \rangle\) and set \(\text{ILConstraints}(v_5) = \{x_{v_8,e'} \geq 0, x_{v_8,e'} \geq 1\}\).

Applying the non-unary static expansion rule to \(v_8\), we connect it to new simple non-states \(v_9\) and \(v_{10}\) with \(\text{Label}(v_9) = \{A\}\) and \(\text{Label}(v_{10}) = \{\{a\}\}\). The status of \(v_9\) is then changed to \text{open}, which causes the statuses of \(v_8\) and \(v_5\) to be updated to \text{open}. The node \(v_{10}\) is not expanded as it does not affect the status of the root node \(\nu\).

Applying the forming-state rule to \(v_6\), the type of this node is changed from \text{non-state} to \text{state}. Next, applying the transitional partial-expansion rule and then the transitional full-expansion rule to \(v_6\), its status is changed to \text{f-expanded}. The status of \(v_6\) is then updated to \text{open}.

Applying the forming-state rule to \(v_7\), the type of this node is changed from \text{non-state} to \text{state}. Next, applying the transitional partial-expansion rule to \(v_7\), its status is changed to \text{p-expanded}. Then, applying the transitional full-expansion rule to \(v_7\), we connect \(v_7\) to a new simple non-state \(v_{11}\) with \(\text{Label}(v_{11}) = \{A \cup \{a\}, \neg A\}\) using the mentioned edge label \(e'\) and set \(\text{ILConstraints}(v_7) = \{x_{v_{11},e'} \geq 0, x_{v_{11},e'} \geq 1\}\).

Applying the non-unary static expansion rule to \(v_{11}\), we connect it to new simple non-states \(v_{12}\) and \(v_{13}\) with \(\text{Label}(v_{12}) = \{A, \neg A\}\) and \(\text{Label}(v_{13}) = \{\{a\}, \neg A\}\). The status of \(v_{12}\) is then changed to \text{closed}. Since \(a : A \in \text{Label}(v_4)\), the status of \(v_{13}\) is updated to \text{closed-wrt}(\{v_4\}), which causes the status of \(v_{11}\) to be updated also to \text{closed-wrt}(\{v_4\}). As the set \(\text{ILConstraints}(v_7) \cup \{x_{v_{11},e'} = 0\}\) is infeasible, the status of \(v_7\) is updated to \text{closed-wrt}(\{v_4\}). Next, as the set \(\text{ILConstraints}(v_4) \cup \{x_{v_7,e} = 0\}\) is infeasible, the status of \(v_4\) is first updated to \text{closed-wrt}(\{v_4\}) and then to \text{closed}. After that, the statuses of \(v_2, v_3, v_4\) are sequentially updated to \text{closed}. Thus, we conclude that the knowledge base \((\mathcal{R}, \mathcal{T}, A)\) is unsatisfiable.

Example 4.5. Let us modify Example 4.4 by deleting the assertion \(a : A\) from the ABox. That is, we are now constructing a \(\text{C}_{\text{SHOQ}}\)-tableau for \((\mathcal{R}, \mathcal{T}, A)\), where
\[
\mathcal{A} = \{a : \exists r.\exists r.(A \cup \{a\}), a : \geq 3 r.\forall r.\neg A, a : \forall r.B, a : \leq 3 r.B, r(a, b), b : \forall r.\neg A, b : \forall r.\neg A \cap \{a\} \cup \neg B\}.
\]
\(\mathcal{R} = \emptyset\) and \(\mathcal{T} = \emptyset\). The first stage of the construction is similar to the one of Example 4.4 up to the step of updating the status of \(v_{12}\) to \text{closed}. This stage is illustrated in Figure 2 which is similar to Figure 1 except that the labels of the nodes \(\nu\) and \(v_1 - v_4\) do not contain \(a : A\). The continuation is described below and illustrated by Figure 3.

Since \(\text{Label}(v_{13}) = \{\{a\}, \neg A\}\), applying the rule for dealing with nominals to \(v_{13}\), we delete the edge \((v_2, v_4)\) (from \(E\)) and re-expand \(v_2\) by connecting it to new complex non-states \(v_{14}\) and \(v_{15}\) with \(\text{Label}(v_{14}) = \text{Label}(v_2) \cup \{a : \neg A\}\) and \(\text{Label}(v_{15}) = \text{Label}(v_2) \cup \{a : A\}\) as shown in Figure 3. The status of \(v_{13}\) is updated to \text{blocked}. The node \(v_4\) is not deleted, but we do not display it in Figure 3.
Applying the forming-state rule to \( v_{14} \) we connect it to a new complex state \( v_{16} \). The label of \( v_{16} \) is computed using \( \text{Label}(v_{14}) \) in a similar way as in Example 4.4 when computing \( \text{Label}(v_4) \).

Applying the transitional partial-expansion rule to \( v_{16} \) we change its status to \( p\text{-expanded} \). After that, applying the transitional full-expansion rule to \( v_{16} \) we connect it to the existing nodes \( v_5, v_6, v_7 \) using the edge label \( e = \langle \text{checkingFeasibility}, \{r\}, a \rangle \). The set \( \text{ILConstraints}(v_{16}) \) is the same as \( \text{ILConstraints}(v_4) \) and \( \text{ILConstraints}(v_{17}) \). Analogously to updating the statuses of the nodes \( v_{13}, v_{11}, v_7 \) in Example 4.4 to \( \text{closed-wrt}(\{v_1\}) \), the statuses of \( v_{13}, v_{11}, v_7 \) are updated to \( \text{closed-wrt}(\{v_{17}\}) \). Next, as \( \text{ILConstraints}(v_{17}) \cup \{x_{v_7,e} = 0\} \) is infeasible, the status of \( v_{17} \) is first updated to closed-wrt(\{v_{17}\}) and then to closed. After that, the status of \( v_{15} \) is also updated to closed. As no more changes that may affect the status of \( \nu \) can be made and \( \text{Status}(\nu) \neq \text{closed} \), we conclude that the knowledge base \( \langle R, T, A \rangle \) is satisfiable. \( \square \)
4.4 Tableau Rules

In this subsection we formally specify the tableau rules of our calculus $C_{SHOQ}$. We also give explanations for them. They are informal and should be understood in the context of the described rule.

We will use the auxiliary procedure $\text{SetClosedWrt}(v, u)$ defined as follows: “if $\text{Status}(v)$ is of the form $\text{closed-wrt}(U)$ then $\text{Status}(v) := \text{closed-wrt}(U \cup \{u\})$, else $\text{Status}(v) := \text{closed-wrt}\{u\}$”. This procedure updates the status of $v$ to reflect that $v$ is closed w.r.t. $u$.

The Rules for Updating Statuses of Nodes:

\textbf{(UPS$_1$)} The first rule is as follows:

1. if one of the following conditions holds:
   (a) there exists $\alpha : \bot \in \text{Label}(v)$ or $\{\varphi, \overline{\varphi}\} \subseteq \text{FullLabel}(v)$,
   (b) there exists $a \neq a \in \text{Label}(v)$,
(c) \( \text{Type}(v) = \text{non-state}, a:(\leq n \text{s}.C) \in \text{Label}(v) \) and there are \( b_0, \ldots, b_n \in I \) such that, for all \( 0 \leq i, j \leq n \) with \( i \neq j \), \( \{s(a, b_i), b_i : C\} \subseteq \text{FullLabel}(v) \) and \( \{b_i \neq b_j, b_j \neq b_i\} \cap \text{Label}(v) \neq \emptyset \).

(d) \( \text{Status}(v) = \text{closed-wrt}(U) \) and \( v \in U \),

then \( \text{Status}(v) := \text{closed} \)

2. else if \( \text{Type}(v) = \text{state}, \text{Status}(v) = \text{f-expanded} \) and \( v \) has no successors then \( \text{Status}(v) := \text{open} \).

**Explanation 1** Informally, \text{closed} means “unsatisfiable w.r.t. \( \mathcal{R} \) and \( T \)”, \text{open} means “satisfiable w.r.t. \( \mathcal{R} \) and \( T \)”, and \text{closed-wrt}(U) means “unsatisfiable w.r.t. \( \mathcal{R} \), \( T \) and any node from \( U \)”. The above rule is thus intuitive. For a formal characterization of the statuses \text{closed} and \text{closed-wrt}(U), we refer the reader to Lemma A.4 (on page 27).

**Explanation 2** This rule deals with nominals. If \( u \) is a complex state and \( v \) is a simple node such that \( \text{Label}(v) \) contains a nominal \( \{a\} \) and \( v \) may affect the status of the root \( \nu \) via a path through \( u \), then, when considering \( u \) for constructing a model for the considered knowledge base, the simple node \( v \) should be merged with the named individual \( a \) in the complex node \( u \). If such a merging causes inconsistency then \( v \) is closed w.r.t. \( u \) and we update \( \text{Status}(v) \) accordingly.

**Explanation 3** This rule states that, if \( v \) is a predecessor of a node \( w \) then, whenever the status of \( w \) changes to \text{closed}, \text{closed-wrt}(\ldots) \) or \text{open}, the status of \( v \) should be updated (as soon as possible by using a priority queue of tasks). The update is done by one of the following subrules:

1. If \( \text{Type}(v) = \text{non-state} \) and \( \text{Status}(v) \notin \{\text{unexpanded}, \text{closed}, \text{open}\} \) then
   (a) if some successor of \( v \) received status \text{open} then \( \text{Status}(v) := \text{open} \)
   (b) else if all successors of \( v \) have status \text{closed} then \( \text{Status}(v) := \text{closed} \)
   (c) else if every successor of \( v \) has status \text{closed} or \text{closed-wrt}(\ldots) \) then:
      i. let \( w_1, \ldots, w_k \) be all the successors of \( v \) such that, for \( 1 \leq i \leq k \), \( \text{Status}(w_i) \) is of the form \text{closed-wrt}(U_i), and let \( U = \bigcap_{1 \leq i \leq k} U_i \)
      ii. for each \( u \in U \) do \( \text{SetClosedWrt}(v, u) \).

2. If \( \text{Type}(v) = \text{state} \), \( \text{Status}(v) \notin \{\text{unexpanded}, \text{closed}, \text{open}\} \) and a successor \( w \) of \( v \) received status \text{closed} then
   (a) if there exists \( e \in \text{ELabels}(v, w) \) such that \( \pi_T(e) = \text{testingClosedness} \)
       then \( \text{Status}(v) := \text{closed} \)
   (b) else
      – for each \( e \in \text{ELabels}(v, w) \) such that \( \pi_T(e) = \text{checkingFeasibility} \)
          add the constraint \( x_w,e = 0 \) to \( \text{ILConstraints}(v) \)
      – if \( \text{ILConstraints}(v) \) is infeasible then \( \text{Status}(v) := \text{closed} \).

