Deciding $\mathcal{SHOQ}^{\sqcap}$ Knowledge Base Consistency using Alternating Automata

Birte Glimm[1] and Ian Horrocks[1] and Ulrike Sattler[2]

[1] University of Oxford, Oxford, OX1 3QD, UK
[2] The University of Manchester, Manchester, M13 9PL

Abstract. We introduce an automata-based method for deciding the consistency of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases. The presented algorithm decides knowledge base consistency in deterministic double exponential time for $\mathcal{SHOQ}^{\sqcap}$, but is in ExpTime if no role conjunctions occur in the input knowledge base. This shows that $\mathcal{SHOQ}$ is indeed ExpTime-complete, which was, to the best of our knowledge, always conjectured but never proved.

1 Introduction

In this paper, we introduce an automata-based method for deciding the consistency of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases (KBs). The use of role conjunction naturally arises, for example, in the context of conjunctive query answering when a query contains two role atoms $r(x, y)$ and $s(x, y)$ for r, s roles and x, y variables. The presented decision procedure can, for example, be used to check the consistency of extended knowledge bases as they arise from the query rewriting procedure for conjunctive queries in $\mathcal{SHOQ}$ [1]. We do not give proofs here, but refer interested readers to [1] for detailed proofs.

In the following section, we give some background information. In Section 3, we show how we can eliminate transitivity, which is non-trivial in the presence of nominals and role conjunctions. In Section 4, we define a suitable alternating automaton and show how we can transform models of the KB to trees that can be processed by the automaton.

2 Preliminaries

We use the DL $\mathcal{SHOQ}$, which extends the basic DL $\mathcal{ALC}$ with transitive roles, role hierarchies, nominals and qualified number restrictions [2, 3]. We further allow for role conjunctions in the place of role names in value and number restrictions, and denote the obtained DL with $\mathcal{SHOQ}^{\sqcap}$. In $\mathcal{SHOQ}^{\sqcap}$ one can, for example, build the concept $\forall(r_1 \sqcap \ldots \sqcap r_n).C$ that is interpreted as $\{d \in \Delta^{\mathcal{I}} \mid$ there is a $(d, d') \in {r_1}^{\mathcal{I}} \cap \ldots \cap {r_n}^{\mathcal{I}}$ with $d' \in C^{\mathcal{I}}\}$. In number restrictions, we allow, as usual, only simple roles to occur in role conjunctions.

Since, in the presence of nominals, the ABox can be internalised, we assume that a $\mathcal{SHOQ}^{\sqcap}$ knowledge base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{R})$ where $\mathcal{T}$ is a TBox and $\mathcal{R}$ is

a role hierarchy. We use $\mathsf{rol}(\mathcal{K})$ for the set of role names used in $\mathcal{K}$ and $\mathsf{nom}(\mathcal{K})$ for the set of individual names that occur in $\mathcal{K}$. We assume that $\mathsf{nom}(\mathcal{K})$ is non-empty. This is w.l.o.g. since we can always add an axiom $\{o\} \sqsubseteq \top$ to $\mathcal{T}$ for a fresh nominal o from the set of individual names N_I.

For a concept C, we use $\mathsf{nnf}(C)$ to denote the negation normal form of C. We define the *closure* $\mathsf{cl}(\mathcal{K})$ of $\mathcal{K}$ as the smallest set containing $\mathsf{nnf}(\neg C \sqcup D)$ if $C \sqsubseteq D \in \mathcal{T}$; D if D is a sub-concept of C and $C \in \mathsf{cl}(\mathcal{K})$; and $\mathsf{nnf}(\neg C)$ if $C \in \mathsf{cl}(\mathcal{K})$.

3 Eliminating Transitivity

Since automata cannot directly handle transitive roles, we first transform a $\mathcal{SHOQ}^{\sqcap}$ KB $\mathcal{K}$ into an equisatisfiable $\mathcal{ALCHOQ}^{\sqcap}$ KB $\mathsf{elimTrans}(\mathcal{K})$. In the presence of role conjunctions, it does not suffice to extend the standard encoding of transitivity [4, 5] in a trivial way. Such a naive extension would result in an $\mathcal{ALCHOQ}^{\sqcap}$ KB $\mathsf{elimTrans}(\mathcal{K})$ that is obtained from $\mathcal{K}$ by treating all transitive roles as non-transitive and by adding an axiom

$$\forall(r_1 \sqcap \ldots \sqcap r_n).C \sqsubseteq \forall(t_1 \sqcap \ldots \sqcap t_n).(\forall(t_1 \sqcap \ldots \sqcap t_n).C)$$

for each concept $\forall(r_1 \sqcap \ldots \sqcap r_n).C \in \mathsf{cl}(\mathcal{K})$ and roles $t_1, \ldots, t_n \in N_{tR}$ such that $t_i \sqsubseteq^{*}_{\mathcal{R}} r_i$ for each i with $1 \leq i \leq n$.

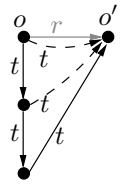

Fig. 1. A representation of a model for $\mathcal{K}$.

