An ExpSpace Tableau-based Algorithm for SHOIQ

Chan Le Duc, Myriam Lamolle, Olivier Curé

Noname manuscript No.
(will be inserted by the editor)

An EXPSPACE Tableau-based Algorithm for $\mathcal{SHOIQ}$

Chan Le Duc · Myriam Lamolle · Olivier Curé

the date of receipt and acceptance should be inserted later

Abstract In this work, we propose a worst-case optimal algorithm for deciding consistency in the description logic $\mathcal{SHOIQ}$. The construction of this algorithm is founded on the standard tableau-based method for $\mathcal{SHOIQ}$ and the technique used for designing a NEXPTIME algorithm for the two-variable fragment of first-order logic with counting quantifiers $\mathcal{C}^2$.

Keywords Description Logics · Decision Procedures · Tableaux · Complexity

1 Introduction

The ontology language OWL-DL [21] 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 [25]. Recently, another expressive description logic, namely $\mathcal{SROIQ}$ [12], has provided a logical base for OWL 2 which is an extension of OWL. An interesting feature of logics with nominals (denoted by $\mathcal{O}$ in $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$) is that they allow for expressing relationships, represented as role instances, between two sets of individuals which are represented as nominals or standard concepts. Such sets of individuals can be finitely enumerable or infinite. For instance, we consider the following axioms:

$$\{\mathsf{Earth}\} \sqsubseteq \exists \mathsf{isPartOf}.\{\mathsf{SolarSystem}\},$$

Chan Le Duc
LIASD Université Paris 8 - IUT de Montreuil
E-mail: leduc@iut.univ-paris8.fr

Myriam Lamolle
LIASD Université Paris 8 - IUT de Montreuil
E-mail: lamolle@iut.univ-paris8.fr

Olivier Curé
LIGM Université Marne La Vallée
E-mail: ocure@univ-mlv.fr

$$\mathsf{Thing} \sqsubseteq \exists \mathsf{isAttractedBy}.\{\mathsf{Earth}\}.$$

The former can be expressed by an ABox assertion. However, it would not be straightforward to use ABox assertions to represent the latter. As a result, a TBox with nominals is more expressive than a non-nominal TBox with an ABox since ABox assertions such as $C(a)$, $R(a, b)$ or $a \neq b$ can be directly expressed as TBox axioms with nominals as follows: $\{a\} \sqsubseteq C$, $\{a\} \sqsubseteq \exists R.\{b\}$, $\{a\} \sqsubseteq \neg\{b\}$.

There were several works on the consistency problem of a $\mathcal{SHOIQ}$ knowledge base. These works have not only shown decidability and complexity of the problem but also led to develop and implement efficient systems for reasoning on OWL-based ontologies. A result in [25] has shown that the consistency problem of a $\mathcal{SHOIQ}$ knowledge base is NEXPTIME-complete. Moreover, tableau-based algorithms presented in [13] and [12] for $\mathcal{SHOIQ}$ have been exploited to implement reasoners such as FaCT++ [27], Pellet [24], which inherit from the success of early Description Logic reasoners such as FaCT [10] and RACER [9].

In addition, there are several results which have provided worst-case optimal algorithms for Description Logics (DL) without nominal. The work in [20] has introduced the so-called $\mathcal{ALCQI}b^{+}_{reg}$ logic (capturing $\mathcal{SRIQ}$ [11]) that allows for a rich set of operators on roles by which one can simulate role inclusion axioms. Based on an automata approach, a powerful tool to establish complexity results for DLs and modal logics ([29], [28]), this work has also proposed a worst-case optimal algorithm for the satisfiability problem in $\mathcal{ALCQI}b^{+}_{reg}$. Another approach, namely *resolution-based*, has been used in [16] to devise a practical and worst-case optimal algorithm for deciding concept satisfiability in $\mathcal{SHIQ}$.

It has been shown that when nominals are added to these DLs the consistency problem is harder. In fact, the complexity jumps from EXPTIME-complete for $\mathcal{SHIQ}$ to NEXPTIME-complete for $\mathcal{SHOIQ}$ [25]. The work in [17] has indicated that when nominals are allowed in $\mathcal{SHIQ}$, the resolution-based approach yields a triple exponential decision procedure for the consistency problem. The authors have also identified that the interaction between nominals, inverse roles and number restrictions makes termination more difficult to be achieved, and thus, is responsible for this hardness. Recently, [19] has introduced a combination of the resolution-based approach with a tableau-based method. It yields an efficient algorithm for checking consistency of a $\mathcal{SHOIQ}^{+}$ knowledge base, which is an extension of $\mathcal{SHOIQ}$ with role operators. This algorithm has been exploited to implement the HermiT reasoner [23].

