



HAL
open science

A Worst-Case Optimal Tableaux-Based Algorithm for SHOIQ

Chan Le Duc, Myriam Lamolle, Olivier Curé

► **To cite this version:**

Chan Le Duc, Myriam Lamolle, Olivier Curé. A Worst-Case Optimal Tableaux-Based Algorithm for SHOIQ. [Research Report] 2011, pp.19. inria-00570161v1

HAL Id: inria-00570161

<https://inria.hal.science/inria-00570161v1>

Submitted on 27 Feb 2011 (v1), last revised 2 Jun 2012 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Worst-Case Optimal Tableaux-Based Algorithm for \mathcal{SHOIQ}

Chan Le Duc
Myriam Lamolle

LIASD Université Paris 8 - IUT de Montreuil

LEDUC@IUT.UNIV-PARIS8.FR

LAMOLLE@IUT.UNIV-PARIS8.FR

Olivier Curé

Université Marne La Vallé

OCURE@UNIV-MLV.FR

Abstract

In this paper, we propose a novel tableaux-based algorithm for checking consistency of a \mathcal{SHOIQ} knowledge base. This algorithm creates a structure consisting of a set of trees which are linked together by a partitioning function over the set of all the trees' nodes. This construction allows us to avoid building explicitly a graph with cycles and ensures that the size of created structures is exponential in size of \mathcal{SHOIQ} knowledge bases. From this, we show that the novel algorithm is worst-case optimal for checking consistency of a \mathcal{SHOIQ} knowledge base.

1. Introduction

The ontology language OWL-DL (Patel-Schneider, Hayes, & Horrocks, 2004) is widely used to formalize semantic resources on the Semantic Web. This language is mainly based on the description logic \mathcal{SHOIQ} which is known to be decidable (Tobies, 2000). There were several works on the consistency problem of a \mathcal{SHOIQ} knowledge base. These works not only have shown decidability and complexity of the problem but also have led to develop and implement efficient systems for reasoning on OWL-based ontologies. Especially, (Tobies, 2000) has shown that the consistency problem of a \mathcal{SHOIQ} knowledge base is NEXPTIME-complete.

Recently, the tableaux-based algorithms presented in (Motik, Shearer, & Horrocks, 2009) and (Horrocks & Sattler, 2007) are used to develop reasoners such as FaCT++, Pellet and Hermit. This kind of algorithms relies on two structures, the so-called *tableau* and *completion graph*. Roughly speaking, a tableau represents a model of a knowledge base and it is possibly infinite. A tableau translates satisfiability of all given concept and role inclusion axioms into the satisfiability of constraints imposed *locally* on each individual of the tableau. To check consistency of a knowledge base, tableaux-based algorithms try to build a completion graph whose finiteness is ensured by a technique, the so-called *blocking technique*. The underlying idea of the blocking technique is to detect “loops” which are repeated tree-like pieces of a completion graph. As a result, this technique works only for tree-like structures on which we can define *blocked* and *blocking* nodes such that (i) each blocked node must have a blocking ancestor, and (ii) each node x between a blocking and blocked node may be duplicated infinitely without violating the semantic constraints imposed by concepts in the label of x . For this reason, tableaux-based algorithms manage to “push” nominal nodes (i.e. nodes contain nominals in their label) outside from tree-like structures.

Although experiments indicate that these tableaux-based algorithms have good behaviours in practice, none of them was shown to be worst-case optimal. The contribution of the present paper consists of proposing a worst-case optimal tableaux-based algorithm for checking consistency of a \mathcal{SHOIQ} knowledge base. This algorithm relies on the tableaux-based algorithm presented in

(Horrocks & Sattler, 2007), from which we have learned the following characteristics of completion graphs for $SHOIQ$:

- A completion graph for $SHOIQ$ is formed of *nominal nodes* whose label contains a nominal, and *non-nominal nodes*. Such a graph consists of two parts: the so-called *non-tree part* and *tree-like part*. The non-tree part contains nominal nodes which can be arbitrarily interconnected. Nevertheless, the tree-like part includes non-nominal nodes which form a set of tree-like structures rooted in a nominal node i.e. each non-nominal node has at most one predecessor and might have a nominal successor.
- This structure of completion graphs for $SHOIQ$ makes a crucial impact on behaviours of the tableaux-based algorithm. To ensure termination of such an algorithm, (i) we reuse the usual blocking condition for the tree-like part, (ii) we have to control the interaction between these two parts such that if there is a merge between a non-nominal and nominal node the tree-like structure should be maintained in order to take advantage of the blocking condition, and (iii) we have to prevent a blocking node from having a nominal descendant x such that the non-nominal predecessor y of x is a S -neighbor of x , the label of y contains a concept C and the label of x includes a term such as $(\leq nS.C)$. Such a node x may be “embarrassing”.
- To obtain the conditions mentioned for the termination, (Horrocks & Sattler, 2007) introduced some new rules which can add new nominal nodes with new nominals and push “embarrassing” nodes with their labels into the non-tree part such that all merges related to such nodes are no longer required. It was shown that this process may add a double exponential number of new nominal nodes. This explains why the tableaux-based algorithm presented in (Horrocks & Sattler, 2007) is not worst-case optimal.

These observations would make us think that adding new nominal nodes to obtain termination is not intrinsically related to the semantic constraints imposed by the logic constructors in $SHOIQ$. This behaviour is rather originated from a selected structure on which we know how to establish a mechanism to ensure termination. This structure requires to push “embarrassing” nodes into the non-tree part by generating new nominal neighbors, then merging one of them into a non-nominal neighbor. This implies that we could avoid adding new nominal nodes if the tree-shaped structure can be maintained by some way such that it allows us to avoid merging *explicitly* nominal nodes.

The algorithm presented in this paper is aimed at using a structure which contains no non-tree part. As a result, we can reuse the usual blocking mechanism to ensure termination. The underlying idea of our approach is to introduce a function, namely *partitioning function*, that partitions the set of all nodes of tree-like structures. Roughly speaking, each partition represents a set of nodes that could be merged together. The expansion rules perform necessary propagations of labels between nodes and edges as if they are merged. More precisely, the $SHIQ$ rules (i) perform *local merges*, i.e. the merges of a neighbor into another one are triggered by terms such as $(\leq nS.C)$, (ii) propagate the node labels between nodes of a partition, and (iii) propagate the edge labels between edges that connect neighbor partitions. The $SHOIQ$ rules in turn perform *global merges*, i.e. the merges of a neighbor partition into another one are triggered by merging nominal nodes or terms such as $(\leq nS.C)$. The global merges update the partitioning function and do no change to the tree-like structures except for labels.

The present paper is structured as follows. In the next section, we describe the logic $SHOIQ$ and the consistency problem for a $SHOIQ$ knowledge base. In Section 3, we provide a definition

of tableaux for \mathcal{SHOIQ} . Section 2 describes in detail an algorithm for checking consistency of a \mathcal{SHOIQ} knowledge base and shows that this algorithm is a worst-case optimal decision procedure for \mathcal{SHOIQ} . Finally, we discuss the results and future work.

2. The Description Logic \mathcal{SHOIQ}

In this section, we present the syntax, semantics and inference problems of the logic \mathcal{SHOIQ} , which are similar to those introduced in (Horrocks & Sattler, 2007).

Definition 1 Let \mathbf{R} be a non-empty set of role names and $\mathbf{R}_+ \subseteq \mathbf{R}$ be a set of transitive role names. We denote $\mathbf{R}_\perp = \{P^- \mid P \in \mathbf{R}\}$.

* The set of \mathcal{SHOIQ} -roles is $\mathbf{R} \cup \mathbf{R}_\perp$. A role inclusion axiom is of the form $R \sqsubseteq S$ for two \mathcal{SHOIQ} -roles R and S . A role hierarchy \mathcal{R} is a finite set of role inclusion axioms.

* An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (domain) and a function $\cdot^{\mathcal{I}}$ which maps each role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that

$$R^{-\mathcal{I}} = \{\langle x, y \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x \rangle \in R^{\mathcal{I}}\} \text{ for all } R \in \mathbf{R}, \text{ and} \\ \langle x, z \rangle \in S^{\mathcal{I}}, \langle z, y \rangle \in S^{\mathcal{I}} \text{ implies } \langle x, y \rangle \in S^{\mathcal{I}} \text{ for each } S \in \mathbf{R}_+$$

An interpretation \mathcal{I} satisfies a role hierarchy \mathcal{R} if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$. Such an interpretation is called a model of \mathcal{R} , denoted by $\mathcal{I} \models \mathcal{R}$.

* To simplify notations for nested inverse roles, we define a functions \cdot^\ominus as follows:

$$R^\ominus = \begin{cases} R^- & \text{if } R \in \mathbf{R}, \\ S & \text{if } R = S^- \text{ and } S \in \mathbf{R} \end{cases}$$

* A relation \boxplus is defined as the transitive-reflexive closure \mathcal{R}^+ of \sqsubseteq on $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\}$. We denote $S \equiv R$ iff $R \boxplus S$ and $S \boxplus R$.

* A role R is called simple w.r.t. \mathcal{R} iff $Q \boxplus R \notin \mathcal{R}^+$ for each $Q \in \mathbf{R}_+ \cup \{P^\ominus \mid P \in \mathbf{R}_+\}$. We define a function $\text{Trans}(R)$ which returns true iff $Q \boxplus R \in \mathcal{R}^+$ for some $Q \in \mathbf{R}_+ \cup \{P^\ominus \mid P \in \mathbf{R}_+\}$.

Definition 2 Let \mathbf{C} be a non-empty set of concept names with a subset $\mathbf{C}_o \subseteq \mathbf{C}$ of nominals.

* The set of \mathcal{SHOIQ} -concepts is inductively defined as the smallest set containing all C in \mathbf{C} , \top , $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n.S.C)$ and $(\geq n.S.C)$ where n is a positive integer, C and D are \mathcal{SHOIQ} -concepts, R is an \mathcal{SHOIQ} -role and S is a simple role. We denote \perp for $\neg \top$.

* An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ (domain) and a function $\cdot^{\mathcal{I}}$ which maps each concept name to a subset of $\Delta^{\mathcal{I}}$ such that

$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$, $\text{card}\{o^{\mathcal{I}}\} = 1$ for all $o \in \mathbf{C}_o$,

$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}, \\ (\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}, \\ (\geq n.S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{card}\{y \in C^{\mathcal{I}} \mid \langle x, y \rangle \in S^{\mathcal{I}}\} \geq n\}, \\ (\leq n.S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{card}\{y \in C^{\mathcal{I}} \mid \langle x, y \rangle \in S^{\mathcal{I}}\} \leq n\}$$

where $\text{card}\{S\}$ is denoted for the cardinality of a set S .