The following example shows that this encoding does not yield an equisatisfiable knowledge base. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a knowledge base with

$$\mathcal{T} = \{\{o\} \sqsubseteq \exists t.(\exists t.(\exists t.(\{o'\}))),\\ \{o\} \sqsubseteq \exists r.(\{o'\})\},$$

$\mathcal{R} = \emptyset$, and t a transitive role. Figure 1 shows a representation of a model for $\mathcal{K}$, where the grey edge represents the role r and the black edges represent the role t. The dashed black lines represent implicit (due to transitivity) instances of t. It is not hard to check that adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ makes the KB inconsistent.

The trivial encoding $\mathsf{elimTrans}(\mathcal{K})$ of $\mathcal{K}$ contains (among others) the additional axiom $\forall t.(\neg\{o'\}) \sqsubseteq \forall t.(\forall t.(\neg\{o'\}))$ since $\exists t.(\{o'\}) \in \mathsf{cl}(\mathcal{K})$ and, thus, $\forall t.(\neg\{o'\}) \in \mathsf{cl}(\mathcal{K})$. Adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ to $\mathcal{K}$ does not yield any further axioms in $\mathsf{elimTrans}(\mathcal{K})$ since r is a simple role. Since none of the added axioms explicates the implicit t relation between the nominals o and o', extending $\mathcal{K}$ with the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ yields a consistent knowledge base after applying the translation, contradicting our assumption that a knowledge base $\mathcal{K}$ is consistent iff $\mathsf{elimTrans}(\mathcal{K})$ is consistent.

Intuitively, this problem arises since we can have arbitrary relations between nominals. This can lead to situations where, as in the above example, we have an explicit relationship between two nominals, but only together with the implicit transitive shortcut can the universal quantifier over the role conjunction

be applied. Hence, the above described encoding does not suffice. We propose, therefore, a more involved encoding that explicates all transitive shortcuts between nominals.

Definition 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base. The function* $\mathsf{elimTrans}(\mathcal{K})$ *yields the $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base obtained from $\mathcal{K}$ as follows:*

1. *for each transitive role t and nominal $o \in \mathsf{nom}(\mathcal{K})$, add an axiom $\exists t.(\exists t.(\{o\})) \sqsubseteq \exists t.(\{o\})$,*
2. *for each concept $\forall R.C \in \mathsf{cl}(\mathcal{K})$ with $R = r_1 \sqcap \ldots \sqcap r_n$ and transitive roles $t_1, \ldots, t_n$ such that $t_i \sqsubseteq^*_{\mathcal{R}} r_i$ for each $1 \leq i \leq n$, add an axiom $\forall R.C \sqsubseteq \forall T.(\forall T.C)$, where $T = t_1 \sqcap \ldots \sqcap t_n$, and*
3. *all roles in* $\mathsf{elimTrans}(\mathcal{K})$ *are non-transitive.*

With the above definition, $\mathsf{elimTrans}(\mathcal{K})$ contains, additionally, the axiom $\exists t.(\exists t.(\{o\})) \sqsubseteq \exists t.(\{o\})$, which ensures that the implicit t-edges to nominals (the dashed lines in Figure 1) are made explicit. As a consequence, adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ indeed results in an inconsistent knowledge base.

Due to role conjunctions over non-simple roles, the encoding from Definition 1 yields not necessarily a knowledge base whose size is polynomial in the size of the input KB. The size of $\mathcal{K}$, denoted $|\mathcal{K}|$, is simply the number of symbols needed to write it over the alphabet of constructors, concept, role, and individual names that occur in $\mathcal{K}$. To the best of our knowledge, it is unknown if this blow-up can be avoided.

Lemma 1. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base with $|\mathcal{K}| = m$ and where the length of the longest role conjunction occurring in $\mathcal{K}$ is n. Then $\mathcal{K}$ is consistent iff* $\mathsf{elimTrans}(\mathcal{K})$ *is consistent and the size of* $\mathsf{elimTrans}(\mathcal{K})$ *is polynomial in m and exponential in n.*

The number of transitive sub-roles for a role r_i that occurs in a role conjunction is bounded by m. Hence, we can use up to m transitive sub-roles for each of the at most n role conjuncts in the second step of the encoding, which results in at most m^n additional axioms in $\mathsf{elimTrans}(\mathcal{K})$.

4 Deciding $\mathcal{ALCHOQ}^{\sqcap}$ Knowledge Base Consistency

In this section, we show how we can use alternating automata to decide the consistency of an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base. We assume w.l.o.g. that existential and universal restrictions in $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases are expressed using number restrictions.