The presence of nominals in $\mathcal{SHOIQ}$ or $\mathcal{SHOIQ}^{+}$ makes tree-like structures for representing models more difficult to be maintained. As a consequence, the algorithms presented in [13] and [19] have employed non tree-like structures to represent models and adapted the blocking technique, which is a main tool to ensure termination for logics without nominal, such that it works again for the new structures. Moreover, the presence of nominals may trigger a merging process that starts by merging two nominal nodes and may propagate through at-most number restrictions. To deal with these issues, these algorithms have to introduce new expansion rules that may add new nominals in order to, on the one hand, isolate a non like-tree part

consisting of nominal nodes from a tree-like part consisting of non nominal nodes, and on the other hand, avoid infinite sequences of "generating-merging".

Our approach is inspired from a technique that was employed by De Giacomo and Lenzerini in [4] to simulate a functional role in DLs by using constructors in PDL. Intuitively, this technique said that "for every state s of a model M if $(\leq 1\ a)$ holds in s and if there is a a-transition from s to t_1 and a a-transition from s to t_2 then t_1 and t_2 are equivalent". We can apply this technique to DL context as follows : if an individual d must satisfy a term such as $(\leq 1\ a)$ and d has two a-neighbors d_1 and d_2 then d_1, d_2 have to satisfy the same set of semantic constraints, i.e. the label of d_1 is identical to that of d_2. It leads us to develop a technique that replaces merging d_1 into d_2 with (i) *propagating* the label from one to another and (ii) *memorizing* that d_1 and d_2 could be collapsed to a single individual. In other terms, this new technique is founded on the fact that fusions of nodes triggered by merging nominal nodes can be replaced with governing a *partitioning function* for memorizing nodes that could be merged. As a consequence, it allows us to maintain tree-like structures for representing models without adding new nominals, and thus, the blocking technique can be reused over these tree-like structures to achieve termination.

In addition, to obtain an EXPSPACE tableau-based algorithm for $\mathcal{SHOIQ}$, we use a technique that was proposed by Ian Pratt-Hartmann in [22] to construct a NEXPTIME algorithm for the logic $\mathcal{C}^2$ including $\mathcal{SHOIQ}$. Unlike the existing tableau-based algorithms, this technique does not explicitly build a graph for representing a model but it builds a structure, called a *frame*, from *star-types* each of which represents a set of individuals. A result from [22] shows that a model of a $\mathcal{C}^2$ knowledge base can be constructed from a frame tiled by *well selected* star-types.

The present paper is structured as follows. In the next section, we describe the logic $\mathcal{SHOIQ}$ and the consistency problem for a $\mathcal{SHOIQ}$ knowledge base. Section 3 provides an overview of recent tableau-based algorithms and their issues. This section gives also main arguments that show how these issues can be solved in our approach. In Section 4, we provide a definition of tableaux for $\mathcal{SHOIQ}$. Section 5 describes in detail a 2EXPSPACE tableau-based algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base. An advantage of this algorithm is that a tree-like structure can be maintained to obtain termination. Section **??** transfers a result in [22] from $\mathcal{C}^2$ to $\mathcal{SHOIQ}$. Based on these results, we propose a worst-case optimal algorithm for deciding consistency of a $\mathcal{SHOIQ}$ knowledge base. Finally, we discuss the results and future work.

2 The Description Logic $\mathcal{SHOIQ}$

In this section, we present the syntax, the semantics and main inference problems of $\mathcal{SHOIQ}$. We start by defining a role hierarchy and its semantics.

Definition 1 (role hierarchy) 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 use $\mathbf{R}_\text{I} = \{P^- \mid P \in \mathbf{R}\}$ to denote a set of inverse roles. Each element of $\mathbf{R} \cup \mathbf{R}_\text{I}$ is called a $\mathcal{SHOIQ}$-role. We define a function $R^\ominus$ which returns R^- if $R \in \mathbf{R}$, and returns R if $R \in \mathbf{R}_\text{I}$.