* $C \sqsubseteq D$ is called a general concept inclusion (GCI) where C, D are \mathcal{SHOIQ} -concepts (possibly complex), and a finite set of GCIs is called a terminology \mathcal{T} . An interpretation \mathcal{I} satisfies a GCI

$C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and \mathcal{I} satisfies a terminology \mathcal{T} if \mathcal{I} satisfies each GCI in \mathcal{T} . Such an interpretation is called a model of \mathcal{T} , denoted by $\mathcal{I} \models \mathcal{T}$.

* A concept C is called satisfiable w.r.t. a role hierarchy \mathcal{R} and a terminology \mathcal{T} iff there is some interpretation \mathcal{I} such that $\mathcal{I} \models \mathcal{R}$, $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a model of C w.r.t. \mathcal{R} and \mathcal{T} . A pair $(\mathcal{T}, \mathcal{R})$ is called a *SHOIQ knowledge base* and said to be consistent if there is a model \mathcal{I} of both \mathcal{T} and \mathcal{R} , i.e., $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{R}$.

* A concept D subsumes a concept C w.r.t. \mathcal{R} and \mathcal{T} , denoted by $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in each model \mathcal{I} of $(\mathcal{T}, \mathcal{R})$.

Since negation is allowed in the logic *SHOIQ*, unsatisfiability and subsumption w.r.t. $(\mathcal{T}, \mathcal{R})$ can be reduced each other: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable. In addition, we can reduce satisfiability of concepts to knowledge base consistency : C is satisfiable w.r.t. a knowledge base $(\mathcal{T}, \mathcal{R})$ iff $(\mathcal{T} \cup \{o \sqsubseteq C\}, \mathcal{R})$ is consistent where o is a new nominal. Thanks to these reductions, it suffices to study knowledge base consistency.

For the ease of construction, we assume all concepts to be in *negation normal form* (NNF) i.e. negation occurs only in front of concept names. Any *SHOIQ*-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in (Horrocks, Sattler, & Tobies, 1999b). For a concept C , we denote the nnf of C by $\text{nnf}(C)$ and the nnf of $\neg C$ by $\dot{\neg}C$

Let D be an *SHOIQ*-concept in NNF. We define $\text{cl}(D)$ to be the smallest set that contains all sub-concepts of D including D .

For a knowledge base $(\mathcal{T}, \mathcal{R})$, we define a set $\text{cl}(\mathcal{T}, \mathcal{R})$ as follows:

$$\begin{aligned} \text{cl}(\mathcal{T}, \mathcal{R}) &= \bigcup_{C \sqsubseteq D \in \mathcal{T}} \text{cl}(\text{nnf}(\neg C \sqcup D), \mathcal{R}) \text{ where} \\ \text{cl}(E, \mathcal{R}) &= \text{cl}(E) \cup \{\dot{\neg}C \mid C \in \text{cl}(E)\} \cup \\ &\quad \{\forall S.C \mid (\forall R.C \in \text{cl}(E), S \sqsubseteq R) \text{ or } (\dot{\neg} \forall R.C \in \text{cl}(E), S \sqsubseteq R) \\ &\quad \text{where } S \text{ occurs in } \mathcal{T} \text{ or } \mathcal{R}\} \cup \\ &\quad \{\bowtie mS.C \mid (\leq nS.C) \in \text{cl}(E), \bowtie \in \{\leq, \geq\} \text{ and } 1 \leq m \leq n\} \end{aligned} \quad (1)$$

We use $\mathbf{R}_{(\mathcal{T}, \mathcal{R})}$ to denote the set of all role names occurring in \mathcal{T}, \mathcal{R} with their inverse.

The definition of $\text{cl}(\mathcal{T}, \mathcal{R})$ is inspired from the Fischer-Ladner closure that was introduced in (Fischer & Ladner, 1979). The closure $\text{cl}(\mathcal{T}, \mathcal{R})$ contains not only a set $\text{sub}(\mathcal{T})$ of all sub-concepts syntactically obtained from \mathcal{T} but also sub-concepts that are semantically derived from $\text{sub}(\mathcal{T})$ w.r.t. \mathcal{R} . For instance, (i) if $\forall S.C \in \text{sub}(\mathcal{T})$ and $R \sqsubseteq S \in \mathcal{R}$ then $\forall R.C \in \text{cl}(\mathcal{T}, \mathcal{R})$, or (ii) if $(\leq nS.C) \in \text{sub}(\mathcal{T})$ with $n \geq 1$ then $(\leq mS.C), (\geq mS.C) \in \text{cl}(\mathcal{T}, \mathcal{R})$ for all $1 \leq m \leq n$. This means that $\text{cl}(\mathcal{T}, \mathcal{R})$ must be large enough to cover all sub-concepts which represent potential nondeterminisms. These extensions of $\text{cl}(\mathcal{T}, \mathcal{R})$ are anticipated for the construction of tableaux described in Section 3.

3. Tableaux for *SHOIQ*

Tableau structure is introduced to represent a model of a knowledge base. Properties in the tableau definition express the semantic constraints resulting directly from the logic constructors in *SHOIQ*.

These properties show how a given individual of a tableau establishes relationships with other individuals in order to semantically satisfy each concept associated with the individual in question. As a result, these properties would guide us to design expansion rules that allow to build a finite representation of a model.

All properties of Definition 3 except for P15 are taken from (Horrocks et al., 1999b) and (Horrocks & Sattler, 2007). If we say that an individual s has a R -neighbor t when $\langle s, t \rangle$ is an instance of R , then any application of properties to s imposes constraints only on s 's neighbors or itself. This characteristic of tableaux will be called *local property*. The presence of transitivity of roles makes local property of tableaux more difficult to be ensured. (P6) in Definition 3 allows us to overcome this difficulty by avoiding representing all instances of transitive roles in a tableau. In order to satisfy a concept such as $\forall S.C$ from an individual s where S is transitive, instead of propagating the concept C along with each instance of the transitive role S , it is sufficient to “push” C and the term $\forall S.C$ to each S -neighbor t of s . As a consequence, local property is guaranteed for \mathcal{SHIQ} tableaux. However, it is much more difficult to preserve local property of tableaux for logics which allow transitive closure of roles. For instance, tableaux for \mathcal{SHI} with transitive closure of roles, introduced in (Le Duc, 2009), do not have local property.

According to this argument, P13 in Definition 3 is responsible for the loss of local property in \mathcal{SHOIQ} tableaux since P13 can be applied to two arbitrary individuals with which a same nominal are associated. Local property of tableaux makes an important impact on the way by which a tableaux-based algorithm can build completion graphs. The loss of local property may lead to augment the complexity, e.g., from EXPTIME-complete for \mathcal{SHIQ} to NEXPTIME-complete for \mathcal{SHOIQ} .

Definition 3 Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ}_+ knowledge base. A tableau T for $(\mathcal{T}, \mathcal{R})$ is defined to be a triplet $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that \mathbf{S} is a set of individuals, $\mathcal{L}: \mathbf{S} \rightarrow 2^{\text{cl}(\mathcal{T}, \mathcal{R})}$ and $\mathcal{E}: \mathbf{R}_{(\mathcal{T}, \mathcal{R})} \rightarrow 2^{\mathbf{S} \times \mathbf{S}}$. For all $s \in \mathbf{S}$, $C, C_1, C_2 \in \text{cl}(\mathcal{T}, \mathcal{R})$ and $R, S \in \mathbf{R}_{(\mathcal{T}, \mathcal{R})}$, T satisfies the following properties:

- P1 If $C_1 \sqsubseteq C_2 \in \mathcal{T}$ then $\text{nnf}(\neg C_1 \sqcup C_2) \in \mathcal{L}(s)$,
- P2 If $C \in \mathcal{L}(s)$ then $\neg C \notin \mathcal{L}(s)$,
- P3 If $C_1 \sqcap C_2 \in \mathcal{L}(s)$ then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,
- P4 If $C_1 \sqcup C_2 \in \mathcal{L}(s)$ then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,
- P5 If $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$,
- P6 If $\forall S.C \in \mathcal{L}(s)$, $Q \sqsubseteq S$, $\text{Trans}(Q)$ and $\langle s, t \rangle \in \mathcal{E}(Q)$ then $\forall Q.C \in \mathcal{L}(t)$,
- P7 If $\exists P.C \in \mathcal{L}(s)$ then there is some $t \in \mathbf{S}$ such that $\langle s, t \rangle \in \mathcal{E}(P)$ and $C \in \mathcal{L}(t)$,
- P8 $\langle s, t \rangle \in \mathcal{E}(R)$ iff $\langle t, s \rangle \in \mathcal{E}(R^\ominus)$,
- P9 If $\langle s, t \rangle \in \mathcal{E}(R)$ and $R \sqsubseteq S$ then $\langle s, t \rangle \in \mathcal{E}(S)$,
- P10 If $(\leq nS.C) \in \mathcal{L}(s)$ then $\text{card}\{S^T(s, C)\} \leq n$,
- P11 If $(\geq nS.C) \in \mathcal{L}(s)$ then $\text{card}\{S^T(s, C)\} \geq n$,
- P12 If $(\leq nS.C) \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$ then $C \in \mathcal{L}(t)$ or $\neg C \in \mathcal{L}(t)$ where $S^T(s, C) := \{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \wedge C \in \mathcal{L}(t)\}$,
- P13 If $\circ \in \mathcal{L}(s) \cap \mathcal{L}(t)$ for some $\circ \in \mathbf{C}_o$ then $s = t$,
- P14 For each $\circ \in \mathbf{C}_o$, if \circ occurs in \mathcal{T} then there is some $s \in \mathbf{S}$ such that $\circ \in \mathcal{L}(s)$.
- P15 If $(\leq nS.C) \in \mathcal{L}(s)$ and there is some t such that $C \in \mathcal{L}(t)$ and $\langle s, t \rangle \in \mathcal{E}(S)$ then there is some $1 \leq m \leq n$ such that $\{(\leq mS.C), (\geq mS.C)\} \subseteq \mathcal{L}(s)$.