4.1 Alternating Automata

In this section, we devise an alternating automaton that accepts exactly (abstractions) of models of $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases. Such automata have first been used in the context of modal logics [6] and have also been extended to

the hybrid μ-calculus [7] (with converse programs), i.e., for deciding the consistency of $\mathcal{ALCI\mathcal{O}}$ knowledge bases with a universal role and fixpoints. The latter approach, however, lacks support for qualified number restrictions and adding those would result in a logic that is no longer decidable in ExpTime [8]. Recently, alternating automata have also been used for answering regular path queries in $\mathcal{ALCQ\mathcal{I}}b_{reg}$, which are a generalisation of unions of conjunctive queries [9]. Both of the aforementioned approaches use two-way alternating automata that are ideally suited for logics that allow for inverse roles (converse programs in the μ-calculus). Since $\mathcal{SH\mathcal{O}Q}^{\sqcap}$ does not support inverse roles, we choose the slightly simpler standard (one-way) alternating automata, where we can only move downwards in the input tree.

Alternating automata have the power of making both universal and existential choices. Informally, this means that in the transition function, we can create copies of the automaton, send them to successor nodes, and require that either some (existential) or all (universal) of them are accepting. We use, as usual, positive Boolean formulae as defined below in the specification of the transition function.

Definition 2. *Let $\mathbb{N}^*$ be the set of all (finite) words over the alphabet $\mathbb{N}$. A* tree *T is a non-empty, prefix-closed subset of $\mathbb{N}^*$. The empty word ε is called the* root *of T. For $w, w' \in T$, we call w' a* successor *of w if $w' = w \cdot c$ for some $c \in \mathbb{N}$, where "$\cdot$" denotes concatenation and, for $c = 0$, we set $w \cdot c = w$. A* labelled tree *over an alphabet Σ is a pair $(T, \mathcal{L})$, where T is a tree and $\mathcal{L}\colon T \to \Sigma$ maps each node in T to an element of Σ.*

Let X be a set of atoms. The set $\mathcal{B}^+(X)$ of positive Boolean formulae *is built over atoms from X,* true*, and* false *using only the connectives $\wedge$ and $\vee$. Let $X^\top$ be a subset of X. We say that $X^\top$* satisfies *a formula $\phi \in \mathcal{B}^+(X)$ if assigning* true *to all atoms in $X^\top$ and false to all atoms in $X \setminus X^\top$ makes ϕ true.*

Let $[k] = \{0, 1, \ldots, k\}$. An alternating looping tree automaton *on k-ary Σ-labelled trees is a tuple $\mathbf{A} = (\Sigma, Q, \delta, q_0)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta\colon Q \times \Sigma \to \mathcal{B}^+([k] \times Q)$ is the transition function.*

A run *of $\mathbf{A}$ on a Σ-labelled k-ary tree $(T, \mathcal{L})$ is a $(T \times Q)$-labelled tree $(T_r, \mathcal{L}_r)$ that satisfies the following conditions:*

- *$\mathcal{L}_r(\varepsilon) = (\varepsilon, q_0)$,*
- *if $y \in T_r$ with $\mathcal{L}_r(y) = (x, q)$ and $\delta(q, \mathcal{L}(x)) = \phi$, then there is a (possibly empty) set $S \subseteq [k] \times Q$ that satisfies ϕ such that, for each $(c, q') \in S$, y has a successor $y \cdot i$ in T_r with $i \in \mathbb{N}$ and $\mathcal{L}_r(y \cdot i) = (x \cdot c, q')$.*

An automaton $\mathbf{A}$ accepts *an input tree T if there exists a run of $\mathbf{A}$ on T. The language accepted by $\mathbf{A}$, $\mathsf{lang}(\mathbf{A})$, is the set of all trees accepted by $\mathbf{A}$.*

For alternating automata, the *non-emptiness problem* is the following: given an alternating automaton $\mathbf{A}$, is there a tree $(T, \mathcal{L})$ such that $\mathbf{A}$ has an accepting run on $(T, \mathcal{L})$?

Please note that, since we use looping automata, we do not impose any acceptance conditions and each run is accepting, i.e., we require only that the

conditions imposed on a run are satisfied. Other existing automata based procedures for Description or Modal Logics use Büchi or parity acceptance conditions [6, 7, 9], usually because the logics allow for the transitive closure operator to be used on roles, which is not the case for $\mathcal{ALCHOQ}^{\sqcap}$. In the remainder we assume that $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ is an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base.

4.2 Canonical Models of Bounded Branching Degree

Automata rely on the tree/forest model property of the logic. We define, therefore, canonical models of a knowledge base as models that have a domain that consists of a collection of trees. In order to simplify the following definition we make the unique name assumption. This is w.l.o.g. as we can guess an appropriate partition among the individual names and replace the individual names in each partition with one representative individual name from that partition and we can also show that this does not affect the complexity.