3. If \( \text{Type}(v) = \text{state} \), \( \text{Status}(v) \notin \{\text{unexpanded}, \text{closed}, \text{open}\} \), a successor \( w \) of \( v \) received status \text{closed-wrt}(U) \), and \( v \) may affect the status of the root \( \nu \) via a path through \( u \in U \) then
   (a) if there exists \( e \in \text{ELabels}(v, w) \) such that \( \pi_T(e) = \text{testingClosedness} \)
       then \( \text{SetClosedWrt}(v, u) \).
The Unary Static Expansion Rules:

\[ \frac{\text{Explanation 3}}{\text{For simplicity of understanding, one can first consider the case without}} \]
\[ \text{nominals and statuses \textit{closed-wrt}(...). A non-state is like an \textit{"or"}-node, whose status is the}} \]
\[ \text{disjunction of the statuses of its successors, treating \textit{open} as \textit{true} and \textit{closed} as \textit{false}. A state}} \]
\[ \text{is more sophisticated than an \textit{and}-node. The status of a state \( v \) is different from \textit{closed} iff}} \]
\[ \text{the following conditions hold:} \]
\[ \text{– for all successors \( w \) of \( v \), if there exists \( e \in ELabels(v, w) \) with \( \pi_T(e) = \text{testingClosedness} \)} \]
\[ \text{then \( Status(w) \neq \text{closed} \),} \]
\[ \text{– \( ILConstraints(v) \cup \{x_{w,e} = 0 \mid \langle v, w, e \rangle \in E, \ 0 \neq open, \ e \in ELabels(v, w) \text{ and} \)} \]
\[ \pi_T(e) = \text{testingClosedness} \} \text{ is feasible} \]
\[ \text{SetClosedWrt}(v, u). \]

4. If
\[ Type(v) = \text{state} \land Status(v) = \text{f-expanded}, \]
\[ \text{– every successor \( w \) of \( v \) with some \( e \in ELabels(v, w) \) having \( \pi_T(e) = \text{testingClosedness} \)} \]
\[ \text{has status \textit{open}, and} \]
\[ ILConstraints(v) \cup \{x_{w,e} = 0 \mid \langle v, w, e \rangle \in E, \ Status(w) \neq \text{open}, e \in ELabels(v, w) \text{ and} \)} \]
\[ \pi_T(e) = \text{testingClosedness} \} \text{ is feasible} \]
\[ \text{then \( Status(v) := \text{open} \).} \]

The Unary Static Expansion Rules:

\[ \text{(US$_1$) If } Type(v) = \text{non-state and } Status(v) = \text{unexpanded then} \]
\[ 1. \text{let } X = RFmls(v) \cup \{\langle a : C \rangle \in Label(v) \mid C \text{ is of the form } D \sqcap D' \text{ or } D \sqsubseteq s.D \text{ or} \}
\[ \leq 0.s.D \} \cup \{a : \neg \{b \} \in Label(v)\} \]
\[ 2. \text{let } label = Label(v) \cup \{\langle a : D \rangle, \langle a : D' \rangle \mid a : (D \sqcap D') \in Label(v)\}
\[ \cup \{a : \forall s.D \mid a : \leq 0.s.D \in Label(v)\}
\[ \cup \{a : \forall r.D \mid a : \forall s.D \in Label(v) \text{ and } r \subseteq R s\}
\[ \cup \{s(a, b) \mid r(a, b) \in Label(v) \text{ and } r \subseteq R s\}
\[ \cup \{b : D \mid \{a : \forall r.D, r(a, b)\} \subseteq Label(v)\}
\[ \cup \{b : \forall r.D \mid \{a : \forall r.D, r(a, b)\} \subseteq Label(v) \text{ and } trans_R(r)\}
\[ \cup \{a \neq b \mid a : \neg \{b \} \in Label(v)\}
\[ \text{– } X \]
\[ 3. \text{if } label - Label(v) \neq \emptyset \text{ then} \]
\[ \text{(a) } \text{ConToSucc}(v, \text{non-state, SType(v), label, } X, \text{IndRep}(v), \text{null}) \]
\[ \text{(b) } Status(v) := \text{f-expanded}. \]

\[ \text{Explanation 4 This rule makes a necessary expansion for a non-state } v \text{ by connecting it to only one successor } w \text{ which is a copy of } w \text{ with intuitive changes like:} \]
The Rule for Dealing with Nominals:

- if \( \alpha : (D \cap D') \in \text{Label}(v) \) then \( \alpha : (D \cap D') \) in \( \text{Label}(w) \) is replaced by \( \alpha : D \) and \( \alpha : D' \) and we remember this by adding it to \( RFmls(w) \);
- if \( \{a : \forall r.D, r(a, b)\} \subseteq \text{Label}(v) \) then we add \( b : D \) to \( \text{Label}(w) \); and so on.

Note that \( \text{Label}(w) - (\text{Label}(u) \cup RFmls(v)) \neq \emptyset \). That is, \( w \) contains some \textquotedblleft new\textquotedblright\ formulas.

\((\text{US}_2)\) If \( \text{Status}(v) = \text{unexpanded} \) and \( \text{Label}(v) \) contains \( a : \{b\} \) then

1. let \( X \) be the set obtained from \( \text{Label}(v) - \{a : \{b\}\} \) by replacing every occurrence of \( b \) not in \( \text{\textit{\&}} \) expressions by \( \alpha \)
2. let \( Y \) be the set obtained from \( RFmls(v) \) by replacing every occurrence of \( b \) by \( \alpha \)
3. \( w := \text{ConToSucc}(v, \text{non-state, complex}, X \cup \{a : b, b = a\}, Y \cup \{a : \{a\}\}, \text{IndRepl}(v), \text{null}) \)
4. \( \text{IndRepl}(w)(b) := a \)
5. for each \( c \in I \), if \( \text{IndRepl}(v)(c) = b \) then \( \text{IndRepl}(w)(c) := a \).
6. \( \text{Status}(v) := f\text{-expanded} \).

Explanation 5 If \( v \) is an unexpanded complex node with \( \text{Label}(v) \) containing \( a : \{b\} \) then \( a \) and \( b \) should denote the same individual and we expand \( v \) by connecting it to only one successor \( w \) which is a copy of \( v \) with \( b \) replaced by \( \alpha \) in an appropriate way.

\((\text{US}_3)\) If \( \text{Type}(v) = \text{non-state} \) and \( \text{Status}(v) = \text{unexpanded} \) then

1. if \( \text{SType}(v) = \text{simple} \) then let \( X \) be the set of all concepts of the form \( \leq 1 r \{a\} \) that are relevant w.r.t. \( T \) and \( \text{Label}(v) \), else let \( X \) be the set of all formulas of the form \( a : \leq 1 r \{b\} \) that are relevant w.r.t. \( T \) and \( \text{Label}(v) \)
2. if \( X - \text{Label}(v) \neq \emptyset \) then
   \( \text{ConToSucc}(v, \text{non-state, complex}, \text{Label}(v) \cup X, RFmls(v), \text{IndRepl}(v), \text{null}) \).

Explanation 6 This rule deals with interaction between number restrictions and nominals. We want to guarantee that every nominal represents a singleton set and a named individual cannot be cloned.

The Rule for Dealing with Nominals:

\((\text{DN})\) If \( \text{SType}(v) = \text{simple} \), \( \text{Status}(v) \notin \{\text{closed, open}\} \) and \( \text{Label}(v) \) contains \( \{a\} \) then

1. for each complex state \( u \) such that \( v \) is not closed w.r.t. \( u \) (i.e., \( \text{Status}(v) \) is not of the form \( \text{closed-wrt}(U) \) with \( u \in U \) and \( v \) may affect the status of the root \( \nu \) via a path through \( u \), do
   a. let \( X = \{\varphi_1, \ldots, \varphi_k\} \) be the set obtained from \( \{a : C \mid C \in \text{Label}(v)\} \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) when \( \text{IndRepl}(u)(b) \) is defined
   b. if \( X \notin \text{FullLabel}(u) \) then:
      i. delete the edge \( \langle u_0, w \rangle \) from \( E \) and its labels from \( ELabels \)
      ii. \( \text{ConToSucc}(u_0, \text{non-state, complex, Label}(u_0) \cup X, RFmls(u_0), \text{IndRepl}(u_0), \text{null}) \)
      iii. for each \( 1 \leq i \leq k \) such that \( \varphi_i \notin \text{FullLabel}(u) \) do:
           \( \text{ConToSucc}(u_0, \text{non-state, complex, Label}(u_0) \cup \{\varphi_i\}, RFmls(u_0), \text{IndRepl}(u_0), \text{null}) \)
2. \( \text{Status}(v) := \text{blocked} \).

Explanation 7 If \( u \) is a complex state and \( v \) is a simple node such that \( \text{Label}(v) \) contains a nominal \( \{a\} \) and \( v \) may affect the status of the root \( \nu \) via a path through \( u \) then, when considering \( u \) for constructing a model for the considered knowledge base, the simple node \( v \) should be merged with the named individual \( a \) in the complex node \( u \). The case when such a merging causes inconsistency is dealt with by the rule (\( \text{UPS}_2 \)) (with a higher priority). Consider the other case. The set \( X = \{\varphi_1, \ldots, \varphi_k\} \) of assertions computed at the step [a] of the rule would be added to \( \text{Label}(u) \). However, we do not want to modify labels of nodes.
In the case when \( X \not\subseteq \text{FullLabel}(u) \), the label of \( u \) is “incomplete” and we re-expand every predecessor \( u_0 \) of \( u \) by deleting the edge \((u_0, u)\) and connecting \( u_0 \) to \( k + 1 \) successors, where the label of the successor number 0 extends \( \text{Label}(u_0) \) with \( \{ \varphi_1, \ldots, \varphi_k \} \) and the label of the successor number \( i \) (\( 1 \leq i \leq k \)) extends \( \text{Label}(u_0) \) with \( \overline{\varphi_i} \) (the negation of \( \varphi_i \) in NNF). This is like an on-demand cut.