Regarding the definition of tableaux for \mathcal{SHOIQ} presented in (Horrocks & Sattler, 2007), we add a new property, namely P15. This new property imposes an exact number of S -neighbor

individuals t of s if $(\leq nS.C) \in \mathcal{L}(s)$. This property makes explicit the nondeterminism implied from the semantics of $(\leq nS.C)$ and leads to need an extra expansion rule, namely NN_φ -rule, introduced in Figure 2. The presence of this rule may have an impact on the so-called “pay-as-you-go” behaviour of the tableaux-base algorithm.

Lemma 4 *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. $(\mathcal{T}, \mathcal{R})$ is consistent iff there is a tableau for $(\mathcal{T}, \mathcal{R})$.*

A proof of a similar lemma for \mathcal{SHIQ} tableaux can be found in (Horrocks, Sattler, & Tobies, 1999a). From this proof, it is not hard to show Lemma 4 for \mathcal{SHOIQ} tableaux with the new properties P13, P14 and P15.

4. A novel decision procedure for \mathcal{SHOIQ}

In this section, we introduce \mathcal{SHOIQ} -trees by Definition 5. Intuitively, a \mathcal{SHOIQ} -tree is defined by considering nominals as non-nominal concepts. The important difference of \mathcal{SHOIQ} -trees from completion trees for \mathcal{SHIQ} is the presence of a *blocking condition with two layers* that is governed by *1-iterated* and *2-iterated* nodes. A 1-iterated node in a \mathcal{SHOIQ} -tree whose root is a nominal node corresponds exactly to a blocked node in a completion tree for \mathcal{SHIQ} . A 2-iterated node y in a \mathcal{SHOIQ} -tree is a descendant of a 1-iterated node x such that the blocking condition for y is applied from x i.e. x is considered as root of the subtree on which the condition blocking is applied for each 2-iterated descendant y of x . The reason for introducing the blocking condition with two layers is to facilitate the definitions of inner and blocked nodes which will be presented in Definition 6.

Definition 5 *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. For each $o \in \mathbf{C}_o$, a \mathcal{SHOIQ} -tree for $(\mathcal{T}, \mathcal{R})$, denoted by $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \hat{x}_o, \neq)$, is defined as follows:*

* V_o is a set of nodes containing a root node $\hat{x}_o \in V_o$. Each node $x \in V_o$ is labelled with a function \mathcal{L} such that $\mathcal{L}(x) \subseteq \text{cl}(\mathcal{T}, \mathcal{R})$ and $o \in \mathcal{L}(\hat{x}_o)$. A node $x \in V_o$ is called nominal if $o' \in \mathcal{L}(x)$ for some $o' \in \mathbf{C}_o$. In addition, \neq is a symmetric binary relation over V_o .

* E_o is a set of edges. Each edge $\langle x, y \rangle \in E_o$ is labelled with a function \mathcal{L} such that $\mathcal{L}(\langle x, y \rangle) \subseteq \mathbf{R}_{(\mathcal{T}, \mathcal{R})}$.

* If $\langle x, y \rangle \in E_o$ then y is called a successor of x , denoted by $y \in \text{succ}^1(x)$, or x is called the predecessor of y , denoted by $x = \text{pred}^1(y)$. In this case, we say that x is a neighbor of y or y is a neighbor of x . If $z \in \text{succ}^n(x)$ (resp. $z = \text{pred}^n(x)$) and y is a successor of z (resp. y is the predecessor of z) then $y \in \text{succ}^{(n+1)}(x)$ (resp. $y = \text{pred}^{(n+1)}(x)$) for all $n \geq 0$ where $\text{succ}^0(x) = \{x\}$ and $\text{pred}^0(x) = x$.

* A node y is called a descendant of x if $y \in \text{succ}^n(x)$ for some $n > 0$. A node y is called an ancestor of x if $y = \text{pred}^n(x)$ for some $n > 0$.

* A node y is called a R -successor of x , denoted by $y \in \text{succ}_R^1(x)$ (resp. y is called the R -predecessor of x , denoted by $y = \text{pred}_R^1(x)$) if there is some role R' such that $R' \in \mathcal{L}(\langle x, y \rangle)$ (resp. $R' \in \mathcal{L}(\langle y, x \rangle)$) and $R' \sqsubseteq R$. A node y is called a R -neighbor of x if y is either a R -successor or R -predecessor of x . If z is a R -successor of y (resp. z is the R -predecessor of y) and $y \in \text{succ}_R^n(x)$ (resp. $y = \text{pred}_R^n(x)$) then $z \in \text{succ}_R^{(n+1)}(x)$ (resp. $z = \text{pred}_R^{(n+1)}(x)$) for $n \geq 0$ with $\text{succ}_R^0(x) = \{x\}$ and $x = \text{pred}_R^0(x)$.

* For a node x and a role S , we define the set $S^{\mathbf{T}}(x, C)$ of x 's S -neighbors as follows:

$$S^{\mathbf{T}}(x, C) = \{y \in V \mid y \text{ is a } S\text{-neighbor of } x \text{ and } C \in \mathcal{L}(x)\}$$

* A node x is called iterated by y w.r.t. a node v , denoted by $y = \mathbf{b}(x)$, if x has no nominal ancestor except for \hat{x}_o and there are numbers $n, m > 0$ and nodes x', y' such that

1. $v = \text{pred}^n(y)$, $y = \text{pred}^m(x)$, and
2. $x' = \text{pred}^1(x)$, $y' = \text{pred}^1(y)$, and
3. $\mathcal{L}(x) = \mathcal{L}(y)$, $\mathcal{L}(x') = \mathcal{L}(y')$, and
4. $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle)$, and
5. if there are z, z' such that $z' = \text{pred}^1(z)$, $\text{pred}^i(z') = v$, $\mathcal{L}(z) = \mathcal{L}(y)$, $\mathcal{L}(z') = \mathcal{L}(y')$ and $\mathcal{L}(\langle z', z \rangle) = \mathcal{L}(\langle y', y \rangle)$ then $n \leq i$.

A node x is called 1-iterated by y , denoted by $y = \mathbf{b}_1(x)$, if x is iterated by y w.r.t. \hat{x}_o . A node x is called 2-iterated by y , denoted by $y = \mathbf{b}_2(x)$, if x is iterated by y w.r.t. a 1-iterated node v . \triangleleft

As mentioned in Section 1, we do not merge explicitly nominal nodes of \mathcal{SHOIQ} -trees. Instead, we define a partition function over the set of all \mathcal{SHOIQ} -trees' nodes. This function governs partitions of nodes, S -neighbor partitions of a given partition and propagates labels between nodes and edges which should be merged. Intuitively, a S -neighbor partition of a given partition if the former contains a S -neighbor of a node of the latter. A partition function ensures that (i) each node x whose label contains the same nominal o belongs to a partition, (ii) all nodes of a partition have the same label, (iii) all non-empty edges $\langle x, y \rangle$ (i.e. its label is not empty) where x, y belong to two partitions, have the same label, and (iii) if there is a term such as $(\leq nS.C)$ belongs to the label of a partition's node then this partition has at most S -neighbor partitions. These characteristics of a partition function allow us to replace merges between nominal nodes with a partition function. The following definition provides a definition of such a partition function over the set of \mathcal{SHOIQ} -trees.

Definition 6 Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. Let $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ be a set of \mathcal{SHOIQ} -trees for $(\mathcal{T}, \mathcal{R})$ where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \hat{x}_o, \neq)$.

* A **partitioning function** $\varphi : \bigcup_{o \in \mathbf{C}_o} V_o \rightarrow 2^{\bigcup_{o \in \mathbf{C}_o} V_o}$ is defined as follows:

1. For $x, x' \in \bigcup_{o \in \mathbf{C}_o} V_o$, if $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$ and $x \neq x'$ does not hold then $\varphi(x) = \varphi(x')$;
2. For $x, x' \in \bigcup_{o \in \mathbf{C}_o} V_o$, if $x' \in \varphi(x)$ then $\varphi(x) = \varphi(x')$, and if $\varphi(x) = \varphi(x')$ then $\mathcal{L}(x) = \mathcal{L}(x')$. Therefore, we define $\mathcal{L}(\alpha) = \mathcal{L}(z)$ for $\alpha \in 2^{\bigcup_{o \in \mathbf{C}_o} V_o}$ and $z \in \alpha$. In addition, for $x, x' \in \bigcup_{o \in \mathbf{C}_o} V_o$ we define $\varphi(x) \neq \varphi(x')$ if there are two nodes y, y' such that $\varphi(x) = \varphi(y)$, $\varphi(x') = \varphi(y')$ and $y \neq_o y'$ for some $o \in \mathbf{C}_o$;
3. For $x, x', y, y' \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $\langle x, x' \rangle \in E_g$ (or $\langle x', x \rangle \in E_g$) and $\langle y, y' \rangle \in E_h$ (or $\langle y', y \rangle \in E_h$) with $g, h \in \mathbf{C}_o$, if $\varphi(x) = \varphi(y)$, $\varphi(x') = \varphi(y')$ and $\mathcal{L}'(\langle x, x' \rangle) \neq \emptyset$, $\mathcal{L}'(\langle y, y' \rangle) \neq \emptyset$ then $\mathcal{L}'(\langle x, x' \rangle) = \mathcal{L}'(\langle y, y' \rangle)$ where $\mathcal{L}'(\langle v, v' \rangle) = \mathcal{L}(\langle v, v' \rangle)$ if $\langle v, v' \rangle \in E_{o'}$ and $\mathcal{L}'(\langle v, v' \rangle) = \{S^\ominus \mid S \in \mathcal{L}(\langle v', v \rangle)\}$ if $\langle v', v \rangle \in E_{o'}$ with $o' \in \mathbf{C}_o$. Therefore, we define $\mathcal{L}(\langle \varphi(x), \varphi(x') \rangle) = \mathcal{L}'(\langle z, z' \rangle)$ with $z \in \varphi(x)$, $z' \in \varphi(x')$, $\mathcal{L}'(\langle z, z' \rangle) \neq \emptyset$ and $\langle z, z' \rangle \in E_{o'}$;

4. If $(\leq nR.C) \in \mathcal{L}(\varphi(x))$ for some $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ then there do not exist $(n + 1)$ nodes $x_0, \dots, x_n \in \bigcup_{o \in \mathbf{C}_o} V_o$ with $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and $i \neq j$ such that

- $C \in \mathcal{L}(\varphi(x_i))$, $R \in \mathcal{L}(\langle \varphi(x), \varphi(x_i) \rangle)$ for $i \in \{0, \dots, n\}$, and
- there are two nodes x_l, x_m with $l, m \in \{0, \dots, n\}$ such that $\varphi(x_l) \neq \varphi(x_m)$ does not hold.

