A Tableau System for the Description Logic $\mathcal{SHIO}$

Jan Hladik
hladik@tcs.inf.tu-dresden.de

Chair for Automata Theory
University of Technology, Dresden

1 Introduction

For the satisfiability test of description logics (DLs) [1], there are two prominent families of algorithms with complementary advantages and disadvantages: firstly, *automata-based algorithms* (see e.g. [6]), which translate a DL expression φ into an automaton A_φ which accepts all (abstractions of) models for φ, so that the satisfiability test for φ can be reduced to the emptiness test of A_φ; and secondly *tableau algorithms* [2], which incrementally create a tree-shaped (pre-) model for φ using a set of *rules* labelling each tree node with the appropriate subformulas of φ. In short, the advantages of automata algorithms are on the theoretical side, because in many cases the proofs are very elegant and offer tight complexity bounds (in particular for ExpTime-complete logics), whereas the advantages of tableau algorithms are on the practical side, since they are well suited for implementation and optimisation. In contrast, intuitive automata algorithms usually require exponential time also in the best case, and thus an efficient implementation requires considerable modifications.

For this reason, an approach combining both advantages is highly desirable. In [3], we introduced *tableau systems (TS)*, a framework for tableau algorithms. If a tableau algorithm can be described within this framework, the existence of an automata algorithm (and thus the ExpTime upper bound) directly follows. Moreover, only soundness and completeness of the tableau algorithm have to be proved; termination and a practical procedure to ensure termination, called a *blocking test*, also follow from the prerequisites of the framework.

As an example for the usefulness of this framework, we present in this paper a TS for $\mathcal{SHIO}$, an expressive DL whose computational properties to the best of our knowledge have not been analysed so far. The purpose of this is two-fold: firstly, we obtain that $\mathcal{SHIO}$ satisfiability is ExpTime-complete and can be decided by a tableau algorithm. Secondly, the succinctness of the proofs demonstrates how our framework simplifies the design of tableau algorithms.

2 The Tableau Framework for ExpTime Logics

In DL tableau algorithms, a (pre-)model is represented by a tree with node and edge labels, called a *completion tree*.[1] The *completion rules* describe how to sat-

[1] This is different from first-order tableaus, where a model corresponds to a *path* within the generated tree.

isfy a formula contained in a node label, essentially by describing a subtree of bounded width and depth in which a particular rule can be applied, and the modification of this subtree through application of this rule. It is possible that several rules are applicable to a node at the same time, but the sequence of rule applications is *don't-care*-nondeterministic, i.e. every sequence will lead to the same result. In contrast, some rules, e.g. for disjunction, are *don't-know*-nondeterministic, i.e. it is possible that one disjunct may lead to a model, while another one may not. Unsatisfiability of the generated completion tree is detected through *clash triggers*, subtrees containing an obvious contradiction, thus a completion tree containing a clash trigger cannot be transformed into a model.

Due to space restrictions, we cannot present all details of the tableau framework; instead, we will describe the intuition which motivated the design and refer the reader to [3] for details. A completion tree is formalised as an *S-tree* $((V, E, n, \ell), \mu)$, where (V, E) is a bounded width tree and n, ℓ are node and edge labelling functions; and we capture the rules by using *S-patterns*, which are essentially S-trees of bounded depth. Thus for every possible S-pattern, we describe the possible modifications by all rules: a pattern P is mapped to a set of sets of patterns $\{S_1, S_2, \ldots, S_n\}$. The choice of the set S_i represents the don't-care choice of the rule that is applied, whereas the choice of the pattern within the set S_i represents the don't-know choice of the alternative (which means that a deterministic rule corresponds to a singleton set S_i). In particular, if no rule is applicable to P, then P is mapped to the empty set. As an example, consider a pattern P in which the node n is labelled with $\{C \sqcap D, E \sqcup F\}$. Here, one can define the rules as follows: $P \mapsto \{\{P_1\}, \{P_2, P_3\}\}$, and in P_1, C and D are added to the label of n, whereas E is added in P_2 and F in P_3. Patterns are also used to describe clash triggers: within our framework, a clash trigger is simply a pattern which contains a contradiction (in the context of the logic under consideration).