The Non-unary Static Expansion Rule:

(NUS) If \( \text{Type}(v) = \text{non-state} \) and \( \text{Status}(v) = \text{unexpanded} \) then

1. if \( \alpha : (C \sqcup D) \in \text{Label}(v) \) and \( \{ \alpha : C, \alpha : D \} \cap \text{FullLabel}(v) = \emptyset \) then
   (a) let \( X = \text{Label}(v) - \{ \alpha : (C \sqcup D) \} \)
   (b) let \( Y = \text{RFmls}(v) \cup \{ \alpha : (C \sqcup D) \} \)
   (c) \( \text{ConToSucc}(v, \text{non-state}, \text{SType}(v), X \cup \{ \alpha : C \}, Y, \text{IndRepl}(v), \text{null}) \)
   (d) \( \text{ConToSucc}(v, \text{non-state}, \text{SType}(v), X \cup \{ \alpha : D \}, Y, \text{IndRepl}(v), \text{null}) \)
   (e) \( \text{Status}(v) := \text{f-expanded} \)

Explanation 8 This subrule deals with syntactic branching on \( \alpha : (C \sqcup D) \in \text{Label}(v) \).
We expand \( v \) by connecting it to two successors \( w_1 \) and \( w_2 \), whose label are the label of \( v \) with \( \alpha : (C \sqcup D) \) replaced by \( \alpha : C \) or \( \alpha : D \), respectively. The formula \( \alpha : (C \sqcup D) \) is put into both \( \text{RFmls}(w_1) \) and \( \text{RFmls}(w_2) \). The expansion is done only when both \( w_1 \) and \( w_2 \) have a larger \( \text{FullLabel} \) than \( v \).

2. else if \( \text{SType}(v) = \text{complex}, \, s(a, b) \in \text{Label}(v) \) and
   (a) \( \text{Label}(v) \) contains \( a : (\leq n \, s.C) \), or
   (b) \( \text{Label}(v) \) contains \( a : (\geq n \, s.C) \) or \( a : (\exists s.C) \), where \( s \) is a numeric role,
   and \( \{ b : C, b \overline{C} \} \cap \text{FullLabel}(v) = \emptyset \) then
   (a) let \( X = \text{Label}(v) \cup \{ b : C \} \) and \( X' = \text{Label}(v) \cup \{ b : \overline{C} \} \)
   (b) \( \text{ConToSucc}(v, \text{non-state}, \text{complex}, X, \text{RFmls}(v), \text{IndRepl}(v), \text{null}) \)
   (c) \( \text{ConToSucc}(v, \text{non-state}, \text{complex}, X', \text{RFmls}(v), \text{IndRepl}(v), \text{null}) \)
   (d) \( \text{Status}(v) := \text{f-expanded} \)

Explanation 9 This subrule deals with the case when there is a lack of information about \( b \) for deciding how to satisfy the number restrictions about \( a \). We want to have either \( b : C \) or \( b : \overline{C} \) in \( \text{FullLabel}(v) \). So, we expand \( v \) by semantic branching: we connect it to two successors, one with label \( \text{Label}(v) \cup \{ b : C \} \) and the other with label \( \text{Label}(v) \cup \{ b : \overline{C} \} \). The expansion is done only when both the successors have a larger \( \text{FullLabel} \) than \( v \).

3. else if \( \text{SType}(v) = \text{complex}, \{ a : (\leq n \, s.C), s(a, b), s(a, b'), b : C, b : \overline{C} \} \subseteq \text{FullLabel}(v), b \neq b' \) and \( \{ b \neq b', b : \overline{b} \} \cap \text{Label}(v) = \emptyset \) then
   (a) let \( X_1 = \text{Label}(v) \cup \{ b \neq b', b : \overline{b} \} \) and let \( X_2 \) be the set obtained from \( \text{Label}(v) \) by
      replacing every occurrence of \( b' \) not in \( = \) expressions by \( b \)
   (b) let \( Y \) be the set obtained from \( \text{RFmls}(v) \) by replacing every occurrence of \( b' \) by \( b \)
   (c) \( \text{ConToSucc}(v, \text{non-state}, \text{complex}, X_1, \text{RFmls}(v), \text{IndRepl}(v), \text{null}) \)
   (d) \( w := \text{ConToSucc}(v, \text{non-state}, \text{complex}, X_2 \cup \{ b \neq b', b : \overline{b} \}, Y, \text{IndRepl}(v), \text{null}) \)
   (e) \( \text{IndRepl}(w)(b') := b \)
   (f) for each \( c \in I \), if \( \text{IndRepl}(v)(c) = b' \) then \( \text{IndRepl}(w)(c) := b \)
   (g) \( \text{Status}(v) := \text{f-expanded} \)

Explanation 10 This subrule deals with the case when there is a lack of information about whether \( b \) and \( b' \) denote the same individual for deciding how to satisfy the number restrictions about \( a \). We expand \( v \) by semantic branching: either \( b \) and \( b' \) denote the same individual or they do not. Technically, we connect \( v \) to two successors with appropriate contents.

\[ ^{14} \text{Fix a linear order between individual names. Then we can also assume that } b \text{ is less than } b' \text{ in that order.} \]
4. else if $SType(v) = \text{complex}$, \(a : (\leq m \cdot r.C), r(a,b) \subseteq Label(v)\), $Label(v)$ contains $a : (\geq n \cdot s.D)$ or $a : \exists s.D$ with $s \subseteq R$ $r$, and \(\{s(a,b), \neg s(a,b)\} \cap Label(v) = \emptyset\) then
(a) let $X_1 = Label(v) \cup \{s(a,b)\}$ and $X_2 = Label(v) \cup \{\neg s(a,b)\}$
(b) $\text{ConToSucc}(v, \text{non-state, complex}, X_1, RFmls(v), \text{IndRepl}(v), \text{null})$
(c) $\text{ConToSucc}(v, \text{non-state, complex}, X_2, RFmls(v), \text{IndRepl}(v), \text{null})$
(d) $\text{Status}(v) := f\text{-expanded}$.

**Explanation 11** This subrule deals with the case when there is a lack of information for deciding how to satisfy the number restrictions about $a$. We want to decide whether $b$ is an $s$-successor of $a$ or not. So, we expand $v$ by semantic branching: we connect it to two successors, one with label containing $s(a,b)$ and the other with label containing $\neg s(a,b)$. The expansion is done only when both the successors have a larger label than $v$.

---

**The Forming-State Rule:**

(FS) If $Type(v) = \text{non-state}$ and $\text{Status}(v) = \text{unexpanded}$ then
1. if $SType(v) = \text{simple}$ then $Type(v) := \text{state}$ and $\text{ILConstraints}(v) := \emptyset$
2. else
   (a) set $X := Label(v)$
   (b) for each $a : (\leq n \cdot s.D) \in Label(v)$ do
      i. let $m = \sharp\{b \mid \{s(a,b), b : D\} \subseteq \text{FullLabel}(v)\}$
      ii. add $a : (\leq (n - m) \cdot s.D)$ to $X$
   (c) for each $(a:C) \in Label(v)$, where $C$ is $\geq n \cdot s.D$ or $\exists s.D$ and $s$ is a numeric role, do
      i. if $C = \exists s.D$ then let $n = 1$
      ii. let $m = \sharp\{b \mid \{s(a,b), b : D\} \subseteq \text{FullLabel}(v)\}$
      iii. if $n > m$ then add $a : (\geq (n - m) \cdot s.D)$ to $X$
   (d) $\text{ConToSucc}(v, \text{state, complex}, X, RFmls(v), \text{IndRepl}(v), \text{null})$
   (e) $\text{Status}(v) := f\text{-expanded}$.

**Explanation 12** When the rules (UPS), (US), (DN) and (NUS) are not applicable to the non-state $v$, we apply this forming-state rule to $v$. If $v$ is a complex node then we connect it to a complex state $w$. When computing contents for $w$ we put into $Label(w)$ the requirements from $Label(v)$ after an appropriate modification that takes into account the assertions in $Label(v)$ that represent the relationship between named individuals. For example, if $a : (\leq n \cdot s.D) \in Label(v)$ and there are $m$ pairwise different individuals $b_1, \ldots, b_m$ such that $\{s(a,b_i), b_i : D \mid 1 \leq i \leq m\} \subseteq \text{FullLabel}(v)$ then we add to $Label(w)$ the requirements $a : (\leq (n - m) \cdot s.D)$. Notice the use of $\leq$ instead of $\leq$. Note that, since the rule (NUS) is not applicable to $v$, we must have that $(b_i \neq b_j) \in Label(v)$ for any pair $i \neq j$, and for any individual $b$ such that $s(a,b) \in Label(v)$, either $b : D$ or $b : \overline{D}$ must belong to $\text{FullLabel}(v)$. When expanding $w$ we will not have to pay attention to the relationship between the individuals occurring in $Label(w)$.

If $v$ is a simple node then we just change $Type(v)$ to $\text{state}$ and initialize $\text{ILConstraints}(v)$ to $\emptyset$. Number restrictions about $v$ are dealt with later by the transitional full-expansion rule.

The way of forming a state for a complex node $v$ is more sophisticated (than for a simple node) because we may need to re-expand $v$ later due to nominals (as done by the rule (DN) for $u_0$).

---

**The Transitional Partial-Expansion Rule:**

(TP) If $Type(v) = \text{state}$ and $\text{Status}(v) = \text{unexpanded}$ then
1. for each $(a : \exists r.D) \in Label(v)$, where $r$ is a non-numeric role, do
The Transitional Full-Expansion Rule:

**Explanation 13** To realize a requirement \( \alpha : \exists r. D \) at a state \( v \), where \( r \) is a non-numeric role, we connect \( v \) to a new simple non-state \( w \) with appropriate contents as shown in the rule.