* A node x is called *inner* if (i) x is a descendant of a 1-iterated node, and (ii) there is a node y such that y is not a descendant of a 1-iterated node and $\varphi(x) = \varphi(y)$.

* For a node x such that x is 2-iterated by y , if the two following conditions hold:

- If y has nominal descendant z then there is a node w such that (i) w is a descendant of y , w is an ancestor of z or $w = z$, and (ii) if $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', w \rangle)$ with $\langle w', w \rangle \in E_o$ then $(\leq nR.C) \notin \mathcal{L}_o(w)$ with $n \geq 0$, and
- If y has a descendant z of y such that z is i -iterated by z' then either $z' = y$ or z' is a descendant of y ,

then x is blocked by y , denoted by $y = \mathbf{b}(x)$. In addition, if there is a node z such that z is blocked by z' and z' is an ancestor of y then x is blocked by y .

* **Clashes** : A set of *SHOIQ*-trees $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ with a partitioning function φ is said to contain a clash if one of the following conditions holds:

1. There is some node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $\{A, \neg A\} \subseteq \mathcal{L}(\varphi(x))$ for some concept name $A \in \mathbf{C}$;
2. There are nodes $x \neq y$ such that $o \in \mathcal{L}(\varphi(x)) \cap \mathcal{L}(\varphi(y))$ for some $o \in \mathbf{C}_o$;
3. There is a node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ with $(\leq nR.C) \in \mathcal{L}(\varphi(x))$ and there are $(n + 1)$ nodes $x_0, \dots, x_n \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$, $\varphi(x_i) \neq \varphi(x_j)$ with $i \neq j$, and $C \in \mathcal{L}(\varphi(x_i))$, $R \in \mathcal{L}(\langle \varphi(x), \varphi(x_i) \rangle)$ for $i \in \{0, \dots, n\}$. ◁

There are two crucial notions which are exhibited in Definition 6.

• The first one is *inner nodes*. An inner node must belong to the second layer of a *SHOIQ*-tree. When a node x of the second layer becomes inner we no longer need to take care of satisfying concepts contained in its label since this is done with a node y of the first layer such that x, y belong to the same partition.

• The second one is *blocking condition* with blocking and blocked nodes located in the second layer of a *SHOIQ*-tree. The characteristics of *SHOIQ*-trees with a partition function, described by Lemma 7, allow us to avoid duplicating nominal nodes when constructing a model from these trees. The definition of blocked nodes says that if a 2-iterated node x is blocked by an ancestor y then it is necessary that any nominal descendant z of y does not have a “hard path” from y to z i.e. the label of each node y_i between z and y including z includes a term such as $(\leq n_i R_i.C_i)$ such that the predecessor y_j of y_i is a R -neighbor of y_i . This implies that we can avoid infinitely duplicating the nominal node z when constructing a model.

The *SHIQ* expansion rules described in Figure 1 are similar to those presented in (Horrocks et al., 1999b) except that they may propagate a label update between nodes of a partition or between

\sqsubseteq -rule:	if $C \sqsubseteq D \in \mathcal{T}$ and $\text{nnf}(\neg C \sqcup D) \notin \mathcal{L}(x)$ then $\mathcal{L}(x') = \mathcal{L}(x) \cup \{\text{nnf}(\neg C \sqcup D)\}$ for all $x' \in \varphi(x)$.
\sqcap -rule:	if $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x') = \mathcal{L}(x) \cup \{C_1, C_2\}$ for all $x' \in \varphi(x)$.
\sqcup -rule:	if $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x') = \mathcal{L}(x) \cup \{C\}$ with some $C \in \{C_1, C_2\}$ for all $x' \in \varphi(x)$.
\exists -rule:	if 1. $\exists S.C \in \mathcal{L}(x)$, x is not 2-iterated, and 2. x has no S -neighbour y with $C \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{S\}$, $\mathcal{L}(y) = \{C\}$ and $\varphi(y) = \{y\}$.
\forall -rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, and 2. there is a S -neighbour y of x such that $C \notin \mathcal{L}(y)$ then $\mathcal{L}(y') = \mathcal{L}(y) \cup \{C\}$ for all $y' \in \varphi(y)$.
\forall_+ -rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, and 2. there is some Q with $Q^{\oplus} \sqsubseteq S$, and 3. there is an Q -neighbour y of x such that $\forall Q^{\oplus}.C \notin \mathcal{L}(y)$ then $\mathcal{L}(y') = \mathcal{L}(y) \cup \{\forall Q^{\oplus}.C\}$ for all $y' \in \varphi(y)$.
ch -rule:	if 1. $(\leq n S.C) \in \mathcal{L}(x)$, and 2. there is an S -neighbour y of x with $\{C, \dot{C}\} \cap \mathcal{L}(y) = \emptyset$ then $\mathcal{L}(y') = \mathcal{L}(y) \cup \{E\}$ with some $E \in \{C, \dot{C}\}$ for all $y' \in \varphi(y)$.
\geq -rule:	if 1. $(\geq n S.C) \in \mathcal{L}(x)$ and x is not 2-iterated, and 2. there are no n S -neighbors y_1, \dots, y_n such that $C \in \mathcal{L}(y_i)$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$, then create n new nodes y_1, \dots, y_n with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, $\varphi(y_i) = \{y_i\}$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
\leq -rule:	if 1. $(\leq n S.C) \in \mathcal{L}(x)$, and 2. $\text{card}\{S^{\text{T}}(x, C)\} > n$ and there are two S -neighbors y, z of x with $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, y is not an ancestor of z and $y \neq z$ does not hold then 1. for all $z' \in \varphi(z)$, $x' \in \varphi(x)$ such that $\mathcal{L}'(\langle x', z' \rangle) \neq \emptyset$, if x' is an ancestor of z' then $\mathcal{L}(\langle x', z' \rangle) = \mathcal{L}'(\langle x', z' \rangle) \cup \mathcal{L}(\langle x, y \rangle)$ else $\mathcal{L}(\langle z', x' \rangle) = \mathcal{L}'(\langle z', x' \rangle) \cup \{R^{\ominus} \mid R \in \mathcal{L}(\langle x, y \rangle)\}$ 2. $\mathcal{L}(z') = \mathcal{L}(z) \cup \mathcal{L}(y)$ for all $z' \in \varphi(z)$ and $\mathcal{L}(\langle x, y \rangle) = \emptyset$ 3. add $u \neq z$ for all u such that $u \neq y$.

Figure 1: Expansion rules for $SHIQ$

NN_φ -rule:	if 1. $(\leq nR.C) \in \mathcal{L}(x)$, $\{(\leq l R.C), (\geq l R.C)\} \not\subseteq \mathcal{L}(x)$ for all l , 2. $(\leq k R.C) \notin \mathcal{L}(x)$ for all $k < n$, and 3. x has a R -neighbor y such that $C \in \mathcal{L}(y)$ then 1. guess m with $1 \leq m \leq n$, and 2. $\mathcal{L}(x') = \mathcal{L}(x) \cup \{\leq mR.C, \geq mR.C\}$ for all $x' \in \varphi(x)$.
o_φ -rule:	if 1. there are nodes x, x' with $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$, 2. $\varphi(x) \cap \varphi(x') = \emptyset$ and $\varphi(x) \neq \varphi(x')$ does not hold, then 1. $\varphi(y') = \varphi(x) \cup \varphi(x')$ for all $y' \in \varphi(x) \cup \varphi(x')$, 2. $\mathcal{L}(y') = \mathcal{L}(x) \cup \mathcal{L}(x')$ for all $y' \in \varphi(x) \cup \varphi(x')$, 3. for all $z, z' \in \varphi(x) \cup \varphi(x')$ and for all $w, w' \in \varphi(y)$ with some y , if w is a S -neighbor of z , w' is not a S -neighbor of z' , $\mathcal{L}'(\langle z', w' \rangle) \neq \emptyset$ then $\mathcal{L}(\langle z', w' \rangle) = \mathcal{L}(\langle z', w' \rangle) \cup \{S\}$ if w' is a successor of z' , or $\mathcal{L}(\langle w', z' \rangle) = \mathcal{L}(\langle w', z' \rangle) \cup \{S^\ominus\}$ if z' is a successor of w' .
\leq_φ -rule:	if 1. $(\leq nR.C) \in \mathcal{L}(\varphi(x))$, there are $x', x'' \in \varphi(x)$ with $x' \neq x''$, and 2. there are nodes y_0, \dots, y_n with $\varphi(y_i) \cap \varphi(y_j) = \emptyset$, $0 \leq i < j \leq n$, $C \in \mathcal{L}(\varphi(y_i))$, $R \in \mathcal{L}(\langle \varphi(x), \varphi(y_i) \rangle)$ for all $0 \leq i \leq n$, and 3. there are $x', x'' \in \varphi(x)$ with $x' \neq x''$, and x' has a R -neighbor y' , x'' has a R -neighbor y'' such that $C \in \mathcal{L}(y') \cap \mathcal{L}(y'')$, $\varphi(y') \cap \varphi(y'') = \emptyset$, and $\varphi(y') \neq \varphi(y'')$ does not hold then 1. $\varphi(y) = \varphi(y') \cup \varphi(y'')$ for all $y \in \varphi(y') \cup \varphi(y'')$, 2. $\mathcal{L}(y) = \mathcal{L}(y') \cup \mathcal{L}(y'')$ for all $y \in \varphi(y') \cup \varphi(y'')$, 3. for all $w, w' \in \varphi(x)$ and $z, z' \in \varphi(y') \cup \varphi(y'')$, if z is a S -neighbor of w , z' is not a S -neighbor of w' , $\mathcal{L}'(\langle w', z' \rangle) \neq \emptyset$ then $\mathcal{L}(\langle w', z' \rangle) = \mathcal{L}(\langle w', z' \rangle) \cup \{S\}$ if z' is a successor of w' , or $\mathcal{L}(\langle z', w' \rangle) = \mathcal{L}(\langle z', w' \rangle) \cup \{S^\ominus\}$ if w' is a successor of z' .

Figure 2: New expansion rules for $SHOIQ$

edges connecting two neighbor partitions. In particular, \exists -rule and \geq -rule may add new nodes and extend the partition function φ to the added nodes. Notice that these rules consider a 2-iterated node as leaf node i.e. no successor of a 2-iterated node will be added unless it is no longer a 2-iterated node.

However, the $SHOIQ$ expansion rules described in Figure 2 (except for NN_φ -rule) perform on the partition function φ necessary updates triggered by merging partitions. We adopt a strategy of rule applications as follows:

The NN_φ -rule and all $SHIQ$ -rules are applied with higher priority.