Since our intention was to define one tableau system for one logic rather than a particular one for every possible input, we have to include all possible node labels in these rules and clash triggers. For a particular input Γ, these sets are mapped to the appropriate subsets, which is necessary to ensure the ExpTime upper bound. Finally, we also wanted to allow for rules which are not applied locally, but globally, which is necessary e.g. for handling TBoxes or nominals. For this purpose, S-trees and -patterns also contain a *global memory* μ, which can be read and modified by a rule just like a node label.

Formally, a tableau system S is a tuple $(\mathsf{NLE}, \mathsf{GME}, \mathsf{EL}, \cdot^S, k, \mathcal{R}, \mathcal{C})$, where NLE is the set of all possible node label elements, GME is the set of global memory elements, EL is the set of edge labels, k is the maximum pattern depth, i.e. the number of edges on a longest path, and $\mathcal{R}$ and $\mathcal{C}$ are the sets of rules and clash triggers as defined above. The function $\cdot^S$ maps an input Γ to the tuple $(\mathsf{nle}, \mathsf{gme}, \mathsf{el}, \mathsf{ini})$, where nle, gme and el are finite subsets of the corresponding sets in S, and $\mathsf{ini} \subseteq \wp(\mathsf{nle}) \times \wp(\mathsf{gme})$ describes the possible initial states for Γ ($\wp$ denotes power set).

3 The Description Logic $\mathcal{SHIO}$

The DL $\mathcal{SHIO}$ extends the basic DL $\mathcal{ALC}$ [8] by transitive roles, role hierarchies, inverse roles, and *nominals*, i.e. concepts that have to be interpreted by singleton sets. The presence of both transitive roles together with role hierarchies allows for the internalisation of general concept inclusion axioms [5]; and nominals together with inverse roles require additional measures to ensure that nominals are handled correctly; e.g. we cannot use a forest model as for $\mathcal{SHOQ}$ (D) [4].

Definition 1 ($\mathcal{SHIO}$ syntax and semantics). Let CON be a set of concept names, ROL be a set of role names, the set of nominal names $\mathsf{NOM} \subseteq \mathsf{CON}$, and the set of transitive role names $\mathsf{TRA} \subseteq \mathsf{ROL}$. If r is a role name, then both r and r^-, the inverse of r, are roles. The set of $\mathcal{SHIO}$ concepts is inductively defined as follows: every concept name is a concept; and if C and D are concepts and r is a role, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall r.C$, and $\exists r.C$ are also concepts. If r and s are roles, then $r \sqsubseteq s$ is a *role inclusion axiom*. An *RBox* is a finite set of role inclusion axioms.

An *interpretation* of a concept C w.r.t. an RBox B is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and $\cdot^{\mathcal{I}}$ maps every concept name C to a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name r to a set $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For all $O \in \mathsf{NOM}$, it holds that $\#O^{\mathcal{I}} = 1$, where $\#S$ denotes the cardinality of a set S. For all $t \in \mathsf{TRA}$, it holds that $t^{\mathcal{I}} = (t^{\mathcal{I}})^+$, where $\cdot^+$ denotes the transitive closure of a relation. Complex roles and concepts are interpreted as follows:

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $\quad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $\quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
- $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{there is an } e \in \Delta^{\mathcal{I}} \text{ with } (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\}$
- $(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{for all } e \in \Delta^{\mathcal{I}}, \text{ if } (d,e) \in r^{\mathcal{I}}, \text{ then } e \in C^{\mathcal{I}}\}$
- $(r^-)^{\mathcal{I}} = \{(x,y) \mid (y,x) \in r^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ is a *model* for an RBox B if, for all $r \sqsubseteq s \in B$, it holds that $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. A *model* for C w.r.t. B is a model for B where $C^{\mathcal{I}}$ is a nonempty set. If a model exists, we say that C is *satisfiable* w.r.t. B.