Definition 3. *Given a set of elements $O = \{o_1, \ldots, o_n\}$, a* forest F w.r.t. O *is a subset of $O \times \mathbb{N}^*$ such that, for each $o_i \in O, (o_i, \varepsilon) \in F$ and the set $\{w \mid (o_i, w) \in F\}$ is a tree.*

A canonical model *for $\mathcal{K}$ is a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for $\mathcal{K}$ that satisfies the following conditions:*

F1 *$\Delta^{\mathcal{I}}$ is a forest w.r.t. $\mathsf{nom}(\mathcal{K})$;*
F2 *if $((o, w), (o', w')) \in r^{\mathcal{I}}$, then either*
   (a) $w' = \varepsilon$ or
   (b) $o = o'$ and w' is a successor of w;
F3 *for each $o \in \mathsf{nom}(\mathcal{K})$, $o^{\mathcal{I}} = (o, \varepsilon)$.*

Usually, automata work on trees of bounded branching degree, where the *branching degree* $d(w)$ of a node w in a tree T is the number of successors of w. If there is a k such that $d(w) \leq k$ for each $w \in T$, then we say that T has branching degree k. We can show that a consistent $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base has a canonical model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where, for each $o \in \mathsf{nom}(\mathcal{K})$, the branching degree of the tree $\{w \mid (o, w) \in \Delta^{\mathcal{I}}\}$ is bounded by some k that is polynomial in the size of $\mathcal{K}$ assuming unary coding of numbers. For such a canonical model $\mathcal{I}$, we say that $\mathcal{I}$ has branching degree k. We use this result when we introduce the abstractions of models that are accepted by our automata.

Lemma 2. *Let $|\mathcal{K}| = m, n_{max}$ the maximal number occurring in number restrictions, and $k = m \cdot n_{max}$. If $\mathcal{K}$ is consistent, then $\mathcal{K}$ has a canonical model $\mathcal{I}$ with branching degree k.*

4.3 Tree Relaxations

In this section, we show how we can obtain labelled k-ary trees from a canonical model for an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base such that these trees can be used as input for our automaton. Since the labelled trees that an automaton takes

as input cannot have labelled edges, we additionally store, in the label of a node, with which roles it is related to its predecessor. Unfortunately, this does not work for the nominal nodes since a nominal node can be the successor of arbitrary elements and does not necessarily have a unique predecessor. In a first step, we build, therefore, a *relaxation* for a canonical model where, for each relationship between a node and a nominal node, we create a dummy node that is a representative of the nominal node. The label of the representative node is the extension of the label for the nominal node with rep and the role names with which the node is related to the nominal. For a graphical illustration, let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ with

$$\begin{aligned}\mathcal{T} = \{\ & \{o\} \sqsubseteq \exists t.(C \sqcap \exists r.(\exists r.(D \sqcap \exists t.(\{o\})))) \\ & \{o'\} \sqsubseteq \exists s.\top \sqcap \exists s.(\{o\}) \\ \mathcal{R} = \{\ & r \sqsubseteq t'\}.\end{aligned}$$

Figure 2 shows a canonical model $\mathcal{I}$ for $\mathcal{K}$ and a relaxation for $\mathcal{K}$ built from $\mathcal{I}$. In a second step, we build labelled trees from a relaxation, which we call *tree relaxations*.

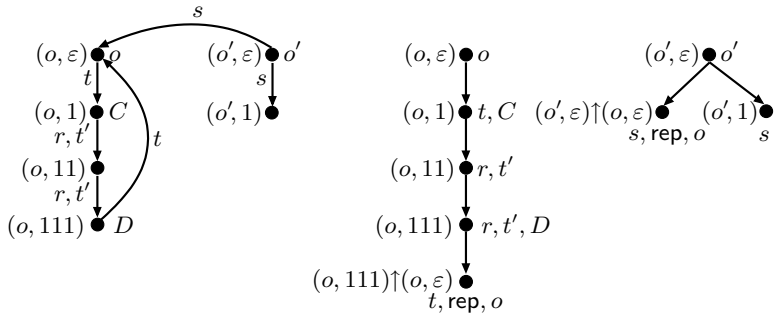

Fig. 2. A representation of a canonical model $\mathcal{I}$ for $\mathcal{K}$ (left) and its relaxation (right).

Definition 4. *A set $H \subseteq \mathsf{cl}(\mathcal{K})$ is called a* Hintikka set *for $\mathcal{K}$ if the following conditions are satisfied:*

1. *For each $C \sqsubseteq D \in \mathcal{T}$, $\mathsf{nnf}(\neg C \sqcup D) \in H$.*
2. *If $C \sqcap D \in H$, then $\{C, D\} \subseteq H$.*
3. *If $C \sqcup D \in H$, then $\{C, D\} \cap H \neq \emptyset$.*
4. *For all $C \in \mathsf{cl}(\mathcal{K})$, either $C \in H$ or $\mathsf{nnf}(\neg C) \in H$.*

We use $H(\mathcal{K})$ to denote the set of all Hintikka sets for $\mathcal{K}$.