The precondition “ $\{(\leq l R.C), (\geq l R.C)\} \not\subseteq \mathcal{L}(x)$ for all l in NN_φ -rule ensures that if $(\leq nR.C), (\leq n'R.C) \in \mathcal{L}(x)$ with $n < n'$ then this rule is applied at most once with $(\leq nR.C)$.”

Lemma 7 presents the most important characteristics of $SHOIQ$ -trees with a partition tree φ . They allow us to use the partitioning function φ to govern all fusions of nodes and edges triggered by the necessity of merging nominal nodes instead of performing explicitly these fusions on $SHOIQ$ -trees. One of these characteristics says that if a node x does not have any inner ancestor then x is blocked. This simplifies dramatically termination condition for the tableaux-based algorithm.

Input : A \mathcal{SHOIQ} knowledge base $(\mathcal{T}, \mathcal{R})$

Output: Is $(\mathcal{T}, \mathcal{R})$ consistent ?

- 1 For each $o \in \mathbf{C}_o$, let $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \hat{x}_o, \neq)$ be an initial tree such that $V_o = \{\hat{x}_o\}$, $\mathcal{L}(\hat{x}_o) = \{o\}$, and there is no $x, y \in V_o$ such that $x \neq y$;
- 2 A partitioning function φ is initialized with $\varphi(\hat{x}_o) = \{\hat{x}_o\}$;
- 3 **while** there is a non-empty set S of expansion rules in Figure 1 and 2 such that each $r \in S$ can be applied to a node $x \in V_o$ **do**
- 4 Apply r to x ;
- 5 **if** there is a clash-free set of trees $\{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ which is built by Line 1 to 4 **then**
- 6 **YES** ;
- 7 **else**
- 8 **NO** ;

Algorithm 1: Algorithm for checking consistency of a \mathcal{SHOIQ} knowledge base

Lemma 7 Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. Let $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ be a set of \mathcal{SHOIQ} -trees with a function φ which are built by Algorithm 1 where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \hat{x}_o, \neq)$. If \mathbf{T} is clash-free then

1. φ is a partitioning function.
2. For two nodes non-nominal $x, y \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $x \neq y$ and $y \in \varphi(x)$, there are nodes $x_0, \dots, x_k = x$ and y_0, \dots, y_k with $x_k = x, y_k = y$ and $x_i \neq y_i$ for $0 \leq i \leq k$ such that $\varphi(x_i) = \varphi(y_i)$, $\mathcal{L}'(\langle x_j, x_{j+1} \rangle) = \mathcal{L}'(\langle y_j, y_{j+1} \rangle)$, $(\leq n_j R_j.C_j) \in \mathcal{L}(x_j)$, $C_j \in \mathcal{L}(x_{j+1})$, $\mathcal{L}(\langle x_j, x_{j+1} \rangle)$ for all $0 \leq i \leq k, 0 \leq j < k$, and $o \in \mathcal{L}(x_0)$ for some $o \in \mathbf{C}_o$.
3. For each descendant $y \in \bigcup_{o \in \mathbf{C}_o} V_o$ of a 1-iterated node such that y has a nominal descendant z , y is inner if for each descendant w of y such that w is an ancestor of z or $w = z$, there is $(\leq nR.C) \in \mathcal{L}(w)$ with $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', w \rangle)$, $\langle w', w \rangle \in E_o$ for some $o \in \mathbf{C}_o$.
4. For each node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$, if x is 2-iterated and not blocked then x has an inner ancestor.

Let us point out why we need the properties of \mathcal{SHOIQ} -trees formulated in Lemma 7. Property 1 affirms that a partitioning function φ can be maintained by applications of the expansion rules. Property 2 says that grouping any two distinct non-nominal nodes into a partition results from grouping two noninal nodes. This result is necessary to show that if $x \in \varphi(y)$ and y is blocked then x must be a 2-iterated and $\mathcal{L}(x') = \mathcal{L}(y')$, $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle)$ where x', y' are the predecessors of x, y respectively. This allows us to use a unique blocking node for all nodes $w \in \varphi(y)$. We refer the readers to the proof of Lemma 9 for more details. Property 3 affirms that if a node y located in the second layer of a \mathcal{SHOIQ} -tree and y has a “hard path” leading to a nominal descendant then y is inner. This property is obtained thanks to NN_φ -rule, \geq -rule, \leq -rule and the characteristics of φ . They impose that if $x \in \varphi(y)$ and $(\leq nS.C) \in \mathcal{L}(x) = \mathcal{L}(y)$ then NN_φ -rule guesses some $m \leq n$ and adds $(\leq mS.C)$ to $\mathcal{L}(x)$ and $\mathcal{L}(y)$. In turn, \geq -rule and \leq -rule generate exactly m S -neighbors x_i of x containing C , and m S -neighbors y_j of y containing C as well. Properties 3 and 4 ensure that each node located in the second layer of a \mathcal{SHOIQ} -tree either belongs a partition with a node located in the first layer, or is blocked. This is crucial for soundness of the tableaux-based algorithm.

Proof:[Proof of Lemma 7]

1. o_φ -rule ensures that if $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and $x \neq x'$ does not hold then $\varphi(x) = \varphi(x')$ i.e. the condition 1 in Definition 6 holds. Since \exists -rule and \geq -rule extend φ by adding new nodes or edges, it is obvious that the conditions 1 and 2 in Definition 6 hold. In addition, applications of \sqsubseteq -rule, \sqcap -rule, \sqcup -rule, \forall -rule, \forall_+ -rule, ch -rule, NN_φ -rule to a node x change the label of nodes and propagate each added concept to the label of all nodes of a partition $\varphi(x)$. \leq -rule changes the label of nodes and edges. In particular, it sets the label of an edge to empty and does not propagate the emptiness to the other edges that connect the two related partitions. Therefore, the conditions 2 and 3 in Definition 6 hold.

Applications of o_φ -rule and \leq_φ -rule to a node x change the label of nodes and propagate each added concept to the label of all nodes of a partition $\varphi(x)$. In addition, they propagate also each role from an edge $\langle x, y \rangle$ to the label of all edges that connects a node $x' \in \varphi(x)$ to a node $y' \in \varphi(y)$. This implies that the conditions 2 and 3 in Definition 6 hold.

Finally, \leq_φ -rule ensures that the condition 4 in Definition 6 holds.

2. Since $x \neq y$ are non-nominal and $x \in \varphi(y)$, x, y must be grouped by applying \leq_φ -rule to a node z such that $x_1 \neq y_1 \in \varphi(z)$, $\mathcal{L}'(\langle x, x_1 \rangle) = \mathcal{L}'(\langle y, y_1 \rangle)$, $(\leq n_0 R_0.C_0) \in \mathcal{L}(x)$, $C_0 \in \mathcal{L}(x_1)$. According to Lemma 8, each \mathbf{T}_o is built by a finite number of applications of expansion rules and a first merge is started by the application of o_φ -rule. By induction, it follows that there are nodes $x_0, \dots, x_k = x$ and $y_0, \dots, y_k = y$ with $x_i \neq y_i$ for $0 \leq i \leq k$ such that $\varphi(x_i) = \varphi(y_i)$, $\mathcal{L}'(\langle x_j, x_{j+1} \rangle) = \mathcal{L}'(\langle y_j, y_{j+1} \rangle)$, $(\leq n_j R_j.C_j) \in \mathcal{L}(x_j)$, $C_j \in \mathcal{L}(x_{j+1})$, $\mathcal{L}(\langle x_j, x_{j+1} \rangle)$ for all $0 \leq i \leq k$, $0 \leq j < k$, and $o \in \mathcal{L}(x_0)$ for some $o \in \mathbf{C}_o$.
3. By the construction, we have $\varphi(\hat{x}_o) = \varphi(z)$ with $o \in \mathcal{L}(z)$, and thus z is inner. Assume that nodes $w_1 = z, \dots, w_k = w$ between z and w (i.e. w_{i+1} is the predecessor of w_i) are inner with $\varphi(w_i) = \varphi(y_i)$, $\mathcal{L}'(\langle w_j, w_{j+1} \rangle) = \mathcal{L}'(\langle y_j, y_{j+1} \rangle)$ where y_1 is a neighbor of \hat{x}_o , y_{j+1} is a neighbor of y_j for all $0 < i \leq k$, $0 < j < k$. By the definition of 1-iterated and 2-iterated nodes, y is a descendant of a 1-iterated node v and y_i is an ancestor of a 1-iterated node for all $1 \leq i \leq k$. This implies that the predecessor w' of w is a descendant of v . Since $(\leq n R.C) \in \mathcal{L}(w)$ and $\mathcal{L}(w) = \mathcal{L}(y_k)$, y_k has $m \leq n$ R -neighbors v_1, \dots, v_m such that $C \in \mathcal{L}(v_i)$ and w has m R -neighbors u_1, \dots, u_m such that $C \in \mathcal{L}(u_i)$ for $i \in \{1, \dots, m\}$ due to NN_φ -rule and \geq -rule. This implies that $w' \in \{u_1, \dots, u_m\}$ since $C \in \mathcal{L}(w')$, $R^\ominus \in \mathcal{L}(\langle w', w \rangle)$ and the non-applicable of \leq -rule (due to clash-freeness). Due to the non-applicable of \leq_φ -rule there is some $l \in \{1, \dots, m\}$ such that $\varphi(v_l) = \varphi(w')$. Since y_k is an ancestor of a 1-iterated, v_l must be either a 1-iterated node or an ancestor of a 1-iterated node. That means that w' is inner. Thus, y is inner.
4. Let x be a node that is 2-iterated by y . Assume that x is not blocked. According to the definition of blocked nodes, there are the two following possibilities:
 - (a) y has a nominal descendant z such that for each ancestor w of z or $w = z$, that is a descendant of y , there is $(\leq n R.C) \in \mathcal{L}(w)$ with $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', w \rangle)$, $\langle w', w \rangle \in E_o$ for some $o \in \mathbf{C}_o$. By Item 3, we obtain y is inner.
 - (b) y has a descendant y' that is 2-iterated by an ancestor z' of y and z' has a nominal descendant z such that for each node w which is a descendant of z' , and w is an ancestor

of z' or $w = z'$, there is $(\leq nR.C) \in \mathcal{L}(w)$ with $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', w \rangle)$, $\langle w', w \rangle \in E_o$ for some $o \in \mathbf{C}_o$. By Item 3, we obtain z' is inner. □

Lemma 8 (Termination) *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. Algorithm 1 terminates.*

Proof: First, we compute an upper bound of the \mathcal{SHOIQ} tree's height from the condition for 1-iteration and 2-iteration. This upper bound equals $K = 2^{2m+k} \times 2$ where $m = \text{card}\{\text{cl}(\mathcal{T}, \mathcal{R})\}$, and k is the number of roles occurring in \mathcal{T} and \mathcal{R} plus their inverse. Moreover, the number of neighbors of any node is bounded by $M = \sum m_i$ where m_i occurs in a number restriction term $(\geq m_i R.C)$ that appears in \mathcal{T} .