4 A Tableau System for $\mathcal{SHIO}$

Before defining the TS $S_{\mathcal{SHIO}}$, we fix some notation. Firstly, to avoid multiple inverse operators as in r^{--}, we use the notation $\overline{r}$, with the meaning r^-, if r is a role name, and s, if r is an inverse role s^-. Secondly, in an S-pattern $P = ((V, E, n, \ell), \mu)$ with $\{m, n\} \subseteq V$, we call m an *r-neighbour* of n if $\ell(n,m) = r$ or $\ell(m,n) = \overline{r}$. Thirdly, for an RBox B, we define the *role hierarchy* B^+ as $B^+ = B \cup \{\overline{r} \sqsubseteq \overline{s} \mid r \sqsubseteq s \in B\}$, and by $\overset{*}{\sqsubseteq}_B$ we denote the reflexive-transitive closure of $\sqsubseteq$ on B^+. Finally, to capture roles which are implicitly declared to be transitive (e.g. $\overline{r}$ if $r \in \mathsf{TRA}$), we use, for an RBox B, the predicate Trans_B: for a role r, $\mathsf{Trans}_B(r)$ is true iff there exists a role s such that $s \in \mathsf{TRA}$, $s' \overset{*}{\sqsubseteq}_B r$ and $r \overset{*}{\sqsubseteq}_B s''$ for some $s', s'' \in \{s, s^-\}$.

The *closure* $\mathsf{clos}(C,B)$ of a $\mathcal{SHIO}$ concept term C and an RBox B is defined as follows: $C \in \mathsf{clos}(C,B)$; if $\neg D \in \mathsf{clos}(C,B)$, then $D \in \mathsf{clos}(C,B)$; if $D \sqcap E$ or $D \sqcup E \in \mathsf{clos}(C,B)$, then $\{D, E\} \subseteq \mathsf{clos}(C,B)$; if $\exists r.D \in \mathsf{clos}(C,B)$, then $D \in \mathsf{clos}(C,B)$; and if $\forall r.D \in \mathsf{clos}(C,B)$ and the role s appears in C or B, then $\{D, \forall s.D, \forall \overline{s}.D\} \subseteq \mathsf{clos}(C,B)$.[2] For the sake of simplicity, we only deal with con-

[2] The slightly unusual definition for the $\forall$ quantifier is motivated by the $\forall_+$-rule (see below), which in turn is necessary in order to capture transitive sub-roles of non-transitive roles.

cepts in *negation normal form (NNF)*, i.e. where negation appears only directly before concept names. It is easy to see that every concept can be transformed into an equivalent one in NNF in linear time.

We can now define $S_{\mathcal{SHIO}} = (\mathsf{NLE}, \mathsf{GME}, \mathsf{EL}, 1, \cdot^S, \mathcal{R}, \mathcal{C})$. Note that we use the global memory for three purposes: firstly, for transitive roles; secondly, for role inclusion axioms; thirdly, for information about concepts appearing together with a nominal.

- NLE is the set of all $\mathcal{SHIO}$ concepts,
- $\mathsf{GME} = \{(O, C) \mid O \in \mathsf{NOM} \text{ and } C \in \mathsf{NLE}\} \cup$
  $\{\mathsf{Trans}(r) \mid r \text{ is a role}\} \cup \{r \sqsubseteq^{*} s \mid r \text{ and } s \text{ are roles}\}$,[3]
- EL is the set of all $\mathcal{SHIO}$ roles, and
- for an input $\Gamma = (C, B)$, where C is a concept and B is an RBox, the function $\cdot^S$ maps Γ to a tuple $\Gamma^S = (\mathsf{nle}_\Gamma, \mathsf{gme}_\Gamma, \mathsf{el}_\Gamma, \mathsf{ini}_\Gamma)$ with
  - $\mathsf{nle}_\Gamma = \mathsf{clos}(C, B)$,
  - $\mathsf{el}_\Gamma = \{r \mid r \text{ or } \overline{r} \text{ appears in } C \text{ or } B\}$,
  - $\mathsf{gme}_\Gamma = \{(O, D) \mid O \in \mathsf{NOM} \cap \mathsf{clos}(C, B) \text{ and } D \in \mathsf{clos}(C, B)\} \cup$
    $\{\mathsf{Trans}(r) \mid r \in \mathsf{el}_\Gamma\} \cup \{r \sqsubseteq^{*} s \mid \{r, s\} \subseteq \mathsf{el}_\Gamma\}$, and
  - $\mathsf{ini}_\Gamma = \{(\{C\}, \{\mathsf{Trans}(r) \mid \mathsf{Trans}_B(r) \text{ holds}\} \cup \{r \sqsubseteq^{*} s \mid r \sqsubseteq^{*}_B s \text{ holds}\})\}$.