$*$ A *role hierarchy* $\mathcal{R}$ is a finite set of *role inclusion axioms* $R \sqsubseteq S$ where R and S are two $\mathcal{SHOIQ}$-roles. A relation $\sqsubseteq\!\!\!\!\!\!\blacksquare$ is defined as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{R^{\ominus} \sqsubseteq S^{\ominus} \mid R \sqsubseteq S \in \mathcal{R}\}$. We define a function $\mathsf{Trans}(R)$ which returns true iff there is some $Q \in \mathbf{R}_+ \cup \{P^{\ominus} \mid P \in \mathbf{R}_+\}$ such that $Q \sqsubseteq\!\!\!\!\!\!\blacksquare R$. A role R is called *simple* w.r.t. $\mathcal{R}$ if $\mathsf{Trans}(Q) = \mathsf{false}$.

$*$ 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 $\mathcal{I} \models \mathcal{R}$. $\lhd$

Notice that the simplicity of roles which relies on the function $\mathsf{Trans}(\cdot)$ plays a crucial role in guaranteeing decidability of $\mathcal{SHIQ}$ [15]. The underlying idea is that if a role R is simple then it is sufficient to count "direct" R-neighbors t of an individual s, i.e. $\langle s, t\rangle \in R^{\mathcal{I}}$ for some interpretation $\mathcal{I}$, in order to satisfy a restriction that bounds the number of R-neigbors of s.

Definition 2 (terminology) Let $\mathbf{C}$ be a non-empty set of *concept names* with a non-empty 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 w.r.t. a role hierarchy. We denote $\bot$ for $\neg\top$.

$*$ The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps each concept name to a subset of $\Delta^{\mathcal{I}}$ such that
$\top^{\mathcal{I}} = \Delta^{\mathcal{I}},\ (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},\ (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},\ (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$,
$\mathsf{card}\{o^{\mathcal{I}}\} = 1$ for all $o \in \mathbf{C}_o$,
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}, \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$,
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, \langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$,
$(\geq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}\} \geq n\}$,
$(\leq n\, S.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}\} \leq n\}$
where $\mathsf{card}\{S\}$ 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 $\mathcal{I} \models \mathcal{T}$. $\lhd$

Definition 3 (knowledge base) A pair $(\mathcal{T}, \mathcal{R})$ is called a $\mathcal{SHOIQ}$ knowledge base where $\mathcal{R}$ is a $\mathcal{SHOIQ}$ role hierarchy and $\mathcal{T}$ is a $\mathcal{SHOIQ}$ terminology.

$*$ A knowledge base $(\mathcal{T}, \mathcal{R})$ is 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 C is called *satisfiable* w.r.t. $(\mathcal{T}, \mathcal{R})$ 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. $(\mathcal{T}, \mathcal{R})$.

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

Since negation is allowed in the logic $\mathcal{SHOIQ}$, unsatisfiability and subsumption w.r.t. $(\mathcal{T}, \mathcal{R})$ can be reduced to 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 $\mathcal{SHOIQ}$-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in [15]. According to [1], $\mathsf{nnf}(C)$ can be computed in polynomial time in the size of C. For a concept C, we denote the nnf of C by $\mathsf{nnf}(C)$ and the nnf of $\neg C$ by $\dot{\neg} C$

Let D be an $\mathcal{SHOIQ}$-concept in NNF. We define $\mathsf{cl}(D)$ 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 $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ as follows:

$$\mathsf{cl}(\mathcal{T}, \mathcal{R}) = \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{cl}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R}) \ \textit{where}$$

$$\begin{aligned}\mathsf{cl}(E, \mathcal{R}) = {} & \mathsf{cl}(E) \cup \{\dot{\neg} C \mid C \in \mathsf{cl}(E)\} \cup \\ & \{\forall S.C \mid (\forall R.C \in \mathsf{cl}(E), S \boxplus R) \textit{ or } (\dot{\neg}\forall R.C \in \mathsf{cl}(E), S \boxplus R) \\ & \qquad \textit{where } S \textit{ occurs in } \mathcal{T} \textit{ or } \mathcal{R}\} \cup \qquad (1) \\ & \{\bowtie m S.C \mid (\leq n S.C) \in \mathsf{cl}(E), \bowtie \ \in \{\leq, \geq\}, 1 \leq m \leq n\} \qquad (2)\end{aligned}$$

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 $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ is inspired from the Fischer-Ladner closure that was introduced in [6]. The closure $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ contains not only sub-concepts syntactically obtained from $\mathcal{T}$ but also sub-concepts that are semantically derived from $\mathcal{T}$ w.r.t. $\mathcal{R}$. For instance, (i) if $\forall S.C$ is a sub-concept from $\mathcal{T}$ and $R \boxplus S \in \mathcal{R}$ then $\forall R.C \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$, or (ii) if $(\leq n S.C) \in \mathsf{cl}(\mathcal{T})$ with $n \geq 1$ then $(\leq m S.C), (\geq m S.C) \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$ for all $1 \leq m \leq n$. This means that $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ must be large enough to cover all sub-concepts which represent potential nondeterminisms. These extensions of $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ are anticipated for the construction of tableaux described in Section 4.