Applications of rules in Figure 1 and 2 do not remove concepts from the label of nodes. Moreover, applications of these rules do not remove roles from the label of edges except that they may set the label of edges to an empty set. In fact, when \leq -rule is applied to a node x to merge the label of a successor y of x to that of a node z , and the label of $\langle x, z \rangle$ to that of $\langle x, z \rangle$ such that relation \neq is preserved i.e. this does not lead to apply \geq -rule to x more once. If y is a S -neighbor of x created by an application of \geq -rule then y is now replaced with z . When this rule makes the label of an edge become empty it remains to be empty forever. Notice that x cannot be created by applying \geq -rule to y since y is a successor of x .

In addition, o_φ -rule and \leq_φ -rule can be triggered by applying the other rules but this makes each partition of nodes growing only. Therefore, they cannot be applied infinitely. □

Lemma 9 (Soundness) *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. If Algorithm 1 can build a clash-free set of \mathcal{SHOIQ} -trees for $(\mathcal{T}, \mathcal{R})$ then there is a tableau for $(\mathcal{T}, \mathcal{R})$.*

Proof: Assume that $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ is a set of clash-free \mathcal{SHOIQ} -trees for $(\mathcal{T}, \mathcal{R})$ with a partitioning function φ where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \hat{x}_o, \neq_o)$. First, we define an extended function $\widehat{\text{succ}}$ from succ over each \mathbf{T}_o as follows:

- if x has a successor y (resp. x has a R -successor y) that is not blocked then $y \in \widehat{\text{succ}}^1(x)$ (resp. $y \in \widehat{\text{succ}}_R^1(x)$),
- if x has a successor z (resp. x has a R -successor z) that is blocked by $b(z)$ then $b(z) \in \widehat{\text{succ}}^1(x)$ (resp. $b(z) \in \widehat{\text{succ}}_R^1(x)$).
- if $y \in \widehat{\text{succ}}_R^n(x)$ and $z \in \widehat{\text{succ}}_R^1(y)$ then $z \in \widehat{\text{succ}}_R^{(n+1)}(x)$ for $n \geq 0$.

For each blocking node y of a blocked node x , if y has a descendant nominal z then, according to the definition of blocked nodes and Lemma 7, y has a descendant y_z such that (i) y_z is not an ancestor of x , (ii) $y_z = z$ or y_z is an ancestor of z , and (iii) if $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', y_z \rangle)$ with $\langle w', y_z \rangle \in E_o$, $o \in \mathbf{C}_o$ then $(\leq nR.C) \notin \mathcal{L}(y_z)$, and (iv) y_z cannot be an ancestor of any blocked node (*).

In addition, we show that if $w \in \varphi(y)$ and y is blocked then w is a 2-iterated and $\mathcal{L}'(\langle w', w \rangle) = \mathcal{L}'(\langle y', y \rangle)$ where w', y' are the predecessor of w and y respectively (**).

In fact, according to Lemma 7, there are nodes $w_0, \dots, w_l = w$ and $y_0, \dots, y_l = y$ such that $\varphi(w_i) = \varphi(y_i)$, $\mathcal{L}'(\langle w_j, y_{j+1} \rangle) = \mathcal{L}'(\langle w_j, y_{j+1} \rangle)$, $(\leq n_j R_j.C_j) \in \mathcal{L}(w_j)$, $C_j \in \mathcal{L}(w_{j+1})$,

$\mathcal{L}(\langle w_j, w_{j+1} \rangle)$ for all $0 \leq i \leq l, 0 \leq j < l$, and $o \in \mathcal{L}(w_0) = \mathcal{L}(y_0)$ for some $o \in \mathbf{C}_o$. Since y is blocked by $\mathbf{b}(y)$, such a nominal node y_0 must be an ancestor of $\mathbf{b}(y)$ or a descendant of an ancestor of $\mathbf{b}(y)$. This implies that the path from w_0 to w_l have at least two edges whose labels are identical. By the construction of trees with the blocking condition, w_l must be either inner or 2-iterated. It is not possible that w_l is inner since $y \in \varphi(w_l)$ and y is blocked. This implies that w_l is 2-iterated and $w_{l-1} \in \varphi(y_{l-1})$ where $w_{l-1} = w'$ and $y_{l-1} = y'$ and $\mathcal{L}'(\langle w_{l-1}, w_l \rangle) = \mathcal{L}'(\langle y_{l-1}, y \rangle)$.

For each $\varphi(x)$ such that there is some $y \in \varphi(x)$ that is blocked by $\mathbf{b}(y)$, we define an infinite set $\widehat{V}_o^{\varphi(x)}$, a set $\widehat{\text{succ}}^1(v)$ of successors of some $v \in \widehat{V}_o^{\varphi(x)}$ and a function \mathbf{p} from $\widehat{V}_o^{\varphi(x)}$ to V_o as follows:

- We add to $\widehat{V}_o^{\varphi(x)}$ a node $\widehat{v}_o^{\varphi(x)}$ with $\widehat{v}_o^{\varphi(x)} \in \widehat{\text{succ}}^1(x)$, and define $\mathbf{p}(\widehat{v}_o^{\varphi(x)}) = \mathbf{b}(y)$;
- For each $x' \in \widehat{\text{succ}}^1(\mathbf{p}(v))$ with $v \in \widehat{V}_o^{\varphi(x)}$,
 - If x' is not blocked, x' has no nominal descendant then we add to $\widehat{V}_o^{\varphi(x)}$ a successor v' of v , i.e. $v' \in \widehat{\text{succ}}^1(v)$ with $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle x, x' \rangle)$, and define $\mathbf{p}(v') = x'$;
 - If x' is not blocked, x' has a nominal descendant and x' is an ancestor of the node w_y determined by (*), then we add to $\widehat{V}_o^{\varphi(x)}$ a successor v' of v , i.e. $v' \in \widehat{\text{succ}}^1(v)$ with $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle x, x' \rangle)$, and define $\mathbf{p}(v') = x'$;
 - If x' is blocked by $\mathbf{b}(x)$, then we add to $\widehat{V}_o^{\varphi(x)}$ a successor v' of v , i.e. $v' \in \widehat{\text{succ}}^1(v)$ with $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle x, \mathbf{b}(x) \rangle)$, and define $\mathbf{p}(v') = \mathbf{b}(x)$;

Note that if x' has a nominal descendant and $x' = y_z$ or x' is a descendant of the node y_z determined by (*), then we add no node to \widehat{V}_o . In addition, according to (**), $\widehat{v}_o^{\varphi(x)}$ is defined once for all $x' \in \varphi(x)$. We define a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ as follows:

- $\mathbf{S} = \mathbf{S}_1 \cup \mathbf{S}_2$ where
 - $\mathbf{S}_1 = \{\varphi(x) \mid x \text{ is not blocked, } x \text{ has no inner ancestor}\}$, and $\mathcal{L}'(\varphi(x)) = \mathcal{L}(x)$ for $\varphi(x) \in \mathbf{S}_1$;
 - $\mathbf{S}_2 = \bigcup_{x \text{ is blocked, } o \in \mathbf{C}_o} \widehat{V}_o^{\varphi(x)}$, and $\mathcal{L}'(v) = \mathcal{L}(\mathbf{p}(v))$ for $v \in \widehat{V}_o^{\varphi(x)}$.
- $\mathcal{E}(R) = \mathcal{E}_1(R) \cup \mathcal{E}_2(R) \cup \mathcal{E}_3(R) \cup \mathcal{E}_4(R)$ where
 - $\mathcal{E}_1(R) = \{\langle \varphi(x), \varphi(y) \rangle \in \mathbf{S}_1 \times \mathbf{S}_1 \mid R \in \mathcal{L}(\langle \varphi(x), \varphi(y) \rangle) \vee R^\ominus \in \mathcal{L}(\langle \varphi(y), \varphi(x) \rangle)\}$,
 - $\mathcal{E}_2(R) = \{\langle s, t \rangle \in \mathbf{S}_2 \times \mathbf{S}_2 \mid R \in \mathcal{L}(\langle s, t \rangle) \vee R^\ominus \in \mathcal{L}(\langle t, s \rangle)\}$
 - $\mathcal{E}_3(R) = \{\langle \varphi(x'), \widehat{v}_o^{\varphi(x)} \rangle \in \mathbf{S}_1 \times \mathbf{S}_2 \mid R \in \mathcal{L}(\langle x', x \rangle) \vee R^\ominus \in \mathcal{L}(\langle x, x' \rangle)\}$
 - $\mathcal{E}_4(R) = \{\langle s, \varphi(y_z) \rangle \in \mathbf{S}_2 \times \mathbf{S}_1 \mid R \in \mathcal{L}(\langle \mathbf{p}(s), y_z \rangle) \vee R^\ominus \in \mathcal{L}(\langle y_z, \mathbf{p}(s) \rangle)\}$

We now show that T satisfies all properties in Definition 3.