We now define the set of rules $\mathcal{R}$. For each pattern $P = (t, \mu)$, where $t = (V, E, n, \ell)$ has v_0 as root and depth at most 1, $\mathcal{R}(P)$ contains the following sets:

R⊓ If $C \sqcap D \in n(v_0)$ and $\{C, D\} \not\subseteq n(v_0)$, then $\mathcal{R}(P)$ contains $\{((V, E, n', \ell), \mu)\}$, where $n'(v) = n(v)$ for all $v \neq v_0$ and $n'(v_0) = n(v_0) \cup \{C, D\}$.

R⊔ If $C \sqcup D \in n(v_0)$ and $\{C, D\} \cap n(v_0) = \emptyset$, then $\mathcal{R}(P)$ contains $\{((V, E, n', \ell), \mu), ((V, E, n'', \ell), \mu)\}$, where $n'(v) = n''(v) = n(v)$ for all $v \neq v_0$, $n'(v_0) = n(v_0) \cup \{C\}$ and $n''(v_0) = n(v_0) \cup \{D\}$.

R∃ If $\exists r.C \in n(v_0)$, $v_1, \ldots, v_m$ are all the sons of v_0 with $\ell(v_0, v_i) = r$, and $C \notin n(v_i)$ for all $i, 1 \leq i \leq m$, then $\mathcal{R}(P)$ contains the set $\{P_0, P_1, \ldots, P_m\}$ with
  - $P_0 = ((V_0, E_0, n_0, \ell_0), \mu)$, where $v' \notin V$, $V_0 = V \cup \{v'\}, E_0 = E \cup \{(v_0, v')\}, n_0 = n \cup \{v' \mapsto \{C\}\}, \ell_0 = \ell \cup \{(v_0, v') \mapsto r\}$.
  - for all $i, 1 \leq i \leq m, P_i = ((V, E, n_i, \ell), \mu)$, where $n_i(v) = n(v)$ for all $v \neq v_i$ and $n_i(v_i) = n(v_i) \cup \{C\}$.

R∀ If $\forall r.C \in n(v)$ for some $v \in V$, v' is an s-neighbour of v with $C \notin n(v')$ and $s \sqsubseteq^{*} r \in \mu$, then $\mathcal{R}(P)$ contains $\{((V, E, n', \ell), \mu)\}$ with $n'(v) = n(v)$ for $v \neq v'$ and $n'(v') = n(v') \cup \{C\}$.

R$\forall_+$ If $\forall r.C \in n(v)$, $\{\mathsf{Trans}(s), s \sqsubseteq^{*} r, s' \sqsubseteq^{*} s\} \subseteq \mu$ and for some $v \in V$, v' is an s'-neighbour of v with $\forall s.C \notin n(v')$, then $\mathcal{R}(P)$ contains $\{((V, E, n', \ell), \mu)\}$ with $n'(v) = n(v)$ for $v \neq v'$ and $n'(v') = n(v') \cup \{\forall s.C\}$.

R↑ If $\{O, C\} \subseteq n(v_0)$ for some $O \in \mathsf{NOM}$ and $(O, C) \notin \mu$, then $\mathcal{R}(P)$ contains $\{((V, E, n, \ell), \mu')\}$, where $\mu' = \mu \cup \{O, C\}$.

R↓ If $O \in n(v_0)$ for an $O \in \mathsf{NOM}$, $(O, C) \in \mu$ and $C \notin n(v_0)$, then $\mathcal{R}(P)$ contains $\{((V, E, n', \ell), \mu)\}$, where $n'(v) = n(v)$ for $v \neq v_0$ and $n'(v_0) = n(v_0) \cup \{C\}$.