The Transitional Full-Expansion Rule:

- **(TF)** If \( \text{Type}(v) = \text{state} \) and \( \text{Status}(v) = \text{p-expanded} \) then
  1. if \( \text{SType}(v) = \text{complex} \) then let \( \Gamma = \text{Label}(v) \) else let \( \Gamma = \text{Label}(v) \cup \{ \leq n \ r. D \mid \leq n \ r. D \in \text{Label}(v) \} \cup \{ \geq n \ r. D \mid \geq n \ r. D \in \text{Label}(v) \} \cup \{ \geq 1 \ r. D \mid \exists r. D \in \text{Label}(v) \) and \( r \) is a numeric role\}
  2. \( E := \emptyset, E' := \emptyset \)
  3. for each \( (\alpha : \geq n \ r. D) \in \Gamma \) do
    a. for each \( (X, Y, \alpha) \in E \) do
      i. if \( r \in X \) and \( \{C, \overline{C}\} \cap Y = \emptyset \) then
        \[ E' := E' \cup \{(X, Y \cup \{C\}, \alpha) \cup (X, Y \cup \{\overline{C}\}, \alpha) \} \]
        (i.e., \( (X, Y, \alpha) \) is replaced by \( (X, Y \cup \{C\}, \alpha) \) and \( (X, Y \cup \{\overline{C}\}, \alpha) \))
      ii. else \( E' := E' \cup \{(X, Y, \alpha) \} \)
    b. \( E := E', E' := \emptyset \)
  4. repeat
    for each \( (\alpha : \leq n \ r. C) \in \Gamma \) do
      a. for each \( (X, Y, \alpha) \in E \) do
        i. if \( r \in X \) and \( \{C, \overline{C}\} \cap Y = \emptyset \) then
          \[ E' := E' \cup \{(X, Y \cup \{C\}, \alpha) \cup (X, Y \cup \{\overline{C}\}, \alpha) \} \]
          (i.e., \( (X, Y, \alpha) \) is replaced by \( (X, Y \cup \{C\}, \alpha) \) and \( (X, Y \cup \{\overline{C}\}, \alpha) \))
        ii. else \( E' := E' \cup \{(X, Y, \alpha) \} \)
      b. \( E := E', E' := \emptyset \)
    until no tuples were added to \( E \) during the last iteration
  5. for each \( (X, Y, \alpha) \in E \) do
    ConToSucc\( (v, \text{non-state}, \text{simple}, Y, \emptyset, \text{null}, (\text{checkingFeasibility}, X, \alpha)) \)
  6. for each \( (X, Y, \alpha) \in E \) do
    ConToSucc\( (v, \text{non-state}, \text{simple}, Y, \emptyset, \text{null}, (\text{checkingFeasibility}, X, \alpha)) \)
  7. \( \text{ILConstraints}(v) := \{ x_{w,e} \geq 0 \mid (v, w) \in E, e \in \text{ELabels}(v, w) \} \) and \( \pi_T(e) = \text{checkingFeasibility} \)
  8. for each \( (\alpha : C) \in \Gamma \) do
    a. if \( C \) is of the form \( \geq n \ r. D \) then add to \( \text{ILConstraints}(v) \) the constraint \( \sum \{ x_{w,e} \mid (v, w) \in E, e \in \text{ELabels}(v, w), \pi_T(e) = \text{checkingFeasibility}, r \in \pi_R(e), \pi_I(e) = \alpha, D \in \text{Label}(w) \} \geq n \)
    b. if \( C \) is of the form \( \leq n \ r. D \) then add to \( \text{ILConstraints}(v) \) the constraint \( \sum \{ x_{w,e} \mid (v, w) \in E, e \in \text{ELabels}(v, w), \pi_T(e) = \text{checkingFeasibility}, r \in \pi_R(e), \pi_I(e) = \alpha, D \in \text{Label}(w) \} \leq n \)
  9. \( \text{Status}(v) := \text{f-expanded} \).
Explaination 14 Let $\Gamma$ be the set computed at the step $[1]$. It consists of the requirements to be realized for $v$. To satisfy a requirement $\varphi = (\alpha : \exists r.C) \in \Gamma$ for $v$, one can first connect $v$ to a successor $w_\varphi$ using an edge label $e$ specified by the tuple $\langle X, Y, \alpha \rangle$ computed at the step $[8]$ where $\pi_T(e) = \text{checking Feasibility}$, $\pi_R(e) = X$, $\pi_I(e) = \alpha$ and $Y$ represents $\text{Label}(w_\varphi)$, and then clone $w_\varphi$ to create $n$ successors for $v$ (or only record the intention somehow). The label of $w_\varphi$ contains only formulas necessary for realizing the requirement $\alpha : \exists r.C$ and related ones of the form $\alpha : \forall r'.C'$ in $\Gamma$. To satisfy requirements of the form $\alpha : \exists n r.C'$ for $v$, where $r \subseteq_R r'$, we tend to use only copies of $w_\varphi$ extended with either $C'$ or $\overline{C}$ (for easy counting) as well as the mergers of such extended nodes. So, we first start with the set $\mathcal{E}$ constructed at the step $[3]$ which consists of tuples with information about successors to be created for $v$. We then modify $\mathcal{E}$ by taking necessary extensions of the nodes (see the step $[4]$). After that, we continue modifying $\mathcal{E}$ by adding to it also appropriate mergers of nodes and edges (see the step $[5]$). Successors for $v$ are created at the step $[6]$. The number of copies of a node $w$ that are intended to be used as successors of $v$ using an edge label $e$ is represented by a variable $x_{w,e}$ (we will not actually create such copies). The case when $w$ would be a named individual and cannot be cloned is dealt with by the rule $(US_3)$ (see Explanation $[3]$). The set $\mathcal{ILConstraints}(v)$ consisting of appropriate constraints about such variables are set at the steps $[7,8]$. 

4.5 Properties of $C_{SHOQ}$-Tableaux

Define the size of a knowledge base $KB = \langle R, T, A \rangle$ to be the number of bits used for the usual sequential representation of $KB$. It is greater than the number of symbols occurring in $KB$. If $N$ is the size of $KB$ and $\leq n r.C$ or $\geq n r.C$ is a number restriction occurring in $KB$ then:

- when numbers are coded in unary we have that $n \leq N$,
- when numbers are coded in binary we have that $n \leq 2^N$.

Lemma 4.6 (Complexity). Let $\langle R, T, A \rangle$ be a knowledge base in NNF of the logic $SHOQ$ and let $N$ be the size of $\langle R, T, A \rangle$. Then a $C_{SHOQ}$-tableau for $\langle R, T, A \rangle$ can be constructed in (at most) exponential time in $N$ in the following cases:

1. numbers are coded in unary,
2. numbers are coded in binary and $n \leq N$ for every concept $\leq n r.C$ occurring in $\langle R, T, A \rangle$,
3. numbers are coded in binary and $n \leq N$ for every concept $\geq n r.C$ occurring in $\langle R, T, A \rangle$.

Theorem 4.7 (Soundness and Completeness). Let $\langle R, T, A \rangle$ be a knowledge base in NNF of the logic $SHOQ$ and $G = \langle V, E, \nu \rangle$ be an arbitrary $C_{SHOQ}$-tableau for $\langle R, T, A \rangle$. Then $\langle R, T, A \rangle$ is satisfiable iff $\text{Status}(\nu) \neq \text{closed}$. 

See the Appendix for the proofs of the above lemma and theorem.

To check satisfiability of $\langle R, T, A \rangle$ one can construct a $C_{SHOQ}$-tableau for it, then return “no” when the root of the tableau has status closed, or “yes” in the other cases. We call this the $C_{SHOQ}$-tableau decision procedure. The corollary given below immediately follows from Theorem 4.7 and Lemma 4.6.

Corollary 4.8. The $C_{SHOQ}$-tableau decision procedure has $\text{ExpTime}$ complexity when numbers are coded in unary. 

5 Conclusions

Recall that $SHIQ$, $SHOQ$, $SHIO$ are the three most well-known expressive DLs with $\text{ExpTime}$ complexity. (Due to the interaction between $I$, $Q$ and $O$, the complexity of the DL $SHIO$ is $\text{NExpTime}$-complete). In this paper, we have presented the first $\text{ExpTime}$ tableau decision
procedure for checking satisfiability of a knowledge base in the DL $\textit{SHOQ}$ when numbers are coded in unary.

We applied Nguyen’s method \cite{18} of integer linear feasibility checking for dealing with number restrictions. This work differs from the work \cite{18} in that nominals are allowed instead of inverse roles. Without inverse roles, global caching is used instead of global state caching to allow more cache hits. We used special techniques for dealing with nominals and their interaction with number restrictions.

\textbf{Acknowledgments.} This work was supported by Polish National Science Centre (NCN) under Grants No. 2011/01/B/ST6/02759 (for the first author) and 2011/02/A/HS1/00395 (for the second author).

\textbf{References}

1. B. Dunin-Keplicz, L.A. Nguyen, and A. Szalas. Converse-PDL with regular inclusion axioms: A framework for MAS logics. \textit{J. Applied Non-Classical Logics}, 21(1):61–91, 2011.

2. J. Faddoul and V. Haarslev. Algebraic tableau reasoning for the description logic $\textit{SHOQ}$. \textit{J. Applied Logic}, 8(4):334–355, 2010.

3. N. Farsiniamarj. Combining integer programming and tableau-based reasoning: a hybrid calculus for the description logic $\textit{SHIQ}$. Master’s thesis, Concordia University, 2008.

4. R. Goré. Tableau methods for modal and temporal logics. In D’Agostino et al, editor, \textit{Handbook of Tableau Methods}, pages 297–396. Kluer, 1999.

5. R. Goré and L.A. Nguyen. A tableau system with automaton-labelled formulae for regular grammar logics. In B. Becket, editor, \textit{Proceedings of TABLEAUX 2005}, volume 3702 of \textit{LNAI}, pages 138–152. Springer, 2005.

6. R. Goré and L.A. Nguyen. ExpTime tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In N. Olivetti, editor, \textit{Proceedings of TABLEAUX 2007}, volume 4548 of \textit{LNAI}, pages 133–148. Springer, 2007.

7. R. Goré and L.A. Nguyen. Analytic cut-free tableaux for regular modal logics of agent beliefs. In F. Sadri and K. Satoh, editors, \textit{Proceedings of CLIMA VIII}, volume 5056 of \textit{LNAI}, pages 268–287. Springer, 2008.

8. R. Goré and L.A. Nguyen. Exptableu methods for $\textit{ALC}$ using sound global caching. \textit{J. Autom. Reasoning}, 50(4):355–381, 2013.

9. R. Goré and F. Widmann. Sound global state caching for $\textit{ALC}$ with inverse roles. In M. Giese and A. Waaler, editors, \textit{Proceedings of TABLEAUX 2009}, volume 5607 of \textit{LNCS}, pages 205–219. Springer, 2009.

10. R. Goré and F. Widmann. Optimal and cut-free tableaux for propositional dynamic logic with converse. In J. Giesl and R. Hähnle, editors, \textit{Proceedings of IJCAR 2010}, volume 6173 of \textit{LNCS}, pages 225–239. Springer, 2010.