- P1-P4 hold due to the non-applicable of \sqsubseteq -rule, \sqcap -rule and \sqcup -rule in Figure 1 and the facts that \mathbf{T} is clash-free.
- For P6, assume that $s, t \in \mathbf{S}$ with $\forall S.C \in \mathcal{L}'(s), Q^\oplus \boxtimes S$ and $\langle s, t \rangle \in \mathcal{E}(Q)$. By the definition of T and \mathbf{T} , we consider the following cases:

- $s \in \mathbf{S}_1$ and $t \in \mathbf{S}_1$. This implies that there are x, y such that x, y have no inner ancestor, $s = \varphi(x), t = \varphi(y)$ and $\forall S.C \in \mathcal{L}(x), Q \in \mathcal{L}'(\langle x, y \rangle)$. Due to the non-applicable of \forall_+ -rule, it follows that $\forall Q^\oplus.C \in \mathcal{L}(y)$. By the definition of T and \mathbf{T} , we have $\forall Q^\oplus.C \in \mathcal{L}'(t)$.
 - $s \in \mathbf{S}_2$ and $t \in \mathbf{S}_2$. This implies that $\forall S.C \in \mathcal{L}(\mathfrak{p}(s)), Q \in \mathcal{L}'(\langle \mathfrak{p}(s), \mathfrak{p}(t) \rangle)$. Due to the non-applicable of \forall_+ -rule, it follows that $\forall Q^\oplus.C \in \mathcal{L}(\mathfrak{p}(t))$. By the definition of T , we have $\forall Q^\oplus.C \in \mathcal{L}'(t)$.
 - $s \in \mathbf{S}_1$ and $t \in \mathbf{S}_2$. This implies that either (i) there are x, y such that (i) either $s = \varphi(x), \widehat{v}_o^{\varphi(y)} \in \widehat{V}_o^{\varphi(y)}$ with some $o \in \mathbf{C}_o$ and $\forall S.C \in \mathcal{L}(x), Q \in \mathcal{L}'(\langle x, y \rangle)$, or (ii) $s = \varphi(x), \mathfrak{p}(t) \in \widehat{v}_o^{\varphi(y)}$ with some $o \in \mathbf{C}_o$ and $\forall S.C \in \mathcal{L}(x), Q \in \mathcal{L}'(\langle x, \mathfrak{p}(t) \rangle)$. Due to the non-applicable of \forall_+ -rule, it follows that $\forall Q^\oplus.C \in \mathcal{L}(y)$ or $\forall Q^\oplus.C \in \mathcal{L}(\mathfrak{p}(t))$. By the definition of T , we have $\forall Q^\oplus.C \in \mathcal{L}'(t)$.
 - $s \in \mathbf{S}_2$ and $t \in \mathbf{S}_1$. Analogously.
- P5. Analogously.
 - P7 holds due to the non-applicable of \exists -rule and the construction of T .
 - For P10, assume that $s \in \mathbf{S}$ with $(\leq n'S.C) \in \mathcal{L}'(s)$. We consider the following cases:
 - $s \in \mathbf{S}_1$ i.e. $s = \varphi(x)$ for some $x \in V_o$ where x has no inner ancestor and $o \in \mathbf{C}_o$. Assume that x is not 2-iterated. Due to the non-applicable of NN_φ -rule, \leq -rule, \geq -rule and the clash-freeness, x has $n \leq n'$ nodes x_1, \dots, x_n such that $x_i \neq x_j, x_i$ is S -neighbor of x with $C \in \mathcal{L}(x_i)$ for $1 \leq i < j \leq n$. Due to the definition of φ we have $\varphi(x_i) \neq \varphi(x_j)$ for $1 \leq i < j \leq n$. This implies that \leq_φ -rule is never applied to $\varphi(x_1), \dots, \varphi(x_n)$. According to the definition of \mathbf{S}_2 , it is not possible that there is some $v \in \widehat{V}_o^{\varphi(y)}$ such that $\mathfrak{p}(v) = x_i, i \in \{1, \dots, n\}$ since $(\leq nS.C) \in \mathcal{L}(x), C \in \mathcal{L}(\mathfrak{p}(x_i)), S \in \mathcal{L}'(\langle \mathfrak{p}(x_i), x \rangle)$. Assume that there is a x_i with $i \in \{1, \dots, n\}$ and a blocked node y such that $x_i \in \varphi(y)$. According to (**), for any node $w' \in \varphi(x)$, if a node w is a neighbor of w' and $w \in \varphi(y)$ then w' is a 2-iterated node and $\mathcal{L}'(\langle w', w \rangle) = \mathcal{L}'(\langle y', y \rangle)$ where w', y' are the predecessor of w and y respectively. This implies that $\varphi(x_i)$ is replaced with $\widehat{v}_o^{\varphi(y)}$, and $\varphi(x)$ has exactly n distinct elements $s_1, \dots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}'(s_i)$ and $S \in \mathcal{E}(\langle \varphi(x), s_i \rangle)$. Assume that there is a x_i with $i \in \{1, \dots, n\}$ such that $x_i = \mathfrak{p}(v)$ with $v \in \widehat{V}_o^{\varphi(z)}$ for some blocked node z and $o \in \mathbf{C}_o$. By the definition of $\widehat{V}_o^{\varphi(z)}$, $(\leq nSC) \notin \mathcal{L}(x)$, which is a contradiction. Assume that x is 2-iterated. According to Lemma 7, x is blocked and $\varphi(x) \notin \mathbf{S}_1$.
 - $s \in \mathbf{S}_2$ i.e. $s \in \widehat{V}_o^{\varphi(y)}$ for some blocked node y with $o \in \mathbf{C}_o$. P10 holds due to the definition of $\widehat{V}_o^{\varphi(y)}$ and the non-applicable of \leq -rule.
 - To show P11, assume that $s \in \mathbf{S}_1$ i.e. $s = \varphi(x)$ for some $x \in V_o$ where x has no inner ancestor and $o \in \mathbf{C}_o$. Due to the non-applicable of NN_φ -rule, \geq -rule and the clash-freeness, x has n nodes x_1, \dots, x_n such that $x_i \neq x_j, x_i$ is S -neighbor of x with $C \in \mathcal{L}(x_i)$ for $1 \leq i <$

$j \leq n$. If there is a blocked node $y \in \varphi(x_i)$ then $\varphi(x_i)$ is replaced with $\widehat{v}_o^{\varphi(y)}$. Therefore, $\varphi(x)$ has at least n distinct elements $s_1, \dots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}'(s_i)$ and $S \in \mathcal{E}(\langle \varphi(x), s_i \rangle)$.

Assume that $s \in \mathbf{S}_2$ i.e. $s \in \widehat{V}_o^{\varphi(y)}$ for some blocked node y with $o \in \mathbf{C}_o$. P11 holds due to the definition of $\widehat{V}_o^{\varphi(y)}$ and the non-applicable of \geq -rule.

- For P12 assume that $s, t \in \mathbf{S}$ with $(\leq nS.C) \in \mathcal{L}'(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$. By the definition of T , x_t is a S -neighbor of x_s and $(\leq nS.C) \in \mathcal{L}(x_s)$ where $x_s, x_t \in \varphi(x) \cup \varphi(y) \cup \{\mathbf{p}(s), \mathbf{p}(t)\}$, $\varphi(x), \varphi(y) \in \mathbf{S}_1$ and $\{\mathbf{p}(s), \mathbf{p}(t)\} \subseteq \widehat{V}_o^{\varphi(z)} \in \mathbf{S}_2$. Due to the non-applicable of ch -rule, we have $C \in \mathcal{L}(x_t)$ or $\dot{C} \in \mathcal{L}(x_t)$. Due to the definition of T , this implies $C \in \mathcal{L}'(t)$ or $\dot{C} \in \mathcal{L}'(t)$.
- P13 holds due to the non-applicable of o_φ -rule.
- P14 holds due to the construction of $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$.
- P15 holds due to the non-applicable of NN_φ -rule.

□

Lemma 10 (Completeness) *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then Algorithm 1 can build a clash-free set of \mathcal{SHOIQ} -trees for $(\mathcal{T}, \mathcal{R})$.*

Proof: Let $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ be a tableau for $(\mathcal{T}, \mathcal{R})$. Let $\{\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \widehat{x}_o, \neq) \mid o \in \mathbf{C}_o\}$ be a set of \mathcal{SHOIQ} -trees. We show that there exists a sequence of expansion rule applications such that it generates a clash-free \mathcal{SHOIQ} -trees with a partitioning function φ (**).

We maintain functions π from $\bigcup_{o \in \mathbf{C}_o} V_o$ to \mathbf{S} such that they satisfy the following condition, denoted by (*):

- (C1) $\mathcal{L}(x') \subseteq \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$,
- (C2) If y' is a S -neighbor of x' for some $x' \in \varphi(x)$ and $y' \in \varphi(y)$ then $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S)$,
- (C3) $x \neq y$ implies $\pi(x) \neq \pi(y)$,
- (C4) $y, y' \in \varphi(x)$ implies $\pi(y) = \pi(y')$.

To prove (**), we have to show that (i) we can apply expansion rules such that the conditions in (*) are preserved, and (ii) if the conditions (*) are satisfied when constructing \mathcal{SHOIQ} trees by expansion rules then the obtained \mathcal{SHOIQ} trees are clash-free.

According to P14 in Definition 3, for each $o \in \mathbf{C}_o$ there is some $s_o \in \mathbf{S}$ such that $o \in \mathcal{L}'(s_o)$. We initialized $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$ and $\pi(\widehat{x}_o) = s_o$ for each $o \in \mathbf{C}_o$.

1. \sqsubseteq -rule. Due to P1, we have $\text{nnf}(\neg C \sqcup D) \in \mathcal{L}'(\pi(x))$ for all x . Therefore, if $\text{nnf}(\neg C \sqcup D) \in \mathcal{L}(x')$ for each $x' \in \varphi(x)$ then $\text{nnf}(\neg C \sqcup D) \in \mathcal{L}'(\pi(x))$. Moreover, \sqsubseteq -rule adds $\text{nnf}(\neg C \sqcup D)$ to $\mathcal{L}(x')$ for all $x' \in \varphi(x)$ i.e. (C1) is preserved. Since (C2) and (C3) are preserved trivially, (*) holds if \sqsubseteq -rule is applied.