A relaxation *$R = (\Delta^{\mathcal{I}}, \mathcal{L})$ for $\mathcal{K}$ with $\mathcal{L}\colon \Delta^{\mathcal{I}} \to 2^{\mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\mathsf{rep}\}}$ satisfies the following properties:*

(R1) *Let $D = \mathsf{nom}(\mathcal{K}) \times \mathbb{N}^*$ and $B = \{d{\uparrow}d' \mid d \in D$ and $d' \in \mathsf{nom}(\mathcal{K}) \times \{\varepsilon\}\}$, then $\Delta^{\mathcal{I}} \subseteq D \cup B$.*

(R2) *For each* $o \in \mathsf{nom}(\mathcal{K}), (o, \varepsilon) \in \Delta^{\mathcal{I}}$.
(R3) *Each set* $\{w \mid (o, w) \in \Delta^{\mathcal{I}} \cap D$ *is a tree*$\}$.
(R4) *If* $d{\uparrow}d' \in \Delta^{\mathcal{I}} \cap B$, *then* $\{d, d'\} \subseteq \Delta^{\mathcal{I}}, \mathcal{L}(d{\uparrow}d') \cap \mathsf{cl}(\mathcal{K}) = \mathcal{L}(d') \cap \mathsf{cl}(\mathcal{K})$, *and* $\mathsf{rep} \in \mathcal{L}(d{\uparrow}d')$.
(R5) *For each* $d \in \Delta^{\mathcal{I}}, \mathcal{L}(d) \cap \mathsf{cl}(\mathcal{K}) \in H(\mathcal{K})$.
(R6) *For each* $d \in \Delta^{\mathcal{I}}$, *if* $r \sqsubseteq s \in \mathcal{R}$ *and* $r \in \mathcal{L}(d)$, *then* $s \in \mathcal{L}(d)$.
(R7) *For each* $(o, \varepsilon) \in \Delta^{\mathcal{I}}, \mathcal{L}(o, \varepsilon) \cap \mathsf{rol}(\mathcal{K}) = \emptyset$.
(R8) *If* $d = (o, w) \in \Delta^{\mathcal{I}}$ *and* $(\geqslant n\ (r_1 \sqcap \ldots \sqcap r_k).C) \in \mathcal{L}(d)$, *then there are* n *distinct elements* $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$ *such that, for each* i *with* $1 \leq i \leq n, \{r_1, \ldots, r_k, C\} \subseteq \mathcal{L}(d_i)$ *and either* $d_i = (o, w \cdot c)$ *with* $c \in \mathbb{N}$ *or* $d_i = d{\uparrow}d' \in \Delta^{\mathcal{I}} \cap B$.
(R9) *If* $d = (o, w) \in \Delta^{\mathcal{I}}$ *and* $(\leqslant n\ (r_1 \sqcap \ldots \sqcap r_k).C) \in \mathcal{L}(d)$, *then* $\sharp(\{d' \in \Delta^{\mathcal{I}} \mid d' = (o, w \cdot c)$ *for some* $c \in \mathbb{N}$ *or* $d' = d{\uparrow}d_o \in \Delta^{\mathcal{I}} \cap B$ *and* $\{r_1, \ldots, r_k, C\} \subseteq \mathcal{L}(d')\}) \leq n$.

Lemma 3. *$\mathcal{K}$ has a relaxation iff $\mathcal{K}$ is consistent.*

In a second step, we build a so-called *tree relaxation* that is a labelled tree. For this, we additionally add a dummy root node labelled with root that has all nominal nodes as successors, and we require that the domain is a tree. A tree relaxation can, additionally, have dummy nodes labelled with # and we assume in the remainder that all tree relaxations are full trees, i.e., all non-leaf nodes have the same number of successors, and we add dummy nodes labelled with # where necessary.

Definition 5. *A* tree relaxation *for $\mathcal{K}$ is a labelled tree $(T, \mathcal{L})$ with $\mathcal{L}\colon T \to 2^{\mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\mathsf{rep}, \#, \mathsf{root}\}}$ that satisfies the following conditions:*