[3] Note that the relations Trans_B and $\sqsubseteq^{*}_B$ (i.e. the semantics) are distinguished from the global memory elements (the syntax) by the index $_B$.

Most of these rules correspond directly to the "standard" rules known from DL tableaus, with the exception of R$\exists$, which in our framework is non-deterministic. The reason for this is that with a deterministic rule, this TS would not be admissible (see [3] for a discussion of this issue). In an implementation, a deterministic rule would be preferable due to efficiency considerations, since it is easy to see that the creation of duplicate nodes does not compromise completeness of the decision procedure. Finally, the set $\mathcal{C}$ of clash patterns contains all patterns $((V, E, n, \ell), \mu)$ of depth 0 with node v_0 such that $\{C, \neg C\} \subseteq n(v_0)$.

Lemma 2. *The TS $S_{\mathcal{SHIO}}$ is sound and complete for $\mathcal{SHIO}$ satisfiability.*

From this, we can derive that $\mathcal{SHIO}$ satisfiability is decidable through a tableau algorithm, and we know that for the blocking condition, equality blocking suffices, i.e. we do not need pair-wise blocking as e.g. for $\mathcal{SHIQ}$ [5]. The reason for this is that we use only patterns of depth at most 1. We can also derive the ExpTime upper bound:

Theorem 3. *Satisfiability for $\mathcal{SHIO}$ concepts w.r.t. RBoxes is decidable in* ExpTime.

Since $\mathcal{SHIO}$ is an extension of $\mathcal{ALC}$ with TBoxes, for which satisfiability is known to be ExpTime-hard [7], it follows that $\mathcal{SHIO}$ satisfiability is ExpTime-complete.

5 Conclusion

We have introduced the tableau framework for ExpTime logics, defined the description logic $\mathcal{SHIO}$ and developed a tableau system for $\mathcal{SHIO}$. From this, we can derive an automata algorithm deciding satisfiability of $\mathcal{SHIO}$ concepts w.r.t. RBoxes in ExpTime and a tableau algorithm which promises to be amenable to the well-known optimisations and thus to perform well in practice. We believe that the simplicity of the proofs justifies the additional overhead resulting from the formalisation of the algorithm within the tableau framework.

References

[1] F. Baader. Logic-based knowledge representation. In *Artificial Intelligence Today, Recent Trends and Developments*, number 1600 in LNCS. Springer-Verlag, 1999.

[2] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69, 2001.

[3] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57:1–33, 2003.

[4] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In *Proceedings of IJCAI-01*, 2001.

[5] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of LPAR'99*, number 1705 in LNAI. Springer-Verlag, 1999.

[6] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *IJCAR-01*, volume 2083 of *LNAI*. Springer-Verlag, 2001.

[7] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of KR-94*, Bonn, 1994.

[8] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. Tech. Rep. SR-88-21, Univ. Kaiserslautern, Germany, 1988.

Appendix

We will now give the proofs for Lemma 2 and Theorem 3, for which we have to recall some additional definitions from [3]. An S-tree t is called *saturated*[4] if no rule is applicable, i.e. if there is no pattern P matching t with $\mathcal{R}(P) \neq \emptyset$. It is called *clash-free* if no clash trigger $P \in \mathcal{C}$ matches t.

For a TS S, we require *admissibility*: the rules may only add information, but never remove anything from the tree; and if a rule is applicable, then it can be applied in such a way that the tree approaches saturation. This last property is the key to proving completeness of tableau systems: we distinguish an S-tree *compatible with* Γ, i.e. a tree which is labelled according to Γ^S, from an S-tree *for* Γ, i.e. a tree which can be constructed effectively from an initial tree by rule application. For the ExpTime upper bound, we also require ExpTime-*admissibility*: for an input Γ, the cardinality of the sets nle, gme and el and the size of each element has to be polynomial in the size of Γ. If the prerequisites of our framework are met by S, then the existence of a clash-free and saturated S-tree compatible with Γ implies the existence of such a tree for Γ, which means that in the following proofs, we need not show that a particular tree can effectively be created by rule application.