11. J. Hladik and J. Model. Tableau systems for $\textit{SHIO}$ and $\textit{SHIQ}$. In \textit{Proceedings of Description Logics’2004}, volume 104 of \textit{CEUR Workshop Proceedings}, pages 168–177, 2004.

12. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\textit{SROIQ}$. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, \textit{Proc. of KR’2006}, pages 57–67. AAAI Press, 2006.

13. I. Horrocks and U. Sattler. Ontology reasoning in the $\textit{SHOQ(D)}$ description logic. In B. Nebel, editor, \textit{Proceedings of IJCAI2001}, pages 199–204. Morgan Kaufmann, 2001.

14. A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. \textit{Econometrica}, 28(3):497–520, 1960.

15. L.A. Nguyen. Analytic tableau systems and interpolation for the modal logics KB, KDB, K5, KD5. \textit{Studia Logica}, 69(1):41–57, 2001.

16. L.A. Nguyen. Cut-free ExpTime tableaux for checking satisfiability of a knowledge base in the description logic $\textit{ALCI}$. In \textit{Proceedings of ISMIS’2011}, volume 6834 of \textit{LNAI}, pages 465–475. Springer, 2011.

17. L.A. Nguyen. A cut-free ExpTime tableau decision procedure for the description logic $\textit{SHI}$. In \textit{Proceedings of ICCCI’2011 (1)}, volume 6922 of \textit{LNCS}, pages 572–581. Springer, 2011 (see also the long version \url{http://arxiv.org/abs/1106.2305}).

18. L.A. Nguyen. ExpTime tableaux for the description logic $\textit{SHIQ}$ based on global state caching and integer linear feasibility checking, arXiv:1205.5838, 2012.

19. L.A. Nguyen. A tableau method with optimal complexity for deciding the description logic $\textit{SHIQ}$. In \textit{Proceedings of ICCSAMA’2013}, volume 479 of \textit{Studies in Computational Intelligence}, pages 331–342. Springer, 2013.

20. L.A. Nguyen and J. Goliniska-Pilarek. An exptime tableau method for dealing with nominals and quantified number restrictions in deciding the description logic $\textit{SHOQ}$. In \textit{Proceedings of CSÉP’2013}.

21. L.A. Nguyen and A. Szalas. ExpTime tableaux for checking satisfiability of a knowledge base in the description logic $\textit{ALC}$. In N.T. Nguyen, R. Kowalczyk, and S.-M. Chen, editors, \textit{Proceedings of ICCCI’2009}, volume 5796 of \textit{LNAI}, pages 437–448. Springer, 2009.
22. L.A. Nguyen and A. Szalas. An optimal tableau decision procedure for Converse-PDL. In N.-T. Nguyen, T.-D. Bui, E. Szczerbiak, and N.-B. Nguyen, editors, Proceedings of KSE’2009, pages 207–214. IEEE Computer Society, 2009.
23. L.A. Nguyen and A. Szalas. Checking consistency of an ABox w.r.t. global assumptions in PDL. Fundamenta Informaticae, 102(1):97–113, 2010.
24. L.A. Nguyen and A. Szalas. Tableaux with global caching for checking satisfiability of a knowledge base in the description logic $SH$. $T$. Computational Collective Intelligence, 1:21–38, 2010.
25. L.A. Nguyen and A. Szalas. ExpTime tableau decision procedures for regular grammar logics with converse. Studia Logica, 98(3):387–428, 2011.
26. J.Z. Pan and I. Horrocks. Reasoning in the SHOQ($D_n$) description logic. In Proceedings of DL’2002, volume 53 of CEUR Workshop Proceedings, pages 53–62, 2002.
27. V.R. Pratt. A near-optimal method for reasoning about action. J. Comp. Syst. Sci., 20(2):231–254, 1980.
28. W. Rautenberg. Modal tableau calculi and interpolation. JPL, 12:403–423, 1983.
29. S. Tobies. Complexity results and practical algorithms for logics in knowledge representation. PhD thesis, RWTH-Aachen, 2001.
30. http://owl.cs.manchester.ac.uk/navigator/
A Appendix: Proofs

A.1 Complexity

Let $N$ be the size of $\langle R, T, A \rangle$. Define $\text{closure}(R, T, A)$ to be the smallest set $\Gamma$ of formulas such that:

1. all concepts (and subconcepts) used in $\langle R, T, A \rangle$ belong to $\Gamma$,
2. if $r$ is a role and $a$ is an individual used in $\langle R, T, A \rangle$ then $(\leq 1 r, \{a\}) \in \Gamma$,
3. if $\forall s.C \in \Gamma$ and $r \subseteq_R s$ then $\forall r.C \in \Gamma$,
4. if $\leq 0 s.C \in \Gamma$ then $\forall s.C \in \Gamma$,
5. if $C \in \Gamma$ and $C$ is not of the form $\leq n r.C$ or $\geq n r.C$ then $\overline{C} \in \Gamma$,
6. if $\exists r.C \in \Gamma$ and $r$ is a numeric role then $\geq 1 r.C \in \Gamma$,
7. if $\geq n r.C \in \Gamma$, $0 \leq m \leq N$ and $m < n$ then $\geq (n - m) r.C \in \Gamma$,
8. if $\leq n r.C \in \Gamma$, $0 \leq m \leq N$ and $m < n$ then $\leq (n - m) r.C \in \Gamma$,
9. all assertions of $A$ belong to $\Gamma$,
10. if $C \in \Gamma$ and $a$ is an individual used in $\langle R, T, A \rangle$ then $a : C \in \Gamma$,
11. if $a$ and $b$ are individuals used in $\langle R, T, A \rangle$ then $a \equiv b$ and $a \neq b$ belong to $\Gamma$,
12. if $r$ is a role and $a, b$ are individuals used in $\langle R, T, A \rangle$ then $r(a, b)$ and $\neg r(a, b)$ belong to $\Gamma$.

Lemma A.1. The number of formulas of $\text{closure}(R, T, A)$ is of rank $O(N^4)$, where $N$ is the size of $\langle R, T, A \rangle$.

Proof. The set $\Gamma = \text{closure}(R, T, A)$ can be constructed by initializing $\Gamma$ according to the items 1 and 9 and then repeatedly applying the rules stated in the remaining items of the list. After initialization, the set $\Gamma$ has $O(N)$ formulas. The rules in the items 2, 3 add $O(N^2)$ formulas to $\Gamma$. The rule in the item 10 adds $O(N^3)$ formulas to $\Gamma$ (as $\Gamma$ contains $O(N^2)$ concepts and there are $O(N)$ individual names). The rules in the items 11, 12 add $O(N^3)$ formulas to $\Gamma$. Thus, at the end, $\Gamma$ is of rank $O(N^4)$.\qed

We recall below Lemma 4.6 before presenting its proof.

Lemma 4.6. Let $\langle R, T, A \rangle$ be a knowledge base in NNF of the logic SHOQ and let $N$ be the size of $\langle R, T, A \rangle$. Then a $C_{SHOQ}$-tableau for $\langle R, T, A \rangle$ can be constructed in (at most) exponential time in $N$ in the following cases:

1. numbers are coded in unary,
2. numbers are coded in binary and $n \leq N$ for every concept $\leq n r.C$ occurring in $\langle R, T, A \rangle$,
3. numbers are coded in binary and $n \leq N$ for every concept $\geq n r.C$ occurring in $\langle R, T, A \rangle$.\qed

Proof. Let us construct an arbitrary $C_{SHOQ}$-tableau $G = \langle V, E, \nu \rangle$ for $\langle R, T, A \rangle$.

Let $N'$ be the number of formulas of $\text{closure}(R, T, A)$. We have $N' = O(N^3)$. For each $v \in V$, $\text{Label}(v) \subseteq \text{closure}(R, T, A)$. Since nodes of $G$ are globally cached, it follows that $G$ has no more than $2^N$ nodes.

Each state has no more than $O(2^N \cdot 2^{N'} \cdot N)$ outgoing edges (since each outgoing edge created by the transitional full-expansion rule is characterized by a tuple $\langle X, Y, \alpha \rangle$, where $X$ is a set of roles, $Y$ is a set of concepts, and $\alpha$ is null or an individual name).[15] Thus, checking feasibility of $ILConstraints(v)$ for a state $v$ is an IFDL($N, 2^N, 2^{N'} \cdot N$)-problem that satisfies the assumptions of Lemma 2.1 for the first case and satisfies the assumptions of Lemma 2.2 for the remaining two cases, and hence can be solved in (at most) exponential time in $N$.

Therefore, choosing a node to expand, checking whether a rule is applicable, and applying a rule can be done in (at most) exponential time in $N$. As each node is re-expanded at most once (by the rule (DN)), we conclude that the graph $G$ can be constructed in (at most) exponential time in $N$ for the considered cases.\qed

[15] The bound can be made tighter, e.g., using $O(N^2)$ instead of $N'$.
A.2 Soundness

Lemma A.2. Let \( G = \langle V, E, \nu \rangle \) be a \( C_{\text{SHOQ}} \)-tableau for \( \langle R, \mathcal{T}, A \rangle \). Then, for every \( v \in V \), \( \text{FullLabel}(v) \) is equivalent to \( \text{Label}(v) \). That is, for any interpretation \( I \):

- if \( v \) is a simple node then \( (\text{FullLabel}(v))^I = (\text{Label}(v))^I \),
- if \( v \) is a complex node then \( I \) satisfies \( \text{FullLabel}(v) \) iff it satisfies \( \text{Label}(v) \).

The proof of this lemma is straightforward.

Lemma A.3. Let \( G = \langle V, E, \nu \rangle \) be a \( C_{\text{SHOQ}} \)-tableau for \( \langle R, \mathcal{T}, A \rangle \). Then, for every \( v \in V \), if Type\( (v) = \) non-state and \( w_1, \ldots, w_k \) are all the successors of \( v \) then \( \text{FullLabel}(v) \) is satisfiable w.r.t. \( R \) and \( \mathcal{T} \) iff there exists \( 1 \leq i \leq k \) such that \( \text{FullLabel}(w_i) \) is satisfiable w.r.t. \( R \) and \( \mathcal{T} \).

The proof of this lemma is straightforward.