(T1) $\mathcal{L}(\varepsilon) = \{\mathsf{root}\}$ *and, for each* $w \in \mathbb{N}^+, \mathcal{L}(w) \cap \{\mathsf{root}\} = \emptyset$.
(T2) *For each* $o \in \mathsf{nom}(\mathcal{K})$, *there is a unique* $c \in \mathbb{N} \cap T$ *with* $o \in \mathcal{L}(c)$ *and* $\{\mathsf{rep}, \#, \mathsf{rol}(\mathcal{K})\} \cap \mathcal{L}(c) = \emptyset$.
(T3) *If* $c \in \mathbb{N} \cap T$ *and* $\mathsf{nom}(\mathcal{K}) \cap \mathcal{L}(c) = \emptyset$, *then* $\mathcal{L}(c) = \{\#\}$.
(T4) *For each* $w \in \mathbb{N}^+ \cap T, \sharp(\mathcal{L}(w) \cap \mathsf{nom}(\mathcal{K})) \leq 1$.
(T5) *For each* $w = w' \cdot c \in T$ *with* $w' \in \mathbb{N}^+$ *and* $c \in \mathbb{N}$, *if* $\mathcal{L}(w) \cap \mathsf{nom}(\mathcal{K}) \neq \emptyset$, *then* $\mathsf{rep} \in \mathcal{L}(w)$.
(T6) *For each* $w, w' \in T$ *and* $o \in \mathsf{nom}(\mathcal{K})$, *if* $o \in \mathcal{L}(w) \cap \mathcal{L}(w')$, *then* $\mathsf{cl}(\mathcal{K}) \cap \mathcal{L}(w) = \mathsf{cl}(\mathcal{K}) \cap \mathcal{L}(w')$.
(T7) *For each* $w \in \mathbb{N}^+ \cap T$, *if* $\{\mathsf{rep}, \#\} \cap \mathcal{L}(w) \neq \emptyset$, *then, for each successor* w' *of* w, $\# \in \mathcal{L}(w')$.
(T8) *For each* $w \in T$, *if* $\mathcal{L}(w) \cap \{\#, \mathsf{root}\} = \emptyset$, *then* $\mathcal{L}(w) \cap \mathsf{cl}(\mathcal{K}) \in H(\mathcal{K})$.
(T9) *For each* $w \in T$ *and* $r \sqsubseteq s \in \mathcal{R}$, *if* $r \in \mathcal{L}(w)$, *then* $s \in \mathcal{L}(w)$.
(T10) *For each* $w \in T$, *if* $(\geqslant n\ (r_1 \sqcap \ldots \sqcap r_m).C) \in \mathcal{L}(w)$, *then there are at least* n *distinct successors* $w_1, \ldots, w_n$ *of* w *with* $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, *for each* i *with* $1 \leq i \leq n$.
(T11) *For each* $w \in T$, *if* $(\leqslant n\ (r_1 \sqcap \ldots \sqcap r_m).C) \in \mathcal{L}(w)$, *then there are at most* n *distinct successors* $w_1, \ldots, w_n$ *of* w *with* $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, *for each* i *with* $1 \leq i \leq n$.

If T has branching degree k, then we say that $(T, \mathcal{L})$ is a k-ary tree relaxation for $\mathcal{K}$.

Lemma 4. *Let n_{max} be the maximal number occurring in a number restriction in $\mathcal{K}$, and $k = n_{max} \cdot |\mathcal{K}| + \sharp(\mathsf{nom}(\mathcal{K}))$. $\mathcal{K}$ has a k-ary tree relaxation iff $\mathcal{K}$ is consistent.*

4.4 Deciding Existence of Tree Relaxations

In order to decide consistency of an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base $\mathcal{K}$, it remains to devise a procedure that decides whether $\mathcal{K}$ has a tree relaxation. For this, we define an alternating automaton that accepts exactly the tree relaxations of $\mathcal{K}$. More precisely, we first define two alternating automata $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$, and then define an automaton $\mathcal{B}_{\mathcal{K}}$ as their intersection. Informally, the automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ just checks that the input tree has a structure as required whereas the automaton $\mathcal{A}_{\mathcal{K}}$ checks that the input is indeed a tree relaxation for $\mathcal{K}$. For alternating automata, intersection is simple: we introduce a new initial state q_0 and set the transition function for q_0 and each letter σ from the input alphabet Σ to $\delta(q_0, \sigma) = (0, q_{(0,1)}) \wedge (0, q_{(0,2)})$, where $q_{(0,1)}$ and $q_{(0,2)}$ are the initial states of $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$ respectively. The size of the resulting automaton is the sum of the sizes of $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$.

Let n_{max} be the maximal number occurring in number restrictions in $\mathcal{K}$, and $k = n_{max} \cdot |\mathsf{cl}(\mathcal{K})| + \sharp(\mathsf{nom}(\mathcal{K}))$. The alphabet Σ for both automata is

$$2^{\{\mathsf{rep},\#,\mathsf{root}\} \cup \mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \mathsf{nom}(\mathcal{K})}.$$

The automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ is defined as $(\Sigma, \{q_r, q_o, q_n, q_{\mathsf{rep}}, q_{\#}\}, \bar{\delta}, q_r)$ and rather than giving a precise definition of the transition function δ, we informally describe the objectives of the automaton and its states:

- We distinguish root (state q_r), nominal (state q_o), nominal representative (state q_{rep}), dummy (state $q_{\#}$), and normal nodes (state q_n).
- The label root is only found in the root node.
- The level one nodes are either "real" nominal nodes (i.e., they are not marked as representatives with rep) with exactly one nominal and no roles in their label, or they are dummy nodes labelled with # only.
- The level one nominal nodes have either normal, nominal representative, or dummy nodes as successors.
- Nominal representative nodes are marked with rep, and have exactly one nominal in their label.
- Dummy nodes have only dummy nodes as successors.