Lemma 2 (Soundness). *The TS $S_{\mathcal{SHIO}}$ is sound for $\mathcal{SHIO}$ satisfiability.*

Proof. From a saturated and clash-free S-tree (t, μ) with $t = (V, E, n, \ell)$, we generate a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows: $\Delta^{\mathcal{I}} = \{v_O \mid O \in \mathsf{NOM} \cap \mathsf{clos}(C, B)\} \cup \{v \mid v \in V \text{ and } n(v) \cap \mathsf{NOM} = \emptyset\}$, i.e. we have one individual for each nominal name and one individual for every tree node that is not labelled with a nominal. A concept name C is interpreted as follows: for every $O \in \mathsf{NOM} \cap \mathsf{clos}(C, B)$, $v_O \in C^{\mathcal{I}}$ iff there is a node $v \in V$ with $\{O, C\} \subseteq n(v)$. Since R$\uparrow$ and R$\downarrow$ are not applicable, all nodes whose labels contain the same nominal symbol have exactly the same label, and thus $\cdot^{\mathcal{I}}$ is well-defined. For all other individuals, $v \in C^{\mathcal{I}}$ iff $C \in n(v)$.

For a role r, $r^{\mathcal{I}}$ is the smallest set satisfying the following conditions: if $\ell(v, w) = r$ or $\ell(w, v) = \overline{r}$, then $(v^{\mathcal{I}}, w^{\mathcal{I}}) \in r^{\mathcal{I}}$; if $s \boxed{\sqsubseteq}^{*} r \in \mu$, then $s^{\mathcal{I}} \subseteq r^{\mathcal{I}}$; if $\mathsf{Trans}(r) \in \mu$, then $r^{\mathcal{I}}$ is closed under transitivity.

We will now show by induction that complex concepts are interpreted correctly. By definition, all individuals belong to the interpretation of the concept names in their labels, and the interpretation of a nominal contains exactly one element. From our construction, it follows directly that the role hierarchy is respected and transitive roles are interpreted correctly. For a conjunct $C \sqcap D$ (disjunct $C \sqcup D$) in a node label $n(v)$, since R$\sqcap$ (R$\sqcup$) is not applicable, it follows that C and D (C or D) are contained in $n(v)$, and by induction, $v^{\mathcal{I}}$ is contained in $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ ($C^{\mathcal{I}} \cup D^{\mathcal{I}}$).

If $\exists r.C \in n(v)$, we assume w.l.o.g. that r is a role name (if it is an inverse role, the argument is analogous). Since R$\exists$ is not applicable, there exists an r-son w of v with $C \in n(w)$. By construction, $(v, w) \in r^{\mathcal{I}}$ and $w \in C^{\mathcal{I}}$.

[4] Usually, this property is called "completeness"; we use the word "saturated" in order to avoid confusion with the notion of completeness of the decision procedure.

If $\forall r.C \in n(v)$, we again assume that r is a role name. There are two possible reasons why $(v^{\mathcal{I}}, w^{\mathcal{I}})$ can be contained in $r^{\mathcal{I}}$: firstly, if w is an s-neighbour of v for some s with $s \sqsubseteq^* r \in \mu$. In this case, it follows that $C \in n(w)$ because R$\forall$ is not applicable, and thus $w^{\mathcal{I}} \in C^{\mathcal{I}}$. Secondly, if there exist roles $s, s_1, \ldots, s_k$ s.th. $\{\mathsf{Trans}(s), s \sqsubseteq^* r, s_i \sqsubseteq^* s\} \subseteq \mu$ for all $i \in \{1, \ldots, k\}$ and there is an s_i-chain from v to w, i.e. a sequence of nodes $v_1, v_2, \ldots, v_n$ s.th., for all edges $e \in \{(v, v_1), (v_1, v_2), \ldots, (v_n, w)\}$, it holds that $e \in E$ and $\ell(e) = s_i$ for some i. In this case, since R$\forall_+$ is not applicable, all nodes $v_1, \ldots, v_k$ are labelled with $\forall s.C$, and, since R$\forall$ is not applicable to v_k, $n(w)$ contains C. □