3 Related Work and Algorithm Overview

In this section, we discuss several related works that employ tableau-based approach to devise efficient algorithms for the consistency problem of a DL knowledge base. Next, we provide an overview of our algorithm and explain how it addresses the

issues encountered in tableau-based algorithms for DLs with inverse roles, number restrictions and nominals.

3.1 Related Work

Donini and Massacci in [7] have provided a tableau-based algorithm for the logic $\mathcal{ALC}$ that runs in single exponential time. The authors have proposed an unusual strategy that allows for reusing unsatisfiable concepts that are already discovered in a branch. As a result, this strategy may help to reduce the tree expansion required by the main complexity sources : *or-branching* ($\sqcup$) and *and-branching* ($\sqcap$,$\exists$, $\forall$) [5].

Goré and Nguyen in [8] have proposed for the logic $\mathcal{SHI}$ an exponential time algorithm that is based on a *global caching method*. Basically, to decide satisfiability of a concept X w.r.t. a $\mathcal{SHI}$ knowledge base, this algorithm starts by building a and/or graph with an initial node containing X. Each node of the graph is associated with a status sat or unsat. When a node is decided to be sat, this status will be propagated to ancestors. The construction terminates if all nodes are saturated, i.e. no rule is applicable. This strategy of construction allows for reusing both unsatisfiable and satisfiable concepts that are already discovered.

As far as we know, the algorithms developed in these works have not been implemented yet. Moreover, it is not straightforward to extend these algorithms to DLs with number restrictions and nominals.

Motik, Shearer and Horrocks in [19] have recently introduced an algorithm that is founded on a hypertableau-based approach for the logic $\mathcal{SHOIQ}^+$ ($\mathcal{SHOIQ}$ with role operators). First, this approach translates TBox axioms of a DL knowledge base into first-order formulae, the so-called DL-clauses. Next, it applies derivation rules to devise new ABox assertions until saturation (i.e. no new ABox assertion can be generated). To deal with nominals and achieve termination, this approach has introduced a new rule, called *NI*-rule, that adds *implicitly* new nominals to ABox. This rule makes a crucial impact on (i) maintaining the tree-like part enabling to reuse blocking technique on it, and (ii) avoiding infinite sequences of "generating-merging" caused by generating-terms $\exists R.C$, $(\geq nR.C)$ and merging-terms $(\leq nR.C)$ with nominals.

Horrocks and Sattler have presented in [13] a tableau-based algorithm for the DL $\mathcal{SHOIQ}$ that relies on two structures: *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 satisfiability of constraints imposed *locally* on each individual of the tableau.

To check consistency of a knowledge base, tableau-based algorithms try to build a completion graph whose finiteness is ensured by *blocking technique*. The underlying idea of 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, tableau-based algorithms manage to

"push" nominal nodes (i.e. nodes containing nominals in their label) outside from tree-like structures.

Our approach introduced in this paper relies on the tableau-based algorithm presented in [13], from which we have learned the following characteristics of completion graphs for $\mathcal{SHOIQ}$:

3.1.1 Two-part structure of completion graphs

A completion graph for $\mathcal{SHOIQ}$ 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 of this part has at most one predecessor and may have a nominal successor.

3.1.2 The border between two parts of the structure

The two-part structure of completion graphs for $\mathcal{SHOIQ}$ makes a crucial impact on behaviours of the tableau-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 avoid an infinite "generating-merging" sequence, and (iii) we have to prevent a blocking node z from having a nominal descendant x with $(\leq nS.C) \in \mathcal{L}(x)$ ($\mathcal{L}(x)$ denotes a set of x's labels) such that the non-nominal predecessor y of x is an S-neighbor of x and $C \in \mathcal{L}(y)$. Such a node x may be "embarrassing" since this situation may lead to devise by "unraveling" a model that has a nominal individual (corresponding to x) connecting to an infinite number of distinct individuals (corresponding to y).

To obtain the conditions mentioned for termination, [13] introduced some new rules which can add nominal nodes with new nominals and push "embarrassing" nodes 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 tableau-based algorithm presented in [13] is not worst-case optimal.