Let \( G = \langle V, E, \nu \rangle \) be a \( C_{\text{SHOQ}} \)-tableau. For each node \( v \) of \( G \) with \( \text{Status}(v) \in \{ \text{closed}, \text{open} \} \), let DSTimeStamp\( (v) \) be the moment at which \( \text{Status}(v) \) was changed to its final value (i.e., determined to be closed or open). DSTimeStamp stands for “determined-status time-stamp”. For each node \( v \) of \( G \) with \( \text{Status}(v) = \text{closed-wrt}(U) \) and for each \( u \in U \), let CSTimeStamp\( (v, u) \) be the first moment at which \( \text{Status}(v) \) was changed to some closed-wrt\( (U') \) with \( u \in U' \). CSTimeStamp stands for “closed-wrt-status time-stamp”. For each non-state \( v \) of \( G \), let ETimeStamp\( (v) \) be the moment at which \( v \) was expanded the last time.

Lemma A.4. Let \( G = \langle V, E, \nu \rangle \) be a \( C_{\text{SHOQ}} \)-tableau for \( \langle R, \mathcal{T}, A \rangle \). Then, for every \( v \in V \):

1. if \( \text{Status}(v) = \text{closed} \) then \( \text{FullLabel}(v) \) is unsatisfiable w.r.t. \( R \) and \( \mathcal{T} \),
2. if \( \text{Status}(v) = \text{closed-wrt}(U) \) and \( u \in U \) then there does not exist any model of \( \langle R, \mathcal{T}, \text{FullLabel}(u) \rangle \) that satisfies \( \text{FullLabel}(v) \).

Proof. We prove this lemma by induction on the above mentioned time-stamps.

Consider the first assertion of the lemma for the case when \( v \) is a simple state and \( \text{Status}(v) \) is changed to closed by the subrule \([2b]\) of (UPS\(_3\)) because ILConstraints\( (v) \) is infeasible. We prove the contrapositive: suppose \( I \) is a model of \( R \) and \( \mathcal{T} \), and \( y \in (\text{FullLabel}(v))^I \); we show that ILConstraints\( (v) \) is feasible. Without loss of generality, assume that \( I \equiv \) finitely-branching.\(^{17}\) Thus, the set \( Z = \{ z \in \Delta^I \mid \langle y, z \rangle \in r^I \text{ for some } r \in R \} \) is finite. We compute a solution \( S \) for ILConstraints\( (v) \) as follows.

- For each \( \langle v, w \rangle \in E \) and \( e \in \text{ELabels}(v, w) \) such that \( \pi_T(e) = \text{checkingFeasibility} \), set \( n_{w,e} := 0 \).
- For each \( z \in Z \) do:
  - let \( \langle w_1, e_1 \rangle, \ldots, \langle w_k, e_k \rangle \) be all the pairs such that, for each \( 1 \leq i \leq k \):
    - \( \langle v, w_i \rangle \in E \), \( e_i \in \text{ELabels}(v, w_i) \) and \( \pi_T(e_i) = \text{checkingFeasibility} \),
    - \( z \in (\text{FullLabel}(w_i))^I \),
    - \( \langle y, z \rangle \in r^I \) for all \( r \in \pi_R(e_i) \),
    - the pair \( \langle w_i, e_i \rangle \) is “maximal” in the sense that there does not exist any pair \( \langle w'_i, e'_i \rangle \neq \langle w_i, e_i \rangle \) such that
      - \( \langle v, w'_i \rangle \in E \), \( e'_i \in \text{ELabels}(v, w'_i) \) and \( \pi_T(e'_i) = \text{checkingFeasibility} \),
      - \( z \in (\text{FullLabel}(w'_i))^I \),
      - \( \langle y, z \rangle \in r^I \) for all \( r \in \pi_R(e'_i) \);
  - for each \( 1 \leq i \leq k \), set \( n_{w_i,e_i} := n_{w_i,e_i} + 1 \).
- \( S := \{ x_{w,e} = n_{w,e} \mid \langle v, w \rangle \in E, e \in \text{ELabels}(v, w) \text{ and } \pi_T(e) = \text{checkingFeasibility} \} \).

\(^{16}\) Each non-state may be re-expanded at most once (by the rule (DN)) and each state is expanded at most once.\(^{17}\) It is known that the DL \( \text{SHOQ} \) has the finitely-branching model property.
We prove that $S$ is a solution for $ILConstraints(v)$.

If a constraint $x_{w,e} = 0$ was added to $ILConstraints(v)$ because $w$ got status closed then, by the inductive assumption with $v$ replaced by $w$, we can conclude that $n_{w,e}$ was not increased at all and hence must be 0, which means that the constraint $x_{w,e} = 0$ is satisfied by the solution $S$.

Consider a constraint $\sum \{x_{w,e} | (v, w) \in E, e \in ELabels(v, w), \pi_T(e) = checkingFeasibility, r \in \pi_R(e), \pi_I(e) = null, D \in Label(w)\} \geq n$ of $ILConstraints(v)$ and the corresponding concept $\geq n r D$. By the assumptions about $v$ and $y$, it can be derived that $Z$ contains pairwise different $z_1, \ldots, z_n$ such that $(y, z_i) \in r^T$ and $z_i \in D^T$, for $1 \leq i \leq n$. Each $z_i$ makes $n_{w,e}$ increased by 1 for some pair $(w, e)$ such that $(v, w) \in E, e \in ELabels(v, w), \pi_T(e) = checkingFeasibility, r \in \pi_R(e)$ and $D \in FullLabel(w)$. Therefore, the considered constraint is satisfied by the solution $S$.

Consider a constraint $\sum \{x_{w,e} | (v, w) \in E, e \in ELabels(v, w), \pi_T(e) = checkingFeasibility, r \in \pi_R(e), \pi_I(e) = null, D \in Label(w)\} \leq n$ of $ILConstraints(v)$ and the corresponding concept $\leq n r D$. By the assumptions about $v$ and $y$, it can be derived that $Z$ contains no more than $n$ pairwise different elements $z_1, \ldots, z_n$ such that $(y, z_i) \in r^T$ and $z_i \in D^T$, for $1 \leq i \leq n$. For each $z_i$, there exists at most one pair $(w, e)$ such that $(v, w) \in E, e \in ELabels(v, w), \pi_T(e) = checkingFeasibility, r \in \pi_R(e)$, $D \in FullLabel(w)$ and the consideration of $z_i$ causes $n_{w,e}$ to be increased by 1. This is due to the “maximality” of $(w, e)$ and the nature of the transitional full-expansion rule. Therefore, the considered constraint is satisfied by the solution $S$.

Now, consider the second assertion of the lemma for the case when $v$ is a complex state and $Status(v)$ becomes closed-wrt($U$) with $u \in U$ because of the call of SetClosedWrt($v, u$) at the step 3b of the rule (UPS$_3$) due to infeasibility of the set $ILC = ILConstraints(v) \cup \{x_{w,e} = 0 | 1 \leq i \leq k\}$, where $(w_1, e_1), \ldots, (w_k, e_k)$ are the pairs mentioned at that step of (UPS$_3$). We prove the contrapositive: suppose $I$ is a model of $(\mathcal{R}, T, FullLabel(u))$ that satisfies $FullLabel(v)$; we showed that the mentioned set $ILC$ of constraints is feasible. Without loss of generality, assume that $I$ is finitely-branching. We compute a solution $S$ for $ILC$ as follows.

1. For each $(v, w) \in E$ and $e \in ELabels(v, w)$ such that $\pi_T(e) = checkingFeasibility$, set $n_{w,e} := 0$.

2. For each individual $a$ occurring in $Label(v)$ and each $z \in \Delta^T$ such that $(a^T, z) \in r^T$ for some $r \in \mathcal{R}$ do:
   (a) let $(w_1', e_1'), \ldots, (w_{k'}', e_{k'}')$ be all the pairs such that, for each $1 \leq i \leq k'$:
      i. $(v, w_i') \in E, e_i' \in ELabels(v, w_i'), \pi_T(e_i') = checkingFeasibility$ and $\pi_I(e_i') = a$,
      ii. $z \in (FullLabel(w_i'))^T$,
      iii. $(a^T, z) \in r^T$ for all $r \in \pi_R(e_i')$,
   (b) for each $1 \leq i \leq k'$ do
      i. if $z \neq b^T$ for all $b$ occurring in $Label(v)$ then $n_{w_i', e_i'} := n_{w_i', e_i'} + 1$;
      ii. else if $z = b^T$ for some $b$ occurring in $Label(v)$ and there exists $a : (\geq l s D) \in Label(v)$ such that $s \in \pi_R(e_i'), D \in FullLabel(w_i')$ and $s(a, b) \notin Label(v)$ then $n_{w_i', e_i'} := n_{w_i', e_i'} + 1$.

3. $S := \{x_{w,e} = n_{w,e} | (v, w) \in E, e \in ELabels(v, w) and \pi_T(e) = checkingFeasibility\}$.

We prove that $S$ is a solution for $ILC$.

If a constraint $x_{w,e} = 0$ was added to $ILConstraints(v)$ because $w$ got status closed then, by the inductive assumption with $v$ replaced by $w$, we can conclude that $n_{w,e}$ was not increased at all and hence must be 0, which means that the constraint $x_{w,e} = 0$ is satisfied by the solution $S$.

Consider a constraint $(x_{w_i, e_i} = 0) \in ILC$ with $1 \leq i \leq k$ (the pair $(w_i, e_i)$ was mentioned earlier). By the specification of $(w_i, e_i)$ (at the step 3b) of the rule (UPS$_3$), $Status(w_i)$ is of
the form closed-wrt($U_1$) with $u \in U_1$. By the inductive assumption with $v$ replaced by $w_i$, we can conclude that $n_{w_i,e_i}$ was not increased at all and hence must be 0, which means that the constraint $x_{w_i,e_i} = 0$ is satisfied by the solution $S$.