2. \sqcap -rule. Due to P3, if $(C \sqcap D) \in \mathcal{L}'(\pi(x))$ then $\{C, D\} \subseteq \mathcal{L}'(\pi(x))$. Due to (*) if $(C \sqcap D) \in \mathcal{L}(x')$ then $(C \sqcap D) \in \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$. Moreover, \sqcap -rule adds C, D to $\mathcal{L}(x')$ for all $x' \in \varphi(x)$ if $(C \sqcap D) \in \mathcal{L}(x')$ i.e. (C1) is preserved. Since (C2) – (C4) are preserved trivially, (*) holds if \sqcap -rule is applied.
3. \sqcup -rule. Due to P4, if $(C \sqcup D) \in \mathcal{L}'(\pi(x))$ then $\{C, D\} \cap \mathcal{L}'(\pi(x)) \neq \emptyset$. Due to (*) if $(C \sqcup D) \in \mathcal{L}(x')$ then $(C \sqcup D) \in \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$. Therefore, \sqcup -rule can add a concept $E \in \{C, D\}$ to $\mathcal{L}(x')$ such that $\mathcal{L}(x') \subseteq \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$ i.e. (C1) is preserved. Since (C2) – (C4) are preserved trivially, (*) holds if \sqcup -rule is applied.
4. Analogously, we can show \forall -rule, \forall_+ , ch -rule can be applied such that (*) holds.
5. \exists -rule. If $\exists S.C \in \mathcal{L}(x)$ then $\exists S.C \in \mathcal{L}(x')$ for all $x' \in \varphi(x)$ and $\exists S.C \in \mathcal{L}'(\pi(x))$. Due to P7, if $\exists S.C \in \mathcal{L}'(\pi(x))$ there is an individual t such that $\langle \pi(x), t \rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}'(t)$. The application of \exists -rule create a new node y such that $\mathcal{L}(y) = \{C\}$ and $\mathcal{L}(\langle x, y \rangle) = \{S\}$. We set $\pi(y) = t$ and $\varphi(y) = \{y\}$. This implies that (C2) – (C4) are preserved. Thus, (*) holds if \exists -rule is applied.
6. Analogously, we can show \geq -rule can be applied such that (*) holds.
7. \leq -rule. If $(\leq nS.C) \in \mathcal{L}(x)$ then $(\leq nS.C) \in \mathcal{L}(x')$ for all $x' \in \varphi(x)$ and $(\leq nS.C) \in \mathcal{L}'(\pi(x))$. Due to P10, $\text{card}\{S^T(\pi(x), C)\} \leq n$. For some $x' \in \varphi(x)$, \leq -rule is applicable if x' has $(n+1)$ S -neighbors y_0, \dots, y_n with $C \in \mathcal{L}(y_i)$. This implies that there are two nodes $y, z \in \{y_0, \dots, y_n\}$ such that $\pi(y) = \pi(z)$, and thus $y \neq z$ cannot hold due to (*). Therefore, y, z can be chosen to merge by \leq -rule where y is a successor of x' . It is obvious that (C1) is preserved. If y was a R -neighbor of x then z is now a R -neighbor of x and $\langle \pi(x), \pi(z) \rangle \in \mathcal{E}(R)$ due to $\pi(y) = \pi(z)$ and (*). This implies (C1) – (C2) hold. Moreover, (C3) is preserved since $\varphi(y) \neq \varphi(z)$ does not hold. Due to (*) and $\pi(y) = \pi(z)$, (C4) is preserved as well.
8. For NN_φ -rule, assume that $(\leq nS.C) \in \mathcal{L}(x)$ with $n > 0$. If $(\leq nS.C) \in \mathcal{L}(x)$ then $(\leq nS.C) \in \mathcal{L}(x')$ for all $x' \in \varphi(x)$ and $(\leq nS.C) \in \mathcal{L}'(\pi(x))$. By P15, we have $\{\leq mS.C, \geq mS.C\} \subseteq \mathcal{L}'(\pi(x))$ with $1 \leq m \leq n$. This implies that NN_φ -rule can be applied such that (*) holds.
9. For o_φ -rule, assume that there are x, y such that $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and $\varphi(x) \neq \varphi(y)$ does not hold. We have $o \in \mathcal{L}'(\pi(x)) \cap \mathcal{L}'(\pi(y))$ due to (C1). This implies that $\pi(x) = \pi(y)$ due to P13. Thus, $\mathcal{L}'(\pi(x)) = \mathcal{L}'(\pi(y))$. Moreover, it follows that $\mathcal{L}(x) \subseteq \mathcal{L}'(\pi(x))$ and $\mathcal{L}(y) \subseteq \mathcal{L}'(\pi(x))$ due to (C1). This implies that $\mathcal{L}(x') \cup \mathcal{L}(y') \subseteq \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$ and $y' \in \varphi(y)$. Therefore, (C1) is preserved. It is obvious that (C2) and (C4) hold since $\pi(x) = \pi(y)$. The condition (C3) holds since $\varphi(y) \neq \varphi(z)$ does not hold. Due to $\pi(x) = \pi(y)$ and (*), (C4) is preserved as well.
10. \leq_φ -rule. If $(\leq nS.C) \in \mathcal{L}(x)$ then $(\leq nS.C) \in \mathcal{L}(x')$ for all $x' \in \varphi(x)$ and $(\leq nS.C) \in \mathcal{L}'(\pi(x))$. Due to P10, $\text{card}\{S^T(\pi(x), C)\} \leq n$. \leq_φ -rule is applicable if there are $(n+1)$ nodes y_0, \dots, y_n with $C \in \mathcal{L}(y_i)$ and $\varphi(y_i) \cap \varphi(y_j)$ for all $0 \leq i < j \leq n$. This implies that there are two nodes $y, z \in \{y_0, \dots, y_n\}$ such that $\pi(y) = \pi(z)$. Thus, $\varphi(y) \neq \varphi(z)$ cannot hold due to (*). In addition, if there are $x', x'' \in \varphi(x)$ such that x' has a S -neighbor $y' \in \varphi(y)$

and x'' has a S -neighbor $z' \in \varphi(z)$ with $C \in \mathcal{L}(y') \cap \mathcal{L}(z')$ then $x' \neq x''$ since NN_φ -rule and all \mathcal{SHOIQ} -rules are applied with higher priority (otherwise, $\varphi(y) \neq \varphi(z)$ holds).

Therefore, $\varphi(y), \varphi(z)$ and x', x'' can be chosen to merge by \leq_φ -rule. It is obvious that (C1) is preserved. If w' is a R -neighbor of w with $w \in \varphi(x') \cup \varphi(x'')$ and $w' \in \varphi(y') \cup \varphi(z')$ then $\langle \pi(w), \pi(w') \rangle \in \mathcal{E}(R)$ due to $\pi(y) = \pi(z)$ and (*). This implies (C2) holds. Moreover, (C3) is preserved since $\varphi(y) \neq \varphi(z)$ does not hold. Due to (*) and $\pi(y) = \pi(z)$, (C4) is preserved as well.

It holds that φ is a partitioning function since the non-applicable of \leq_φ -rule.

We now show that if \mathcal{SHOIQ} -trees and the partitioning function φ can be built with a function π satisfying (*) then \mathbf{T} is clash-free.

1. \mathbf{T} cannot contain a clash of the first kind due to (C1).
2. Assume that there are $x \neq y$ and $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ for some $o \in \mathbf{C}_o$. We have $\pi(x) = \pi(y)$ due to P13. From (C3) it follows that $x \neq y$ does not hold, which is a contradiction. This implies that \mathbf{T} cannot contain a clash of the second kind.
3. Assume that there is a node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ with $(\leq nR.C) \in \mathcal{L}(\varphi(x))$ and there are $(n+1)$ nodes $x_0, \dots, x_n \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$, $\varphi(x_i) \neq \varphi(x_j)$ with $i \neq j$, and $C \in \mathcal{L}(\varphi(x_i))$, $R \in \mathcal{L}(\langle \varphi(x), \varphi(x_i) \rangle)$ for $i \in \{0, \dots, n\}$. By P10, there are $x_i, x_j \in \{x_0, \dots, x_n\}$, $i \neq j$ such that $\pi(x_i) = \pi(x_j)$. This implies that $x_i \neq x_j$ does not hold. For all $x' \in \varphi(x_i)$ and $x'' \in \varphi(x_j)$ we have $\pi(x') = \pi(x_i)$ and $\pi(x'') = \pi(x_j)$, and thus $\pi(x') = \pi(x'')$ due to (C4). This implies that $x' \neq x''$ does not hold due to (C3), which contradicts $\varphi(x_i) \neq \varphi(x_j)$.

□

Theorem 11 *Let $(\mathcal{T}, \mathcal{R})$ be a \mathcal{SHOIQ} knowledge base. Algorithm 1 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it is worst-case optimal.*

Proof: From Lemma 8, 9 and 10, it follows that Algorithm 1 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$. According to Tobies in (Tobies, 2000), the consistency problem of \mathcal{SHOIQ} is NEXPTIME-complete. Algorithm 1 is worst-case optimal if we can show that it is NEXPTIME. In fact, Lemma 8 affirms that Algorithm 1 creates a set of \mathcal{SHOIQ} -trees whose size is exponential in the size of input $(\mathcal{T}, \mathcal{R})$. Given a set of \mathcal{SHOIQ} -trees with a partitioning function φ that defines an exponential number of partitions over the set of nodes, this algorithm can check whether (i) each expansion rule can be applied to each node of the trees and the partitions $\varphi(x)$ related to this node, and (ii) this set of trees is clash-free. From the pre-condition of the rules and the definition of clashes, it follows that the check process can run in polynomial time in the number of nodes and partitions. □

5. Conclusion and Discussion

We have presented in this paper a practical and worst-case optimal decision procedure for the logic \mathcal{SHOIQ} . From an analysis related to an eventual double exponential blow-up in size of \mathcal{SHOIQ} completion graphs resulting from the tableaux-based algorithm presented in (Horrocks & Sattler,

2007), we have introduced a new technique for constructing completion graphs without adding nominal nodes with new nominals. This technique is founded on the fact that fusions of nodes triggered by merging nominal nodes can be replaced with a function which would simulate this merging process. As a consequence, we can maintain tree-like structures for completion graphs on which the usual blocking mechanism can be reused.

This work can be considered a crucial step toward a tableaux-based algorithm for \mathcal{SHOIQ} with transitive closure of roles in concept and role inclusion axioms. Such tree-like structures for completion graphs would facilitate dramatically checking the existence of a path for satisfying transitive closures of roles occurring in terms such as $\exists Q^+.C$.

We would like to obtain by experiments results and real behaviours of our algorithm in practice. For this purpose, we plan to implement our algorithm.

References

- Fischer, M. J., & Ladner, R. I. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(18), 174–211.
- Horrocks, I., & Sattler, U. (2007). A tableau decision procedure for \mathcal{SHOIQ} . *Journal Of Automated Reasoning*, 39(3), 249–276.
- Horrocks, I., Sattler, U., & Tobies, S. (1999a). A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. In *LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen*.
- Horrocks, I., Sattler, U., & Tobies, S. (1999b). Practical reasoning for expressive description logics. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999)*. Springer.
- Le Duc, C. (2009). Decidability of \mathcal{SHI} with transitive closure of roles. In *Proceedings of the European Semantic Web Conference*. Springer-Verlag.
- Motik, B., Shearer, R., & Horrocks, I. (2009). Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research*, 36, 165–228.
- Patel-Schneider, P., Hayes, P., & Horrocks, I. (2004). Owl web ontology language semantics and abstract syntax. In *W3C Recommendation*.
- Tobies, S. (2000). The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12, 199–217.