The automaton $\mathcal{A}_{\mathcal{K}}$ mainly checks the formulae occurring in the labels of the input. Hence, most of the states correspond to formulae in $\mathsf{cl}(\mathcal{K})$ and the transition function is more or less determined by the semantics. In the root node, we additionally make a non-deterministic choice, for each nominal and each atomic concept, whether the concept or its negation holds at the nominal

node. This choice is propagated downwards in the tree in order to ensure that the nominal representatives agree with their corresponding real nominal nodes on all atomic concepts. We propagate the concepts via a kind of universal role u and we assume that u is a symbol that does not occur in $\mathsf{cl}(\mathcal{K})$ or $\mathsf{rol}(\mathcal{K})$. We define the following set of auxiliary states, where N_C is the set of all concept names

$$Q_{\mathsf{rep}} = \{\neg\{o\} \sqcup C \mid o \in \mathsf{nom}(\mathcal{K}) \text{ and } C = A \text{ or } \neg A \text{ for } A \in N_C \cap \mathsf{cl}(\mathcal{K})\}.$$

We then define $\mathcal{A}_\mathcal{K}$ as (Σ, Q, δ, q_0), where q_0 is the initial state and the set Q of states is

$$\{q_0\} \cup \mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\neg r \mid r \in \mathsf{rol}(\mathcal{K})\} \cup \{q_\mathcal{T}, q_\mathcal{R}\} \cup Q_{\mathsf{rep}} \cup \{\forall u.C \mid C \in Q_{\mathsf{rep}}\} \cup \{\langle \bowtie nR.C, i, j\rangle \mid \bowtie \in \{\leqslant, \geqslant\}, \bowtie nR.C \in \mathsf{cl}(\mathcal{K}), \text{ and } 0 \leq i, j \leq k\},$$

States of the form $\langle \bowtie n\ R.C, i, j\rangle$ are used to check that the number restrictions are satisfied. We now give a definition for the transition function together with an explanation for each of the different types of states. For each $\sigma \in \Sigma$, the transition function δ is defined as follows:

At the root node, we are in the initial state q_0 which has the following tasks: (a) we make the non-deterministic guesses for all atomic concepts, (b) we check that there is exactly one nominal node for each of the nominals in $\mathsf{nom}(\mathcal{K})$, and (c) we make sure that the axioms in $\mathcal{T}$ and $\mathcal{R}$ are satisfied in all non-dummy descendants. Let $\ell = \sharp(\mathsf{nom}(\mathcal{K}))$.

$$\begin{aligned}\delta(q_0, \sigma) = \bigwedge_{A \in \sigma \cap N_C} \bigwedge_{i=1}^{\ell} ((0, \forall u.(\neg\{o_i\} \sqcup A)) \vee (0, \forall u.(\neg\{o_i\} \sqcup \neg A))) \wedge \\ \bigwedge_{i=1}^{\ell} \bigvee_{j=1}^{k} (j, \{o_i\}) \wedge \bigwedge_{1 \leq i < j \leq k} \left(\bigwedge_{o \in \mathsf{nom}(\mathcal{K})} (i, \neg\{o\}) \vee (j, \neg\{o\}) \right) \\ \bigwedge_{i=1}^{k} (i, q_\mathcal{T}) \wedge (i, q_\mathcal{R})\end{aligned}$$

Whenever we are in a state that is used to propagate information downwards through the whole tree via the "universal role" and we are not at a dummy node, we check that the required concept holds at the current node and also check all successors. More precisely, for each $C \in Q_{\mathsf{rep}}$,

$$\delta(\forall u.C, \sigma) = \begin{cases} (0, C) \wedge \bigwedge_{i=1}^{k} (i, \forall u.C) & \text{if } \#, \mathsf{root} \notin \sigma \\ \bigwedge_{i=1}^{k} (i, \forall u.C) & \text{if } \mathsf{root} \in \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

All non-dummy descendants of the root nodes must satisfy the TBox and RBox axioms. We give the definition for TBox Axioms here and the one for the RBox are analogously:

$$\delta(q_\mathcal{T}, \sigma) = \begin{cases} \bigwedge_{C \sqsubseteq D \in \mathcal{T}} ((0, \mathsf{nnf}(\neg C)) \vee (0, D)) \wedge \bigwedge_{i=1}^{k} (i, q_\mathcal{T}) & \text{if } \# \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

The concepts that are used as states are inductively decomposed according to the semantics. We start by defining the base cases. For each $\alpha \in (N_C \cap \mathsf{cl}(\mathcal{K})) \cup \mathsf{rol}(\mathcal{K}) \cup \mathsf{nom}(\mathcal{K})$, we set $\delta(\alpha, \sigma)$ to true if $\alpha \in \sigma$ and to false otherwise. Since we use constructors for nominals, we set, for each $o \in \mathsf{nom}(\mathcal{K})$, $\delta(\{o\}, \sigma) = (0, o)$. For negated names $\neg\alpha$ and nominal constructors we proceed just in the opposite way. Conjunction and disjunction are then handled in the straightforward way. For each $C_1 \sqcap C_2 \in \mathsf{cl}(\mathcal{K})$, we set $\delta(C_1 \sqcap C_2, \sigma) = (0, C_1) \wedge (0, C_2)$ and analogously for disjunction.