Consider a concept $(a : \geq n \cdot s \cdot D) \in Label(v)$ and the corresponding constraint $\sum \{ x_{v,w} | \langle v, w \rangle \in E, e \in Elabels(v, w), \pi_T(e) = \text{checkingFeasibility}, s \in \pi_R(e), \pi_I(e) = a, D \in Label(w) \} \geq n$ of $\text{ILConstraints}(v)$. Let $m = \sharp \{ b | \{ s(a, b), b : D \} \subseteq FullLabel(v) \}$. We have that either $(a : \geq (n + m) \cdot s \cdot D) \in Label(v) \lor n = 1, m = 0$ and $(a : \exists s \cdot D) \in Label(v)$. Since $I$ is a model of FullLabel(v), there exist pairwise different $z_1, \ldots, z_{n+m}$ such that $\langle a^T, z_i \rangle \in s^T$ and $z_i \in D^I$ for all $1 \leq i \leq n + m$. Note that, if $s(a, b) \in Label(v)$ then, by the subrule 2 of (NUS), either $b : D \in FullLabel(v)$ or $b : D \notin FullLabel(v)$. Since $z_i \in D^I$ and $I$ is a model of FullLabel(v), if $z_i = b^I$ then $b : D \notin FullLabel(v)$. Therefore, for every $1 \leq i \leq n + m$, if $z_i = b^I$ and $s(a, b) \in Label(v)$ then $b : D \notin FullLabel(v)$. Let $Z = \{ z_1, \ldots, z_{n+m} \} - \{ b^I | s(a, b) \in Label(v) \}$. We have that $\sharp Z = n$. Each $z$ from $Z$ makes $n_{w,e}$ increased by 1 for some pair $\langle w, e \rangle$ such that $\langle v, w \rangle \in E, e \in Elabels(v, w), \pi_T(e) = \text{checkingFeasibility}, \pi_I(e) = a, s \in \pi_R(e) \lor D \in Label(w)$.

Note that if $\langle v, w \rangle \in E, e \in Elabels(v, w), \pi_T(e) = \text{checkingFeasibility}, \pi_I(e) = a, r \in \pi_R(e)$ and $C \subseteq FullLabel(w)$ then $n_{w,e}$ is increased only due to some $z$ such that $\langle a^I, z \rangle \in r^I$ and $z \in C^I$. Due to the “maximality” of $\langle w, e \rangle$ and the nature of the transitional full-expansion rule, for such a $z$ there exists at most one pair $\langle v, w \rangle$ such that $\langle v, w \rangle \in E, e \in Elabels(v, w), \pi_T(e) = \text{checkingFeasibility}, \pi_I(e) = a, r \in \pi_R(e), C \subseteq FullLabel(w)$ and the consideration of $z$ causes $n_{w,e}$ to be increased by 1. Since $a^I \in \{ (\leq (n + m) \cdot r \cdot C) \} \cup \{ (\geq (n + m) \cdot r \cdot C) \}$, to prove that the considered constraint is satisfied by the solution $S$, it suffices to show that if $z \in Z_1$ causes $n_{w,e}$ to be increased by 1 at the step 2(bii) then $r \notin \pi_R(e'_1) \lor C \notin \pi_R(w'_1)$. Suppose the contrary. We have that:

\begin{align}
- \{ a : \leq (n + m) \cdot r \cdot C, r(a, b), b : C \} \subseteq \text{FullLabel}(v), \\
- \{ v, w \} \in E, e'_1 \in Elabels(v, w'), \pi_T(e'_1) = \text{checkingFeasibility} \land \pi_I(e'_1) = a, \\
- b^I \in (Label(w'_1))^I \land \langle a^I, b^I \rangle \in (r')^I \text{ for all } r' \in \pi_R(e'_1), \\
- a : \geq s \cdot D \in Label(v), \pi_I(e'_1), D \in Label(w'_1) \land s(a, b) \notin Label(v), \\
- r \in \pi_R(e'_1) \land C \subseteq Label(w'_1). 
\end{align}

Since both $s$ and $r$ belong to $\pi_R(e'_1)$ (by 4 and 5), there exist roles

\begin{align}
r_0 = r, r_1, \ldots, r_{h-1}, r_h = s \text{ and } s_1, \ldots, s_h \text{, all belonging to } \pi_R(e'_1)
\end{align}

such that, for every $1 \leq j \leq h$:

\begin{align}
- s_j \leq s \land s_j \leq r, \\
- Label(v) \text{ contains } a : \geq s_j, D'_j \text{ or } a : \geq s_j, D'_j \text{ for some } D'_j \in Label(w'_1) \land n_j > 0, \\
- if \ j < h then Label(v) \text{ contains } a : \leq s_j, C'_j \text{ for some } C'_j \in Label(w'_1) \land m_j.
\end{align}

Note that the subrule 3 of (NUS) was not applicable to $v$. Having 1, 6, 7 and 8, we derive that $s_1(a, b) \in Label(v) \lor \neg s_1(a, b) \in Label(v)$. Since 6 and 7, $(a^I, b^I) \in s^T$. Since $I$ is a model of FullLabel(v), it follows that $\neg s_1(a, b) \notin Label(v)$, and hence $s_1(a, b) \in Label(v)$. Since $s_1 \not\subseteq r_1$ (by 7), by the rule (US1), we also have that $r_1(a, b) \in Label(v)$. Analogously, using
also [9], for every \( j \) from 1 to \( h \), we can derive that \( s_j(a,b) \in \text{Label}(v) \) and \( r_j(a,b) \in \text{Label}(v) \). Since \( s = r_h \), it follows that \( s(a,b) \in \text{Label}(v) \), which contradicts [4]. This completes the induction step for the second assertion of the lemma for the case when \( v \) is a complex state and \( \text{Status}(v) \) becomes \text{closed-wrt}(U) with \( u \in U \) because of the call of \text{SetClosedWrt}(v,u) at the step 3b of the rule (UPS\(_3\)) due to infeasibility of \text{ILC}.

The induction steps for:

- the first assertion of the lemma for the case when \( v \) is a complex state and \( \text{Status}(v) \) is changed to \text{closed} by the subrule 2b of (UPS\(_3\)) because \text{ILConstraints}(v) is infeasible,
- the second assertion of the lemma for the case when \( v \) is a simple state and \( \text{Status}(v) \) becomes \text{closed-wrt}(U) with \( u \in U \) because of the call of \text{SetClosedWrt}(v,u) at the step 3b of the rule (UPS\(_3\)) due to infeasibility of the corresponding set of constraints

can be proved in a similar way as done above for the two dual cases.

The induction steps for the other cases (that correspond to the subrule 1 of (UPS\(_1\)), the rule (UPS\(_2\)) and the subrules 1b, 1c, 2a, 3a of (UPS\(_3\)) are straightforward. \( \square \)

**Corollary A.5 (Soundness of \( C_{\text{SHOQ}} \)).** If \( G = (V,E,\nu) \) is a \( C_{\text{SHOQ}} \)-tableau for \( \langle R,T,A \rangle \) and \( \text{Status}(v) = \text{closed} \) then \( \langle R,T,A \rangle \) is unsatisfiable.

This corollary directly follows from Lemma A.4.

### A.3 Completeness

We prove completeness of \( C_{\text{SHOQ}} \) via model graphs. The technique has been used for other logics (e.g., in [28,15,1]). A model graph is a tuple \( \langle \Delta, \Pi, C, E \rangle \), where:

- \( \Delta \) is a non-empty and finite set,
- \( \Pi \) is a mapping that associates each individual name with an element of \( \Delta \),
- \( C \) is a mapping that associates each element of \( \Delta \) with a set of concepts,
- \( E \) is a mapping that associates each role with a binary relation on \( \Delta \).

A model graph \( \langle \Delta, \Pi, C, E \rangle \) is consistent and \( R \)-saturated if every \( x \in \Delta \) satisfies:

\[
\text{(10)} \quad C(x) \text{ does not contain } \bot \text{ nor any pair } C, \overline{C}.
\]

\[
\text{(11)} \quad \text{if } \langle x, y \rangle \in E(r) \text{ and } r \sqsubseteq R s \text{ then } \langle x, y \rangle \in E(s).
\]

\[
\text{(12)} \quad \text{if } \{ a \} \subseteq C(x) \text{ then } \Pi(a) = x.
\]

\[
\text{(13)} \quad \text{if } C \cap D \subseteq C(x) \text{ then } \{ C, D \} \subseteq C(x).
\]

\[
\text{(14)} \quad \text{if } C \cup D \subseteq C(x) \text{ then } C \subseteq C(x) \text{ or } D \subseteq C(x).
\]

\[
\text{(15)} \quad \text{if } \forall s. C \subseteq C(x) \text{ and } r \sqsubseteq R s \text{ then } \forall r.C \subseteq C(x).
\]

\[
\text{(16)} \quad \text{if } \langle x, y \rangle \in E(r) \text{ and } \forall r.C \subseteq C(x) \text{ then } C \subseteq C(y).
\]

\[
\text{(17)} \quad \text{if } \langle x, y \rangle \in E(r), \text{trans}_{R}(r) \text{ and } \forall r.C \subseteq C(x) \text{ then } \forall r.C \subseteq C(y).
\]

\[
\text{(18)} \quad \text{if } \exists r.C \subseteq C(x) \text{ then } \exists y \in \Delta \text{ such that } \langle x, y \rangle \in E(r) \text{ and } C \subseteq C(y).
\]

\[
\text{(19)} \quad \text{if } (\geq n)r.C \subseteq C(x) \text{ then } \sharp\{ \langle x, y \rangle \in E(r) \mid C \subseteq C(y) \} \geq n
\]

\[
\text{(20)} \quad \text{if } (\leq n)r.C \subseteq C(x) \text{ then } \sharp\{ \langle x, y \rangle \in E(r) \mid C \subseteq C(y) \} \leq n
\]

\[
\text{(21)} \quad \text{if } (\leq n)r.C \subseteq C(x) \text{ and } \langle x, y \rangle \in E(r) \text{ then } C \subseteq C(y) \text{ or } \overline{C} \subseteq C(y).
\]

Given a model graph \( M = \langle \Delta, \Pi, C, E \rangle \), the \( R \)-model corresponding to \( M \) is the interpretation \( \mathcal{I} = \langle \Delta, M \rangle \) where:

- \( a^M = \Pi(a) \) for every individual name \( a \),

\(^{18}\) A consistent and \( R \)-saturated model graph is like a Hintikka structure.
\(-A^I = \{x \in \Delta \mid A \in C(x)\}\) for every concept name \(A\),
\[-r^I = E'(r)\text{ for every role name } r \in R,\text{ where } E'(r)\text{ for } r \in R\] are the smallest binary relations on \(\Delta\) such that:
\begin{itemize}
  \item \(E(r) \subseteq E'(r)\),
  \item if \(r \subseteq R\) then \(E'(r) \subseteq E'(s)\),
  \item if \(\text{trans}_R(r)\) then \(E'(r) \circ E'(r) \subseteq E'(r)\).
\end{itemize}

Note that the smallest binary relations mentioned above always exist: for each \(r \in R\), initialize \(E'(r)\) with \(E(r)\); then, while one of the above mentioned condition is not satisfied, extend the corresponding \(E'(r)\) minimally to satisfy the condition.