Lemma 2 (Completeness). *The TS $S_{\mathcal{SHIO}}$ is complete for $\mathcal{SHIO}$ satisfiability.*

Proof. We have to show that if there exists a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for an input $\Gamma = (C, B)$, then there also exists a clash-free and saturated S-tree (t, μ) with $t = (V, E, n, \ell)$ for Γ. We will create (t, μ) by unravelling $\mathcal{I}$: firstly, we add the appropriate transitivity axioms ($\mathsf{Trans}(r)$ if $\mathsf{Trans}_B(r)$ holds) and role inclusion axioms ($r \sqsubseteq^* s$ if $r \sqsubseteq^*_B s$ holds) to μ. The tree t is inductively defined as follows: since $\mathcal{I} \models \Gamma$, there is an individual i_0 in $\Delta^{\mathcal{I}}$ which satisfies C. We start with $V = \{v_0\}$ and define $n(v_0)$ as the set of all concepts in $\mathsf{clos}(C, B)$ which i_0 satisfies. We define a function $\pi : V \to \Delta^{\mathcal{I}}$ and set $\pi(v_0) = i_0$.

Then we iterate, for every node v, the following procedure: for every existential formula $\exists r.D \in n(v)$ we choose a witness individual $i \in \Delta^{\mathcal{I}}$ with $i \in D^{\mathcal{I}}$ and $(\pi(v), i) \in r^{\mathcal{I}}$ (such a witness exists since $\mathcal{I}$ is a model). We create a new node w with $\pi(w) = i$, $(v, w) \in E$ and $\ell(v, w) = r$. Note that the tree width is bounded by the size of $\mathsf{clos}(C, B)$. Again, we label w with the appropriate concepts in $\mathsf{clos}(C, B)$ and then continue the iteration. For every nominal concept O, we add to μ the pair (O, D) for every concept $D \in \mathsf{clos}(C)$ which the unique element i_O of $O^{\mathcal{I}}$ satisfies.

It is easy to see that (t, μ) is compatible with Γ and clash-free. We will now show that it is also saturated: from the definition of clos, it follows that R$\sqcap$ and R$\sqcup$ are not applicable. If a node label $n(v)$ contains a concept $\exists r.D$, then by construction of t, there is an r-successor of v which is labelled with D. Likewise, if $\forall r.D \in n(v)$, all r-neighbours of v are labelled with D. If $\forall r.D \in n(v)$, μ contains $s \sqsubseteq^* r$, $s' \sqsubseteq^* s$ and $\mathsf{Trans}(s)$, and there is an s'-neighbour w of v, then, since $\mathcal{I}$ is a model, $(\pi(v), \pi(w)) \in s^{\mathcal{I}}$ and, since $s^{\mathcal{I}}$ is transitive, for every node u with $(\pi(w), \pi(u)) \in s^{\mathcal{I}}$, it also holds that $(\pi(v), \pi(u)) \in s^{\mathcal{I}}$, and therefore $\pi(u) \in D^{\mathcal{I}}$. Thus, $\pi(w) \models \forall r.D$ and, since $s \sqsubseteq^*_B r$, $v(w)$ contains $\forall s.D$, which means that R$\forall_+$ is not applicable. Finally, since every node n with $\pi(n) = i_O$ for a nominal O is labelled with exactly those concepts for which μ contains (O, C), R$\uparrow$ and R$\downarrow$ are not applicable. □

Note that in these proofs, we did not have to deal with termination and a blocking condition, and we did not have to prove that the completion tree can effectively be constructed by rule applications. All these results follow from the tableau framework.

Theorem 3. *Satisfiability for $\mathcal{SHIO}$ concepts w.r.t. RBoxes is decidable in* ExpTime.

Proof. It is easy to see that S is ExpTime-admissible: e.g. the size of $\mathsf{nle}_{\mathcal{S}}(\Gamma)$ and $\mathsf{gme}_{\mathcal{S}}(\Gamma)$ is quadratic in the size of the input. The maximum required width of an S-tree is bounded by the number of existential subformulas of the input concept since at most one successor is needed for every existential subformula. Soundness and completeness have been shown above. □