For number restrictions, we have to use a more sophisticated technique that involves states that count how many successors have been checked and how many of the checked ones fulfill the requirements of the number restriction. This technique was introduced by Calvanese et al. [10]. More precisely, for each concept of the form $(\geqslant n\ R.C) \in \mathsf{cl}(\mathcal{K})$ with $R = r_1 \sqcap \ldots \sqcap r_m$,

$$\delta(\geqslant n\ R.C, \sigma) = \begin{cases} (0, \langle \geqslant n\ R.C, 0, 0\rangle) & \text{if } \mathsf{rep} \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

For $1 \leq i \leq k$ and $1 \leq j \leq n$

$$\begin{aligned} \delta(\langle \geqslant n\ R.C, i, j\rangle, \sigma) = {} & (((i, \neg r_1) \vee \ldots \vee (i, \neg r_m) \vee (i, \mathsf{nnf}(\neg C))) \wedge \\ & (0, \langle \geqslant n\ R.C, i+1, j\rangle)) \vee \\ & ((i, r_1) \wedge \ldots \wedge (i, r_m) \wedge (i, C) \wedge \\ & (0, \langle \geqslant n\ R.C, i+1, j+1\rangle)) \end{aligned}$$

Then, for $1 \leq i \leq k$, $\delta(\langle \geqslant n\ R.C, i, n\rangle, \sigma) = \mathsf{true}$ and, for $1 \leq j < n$, $\delta(\langle \geqslant n\ R.C, k, j\rangle, \sigma) = \mathsf{false}$. Informally, we use the counter i to count how many of the k successors have already been checked and j is increased for each successor that fulfills the requirements of the number restriction. The atmost number restrictions are handled analogously.

We can now check whether the language accepted by the automaton $\mathcal{B}_\mathcal{K}$ is empty, which is enough to decide consistency of $\mathcal{K}$:

Theorem 1. *Let $\mathcal{B}_\mathcal{K}$ an alternating automaton as defined above. Then $\mathcal{K}$ is consistent iff the language accepted by $\mathcal{B}_\mathcal{K}$ is non-empty.*

4.5 Combined Complexity

Since looping alternating tree automata are a special case of alternating Büchi tree automata, we can use the result that, for an alternating Büchi automaton $\mathbf{A}$ with n states and input alphabet with ℓ elements, non-emptiness of the language accepted by $\mathbf{A}$ is decidable in time exponential in n and polynomial in ℓ [11]. By analysing the set of states and the size of the input alphabet of our automaton, we get the following result:

Theorem 2. *Checking the consistency of an $\mathcal{ALCHOQ}^\sqcap$ knowledge base $\mathcal{K}$ can be done in deterministic time single exponential in the size of $\mathcal{K}$ assuming unary coding of numbers in number restrictions.*

By Lemma 1, we obtain the following upper bound on deciding consistency of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases.

Theorem 3. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base where $|\mathcal{K}| = m$ and the length of the longest role conjunction occurring in $\mathcal{K}$ is n. Deciding the consistency of $\mathcal{K}$ can be done in deterministic time single exponential in m and double exponential in n assuming unary coding of numbers in number restrictions.*

The above result also shows that $\mathcal{SHOQ}$, i.e., when no role conjunctions are used, is indeed ExpTime-complete. This was always conjectured but, to the best of our knowledge, never proved. It remains an open question whether the exponential blow-up in the presence of role conjunctions can be avoided.

References

1. Glimm, B.: Querying Description Logic Knowledge Bases. PhD thesis, The University of Manchester, Manchester, United Kingdom (2007)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook. Cambridge University Press (2003)
3. Horrocks, I., Sattler, U.: Ontology reasoning in the $\mathcal{SHOQ}$(d) description logic. In Nebel, B., ed.: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), Morgan Kaufmann, Los Altos (2001) 199–204
4. Kazakov, Y., Motik, B.: A resolution-based decision procedure for $\mathcal{SHOIQ}$. In Furbach, U., Harrison, J., Shankar, N., eds.: Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006). Volume 4130., Seattle, WA, USA, Springer-Verlag (2006) 662–667
5. Hustadt, U., Motik, B., Sattler, U.: Reducing $\mathcal{SHIQ}^{-}$ Description Logic to Disjunctive Datalog Programs. In Dubois, D., Welty, C.A., Williams, M.A., eds.: Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004), Whistler, Canada, AAAI Press/The MIT Press (2004) 152–162
6. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Proceedings of the 25th International Colloquium on Automata, Languages, and Programming, London, UK, Springer-Verlag (1998) 628–641
7. Sattler, U., Vardi, M.Y.: The hybrid μ-calculus. In: Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001), London, UK, Springer-Verlag (2001) 76–91
8. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, RWTH Aachen (2001)
9. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proceedings of the 22th National Conference on Artificial Intelligence (AAAI 2007). (2007)
10. Calvanese, D., De Giacomo, G., Lenzerini, M.: 2ATAs make DLs easy. In: Proceedings of the 2002 Description Logic Workshop (DL 2002). Volume 53., CEUR (`http://ceur-ws.org/`) (2002)
11. Vardi, M.Y.: Alternating automata and program verification. In van Leeuwen, J., ed.: Computer Science Today. Volume 1000 of Lecture Notes in Computer Science. Springer-Verlag (1995) 471–485