**Lemma A.6.** If \(\mathcal{T}\) is the \(R\)-model corresponding to a consistent \(R\)-saturated model graph \(\langle \Delta, I, C, E \rangle\), then \(\mathcal{T}\) is a model of \(R\) and, for every \(x \in \Delta\) and \(C \in C(x)\), we have that \(x \in C^I\).

The first assertion of this lemma clearly holds. The second assertion can be proved by induction on the structure of \(C\) in a straightforward way.

Let \(G = \langle V, E, \nu \rangle\) be a \(C_{\text{SHOQ}}\)-tableau for \(\langle R, T, A \rangle\).

Let \(v \in V\) be a complex non-state with \(\text{Status}(v) \neq \text{closed}\). A saturation path of \(v\) is a sequence \(v_0 = v, v_1, \ldots, v_k\) of nodes of \(G\), with \(k \geq 1\), such that \(\text{Type}(v_k) = \text{state}\) and
\begin{itemize}
  \item for every \(1 \leq i \leq k\), \(\text{Status}(v_i) \neq \text{closed}\)
  \item for every \(0 \leq i < k\), \(\text{Type}(v_i) = \text{non-state}\) and \((v_i, v_{i+1}) \in E\).
\end{itemize}
Observe that each saturation path of \(v\) is finite\(^{19}\). Furthermore, if \(v_i\) is a non-state with \(\text{Status}(v_i) \neq \text{closed}\) then \(v_i\) has a successor \(v_{i+1}\) with \(\text{Status}(v_{i+1}) \neq \text{closed}\). Therefore, \(v\) has at least one saturation path.

Let \(u \in V\) be a complex state and \(v \in V\) be a simple non-state such that \(\text{Status}(u) \neq \text{closed}\), \(\text{Status}(v) \neq \text{closed}\) and \(\text{Status}(v) \neq \text{closed-wrt}\{u, \ldots\}\) and \(v\) may affect the status of the root \(v\) via a path through \(u\). A saturation path of \(v\) w.r.t. \(u\) is a sequence \(v_0 = v, v_1, \ldots, v_k\) of nodes of \(G\), with \(k \geq 1\), such that either \(\text{Type}(v_k) = \text{state}\) or \(\text{Status}(v_k) = \text{blocked}\), and
\begin{itemize}
  \item for every \(1 \leq i \leq k\), \(\text{Status}(v_i)\) is not \(\text{closed}\) nor \(\text{closed-wrt}\{u, \ldots\}\),
  \item for every \(0 \leq i < k\), \(\text{Type}(v_i) = \text{non-state}\) and \((v_i, v_{i+1}) \in E\).
\end{itemize}
Observe that each saturation path of \(v\) w.r.t. \(u\) is finite (see the footnote \(^{19}\)). Furthermore, if \(v_i\) is a non-state with \(\text{Status}(v_i)\) different from \(\text{closed}\) and \(\text{closed-wrt}\{u, \ldots\}\), then \(v_i\) has a successor \(v_{i+1}\) with \(\text{Status}(v_{i+1})\) different from \(\text{closed}\) and \(\text{closed-wrt}\{u, \ldots\}\). Therefore, \(v\) has at least one saturation path w.r.t. \(u\).

**Lemma A.7 (Completeness of \(C_{\text{SHOQ}}\)).** Let \(G = \langle V, E, \nu \rangle\) be a \(C_{\text{SHOQ}}\)-tableau for \(\langle R, T, A \rangle\). Suppose \(\text{Status}(\nu) \neq \text{closed}\). Then \(\langle R, T, A \rangle\) is satisfiable.

**Proof.** The root \(\nu\) has a saturation path \(u_0, \ldots, u_k\) with \(u_0 = \nu\). Let \(u = u_k\). We define a model graph \(M = \langle \Delta, I, C, E \rangle\) as follows:

1. Let \(\Delta_0\) be the set of all individual names \(a\) such that \(\text{IndRepl}(u)(a) = a\). Set \(\Delta := \Delta_0\) and \(I := \text{IndRepl}(u)\). If \(a \in I\) does not occur in \(\langle R, T, A \rangle\) then define \(I(a)\) to be some individual occurring in \(\Delta_0\). For each \(a \in \Delta_0\), mark \(a\) as \(\text{unresolved}\)\(^{20}\) and set \(C(a) := \{C \mid (a : C) \in \text{FullLabel}(u)\}\). For each role \(r\), set \(E(r) := \{\langle a, b \rangle \mid r(a, b) \in \text{FullLabel}(u)\}\).

2. For every \(\text{unresolved}\) node \(y \in \Delta\) do:
   \begin{enumerate}
     \item (a) If \(y \in \Delta_0\) then let \(v = u\) and \(WE = \{(w, e) \mid (v, w) \in E, e \in ELabels(v, w)\}\) and \(\pi_I(e) = y\).
   \end{enumerate}

\(^{19}\) If a non-state \(v_{i+1}\) is a successor of a non-state \(v_i\) then either \(\pi[I(a : \text{IndRepl}(v_{i+1})(a) = a)] < \pi[I(a : \text{IndRepl}(v_i)(a) = a)]\) or \(\pi[I(a : \text{IndRepl}(v_i)(a) = a)] < \pi[I(a : \text{IndRepl}(v_{i+1})(a) = a)]\) and \(\text{FullLabel}(v_i) \cap \text{FullLabel}(v_{i+1})\). Recall also that \(\text{FullLabel}(v_{i+1})\) is a subset of \(\text{closure}(R, T, A)\).

\(^{20}\) Each node of \(M\) will be marked either as \(\text{unresolved}\) or as \(\text{resolved}\).
(b) Else:
   i. Let \( v = f(y) \).
      (* \( f \) is a constructed mapping that associates each node of \( M \) not belonging to \( \Delta_0 \) with a simple state of \( G \); as a maintained property of \( f \), \( \text{Status}(v) \neq \text{closed} \).
   ii. Let \( WE = \{ (w, e) \mid \langle v, w \rangle \in E, e \in ELabels(v, w) \} \).
   iii. If \( \text{Status}(v) = \text{closed-wrt}\{u, \ldots\} \) and \( \mathbb{C}(y) \) is the set obtained from \( \text{FullLabel}(v) \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) when \( \text{IndRepl}(u)(b) \) is defined. *)
   iv. Else let \( \mathbb{C}(y) \) be the set obtained from \( \text{FullLabel}(v) \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) when \( \text{IndRepl}(u)(b) \) is defined.
      \( \mathbb{C}(y) \) is feasible because \( \text{Status}(v) \neq \text{closed} \) and \( \text{Status}(v) \neq \text{closed-wrt}\{u, \ldots\} \). *)
   (c) Let \( ILC = ILCConstraints(v) \cup \{ x_{w,e} = 0 \mid (w, e) \in WE \} \), \( \pi_T(e) = \text{checkingFeasibility} \), \( \text{Status}(w) = \text{closed-wrt}\{u, \ldots\} \).
      (* \( ILC \) is feasible because \( \text{Status}(v) \neq \text{closed} \) and \( \text{Status}(v) \neq \text{closed-wrt}\{u, \ldots\} \). *)
   (d) Fix a solution of \( ILC \), and for each \( (w, e) \in WE \):
      i. if \( \pi_T(e) = \text{testingClosedness} \) then let \( n_{w,e} = 1 \),
      ii. else let \( n_{w,e} \) be the value of \( x_{w,e} \) in that solution.
   (e) Delete from \( WE \) all the pairs \( (w, e) \) with \( n_{w,e} = 0 \).
   (f) For each \( (w_0, e) \in WE \) do:
      i. Let \( w_0, \ldots, w_h \) be a saturation path of \( w_0 \) w.r.t. \( u \).
      ii. Let \( X \) be the set obtained from \( \text{FullLabel}(w_h) \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) when \( \text{IndRepl}(u)(b) \) is defined.
      iii. If \( \text{Status}(w_h) = \text{blocked} \) then:
         \( \text{Observe that that the specification of } X, \text{IndRepl}(u)(a) = a \text{ and } a \in \Delta_0 \). *)
         A. Let \( \{ a \} \) be an element of \( X \).
         \( \text{Observe that the mentioned properties of } f \) are maintained here. *)
         B. For each \( r \in \pi_R(e) \), add \( \langle y, a \rangle \) to \( E(r) \).
      iv. Else, for \( i := 1 \) to \( n_{w_0,e} \) do:
         A. Add a new element \( z \) to \( \Delta \) and mark \( z \) as \text{unresolved}.
         B. For each \( r \in \pi_R(e) \), add \( \langle y, z \rangle \) to \( E(r) \).
         C. Set \( C(z) := X \) and \( f(z) := w_h \).
         (* \( \text{Observe that the mentioned properties of } f \) are maintained here. *)
   (g) Mark \( y \) as \text{resolved}.

The defined model graph \( M \) may be infinite. It consists of a finite base created at the step 1 and disjoint trees created at the step 2 possibly with edges coming back directly to \( \Delta_0 \) (nodes of the base) due to nominals.

It is straightforward to prove that \( M \) is a consistent \( \mathcal{R} \)-saturated model graph. Observe that:

- For any individual name \( b \), if \( a = \text{IndRepl}(u)(b) \) then \( \text{IndRepl}(u)(a) = a \) and \( \mathbb{I}(a) = a \).
- If \( (a : C) \in \mathcal{A} \) then the concept obtained from \( C \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) (when \( \text{IndRepl}(u)(b) \) is defined) belongs to \( \mathbb{C}(a') \), where \( a' = \text{IndRepl}(u)(a) \).
- If \( r(a, b) \in \mathcal{A} \) then \( \langle a', b' \rangle \in E(r) \), where \( a' = \text{IndRepl}(u)(a) \) and \( b' = \text{IndRepl}(u)(b) \).
- If \( a \neq b \in \mathcal{A} \) then \( a' \neq b' \in \text{Label}(u) \), where \( a' = \text{IndRepl}(u)(a) \) and \( b' = \text{IndRepl}(u)(b) \).
- Since \( \text{Status}(u) \neq \text{closed} \), we have that \( a' \neq b' \).
- For every \( C \in \mathcal{T} \), the concept obtained from \( C \) by replacing every individual \( b \) by \( \text{IndRepl}(u)(b) \) (when \( \text{IndRepl}(u)(b) \) is defined) belongs to \( \mathbb{C}(x) \) for all \( x \in \Delta \).

Hence, by Lemma \[ A.6 \] the interpretation corresponding to \( M \) is a model of \( \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \). \( \square \)