3.2 Overview of the novel tableau-based algorithm

The observations from the previous section 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 $\mathcal{SHOIQ}$. Rather, this behaviour results from the 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-like structure can be maintained by some way such that it allows us to avoid merging *explicitly* nominal nodes.

Although experiments indicate that the tableau and hypertableau-based algorithms for $\mathcal{SHOIQ}$ have good behaviours in practice, none of them was shown to be worst-case optimal. The contribution of the present paper consists in proposing a worst-case optimal tableau-based algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base. This algorithm is aimed at exploiting a structure which contains no non-tree part. As a result, we can reuse the usual blocking technique to ensure termination. The underlying idea of our approach is to introduce a function, namely *partitioning function*, which 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, expansion rules related only to $\mathcal{SHIQ}$ constructors (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. In turn, expansion rules related to nominals 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 the tree-like structures except for labels.

Summing up,

1. For termination issue, our algorithm maintains tree-like structures which is built by applying the expansion rules related to $\mathcal{SHIQ}$ constructors. Thus, the standard blocking technique can be used to achieve termination. As a consequence, the size of tree-like structures is exponential in the size of input.
2. To avoid infinite sequences of "generating-merging", our algorithm does not perform explicitly merges triggered by nominal nodes. Instead, we employ a partitioning function over nodes for memorizing nodes that could be merged.
3. To help in devising correctly a model by unraveling over tree-like structures, we use a new rule, called $\bowtie$-rule, which ensures that if there is a blocking node y that has a nominal descendant z then there exist two nodes w, w' between y and z such that if w' is a S-predecessor of w and $\mathcal{L}(w')$ contains C then $\mathcal{L}(w)$ contains no term such as $(\leq nS.C)$. In addition, this rule guarantees that if a leaf node z cannot be blocked by such a blocking node then there is a non-leaf node y such that z and y must belong to the same partition.

4 A Tableau for $\mathcal{SHOIQ}$

Basically, a tableau structure is employed to represent a model of a $\mathcal{SHOIQ}$ knowledge base. Properties in a tableau definition for $\mathcal{SHOIQ}$ express the semantic constraints resulting directly from the logic constructors in $\mathcal{SHOIQ}$. 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 for building a finite representation of models.

Regarding the definition of tableaux for $\mathcal{SHOIQ}$ presented in [13], 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 requires an extra expansion rule, namely $\bowtie$-rule, introduced in Algorithm 1. The presence of this rule may have an impact on the so-called "pay-as-you-go" behaviour of the tableau-based algorithm presented in this paper.

Definition 4 (tableau) 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 non-empty set of individuals, $\mathcal{L}$: $\mathbf{S} \rightarrow 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ and $\mathcal{E}$: $\mathbf{R}_{(\mathcal{T},\mathcal{R})} \rightarrow 2^{\mathbf{S}\times\mathbf{S}}$. For all $s \in \mathbf{S}$, $C, C_1, C_2 \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$ 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 $\mathsf{nnf}(\neg C_1 \sqcup C_2) \in \mathcal{L}(s)$,
- P2 If $C \in \mathcal{L}(s)$ then $\doteq C \notin \mathcal{L}(s)$,
- 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 $\forall S.C \in \mathcal{L}(s)$, $Q \sqsubseteq S$, $\mathsf{Trans}(Q)$ 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 $t \in \mathbf{S}$ 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 $\mathsf{card}\{S^T(s, C)\} \leq n$,
- P11 If $(\geq nS.C) \in \mathcal{L}(s)$ then $\mathsf{card}\{S^T(s, C)\} \geq n$ where $S^T(s, C) := \{t \in \mathbf{S} | \langle s, t\rangle \in \mathcal{E}(S) \wedge C \in \mathcal{L}(t)\}$,
- 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 $\doteq C \in \mathcal{L}(t)$
- P13 If $o \in \mathcal{L}(s) \cap \mathcal{L}(t)$ for some $o \in \mathbf{C}_o$ then $s = t$,
- P14 For each $o \in \mathbf{C}_o$, if o occurs in $\mathcal{T}$ then there is $s \in \mathbf{S}$ such that $o \in \mathcal{L}(s)$.
- P15 If $(\leq nS.C) \in \mathcal{L}(s)$ and there is $t \in \mathbf{S}$ 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 $\{(\leq mS.C), (\geq mS.C)\} \subseteq \mathcal{L}(s)$. $\lhd$

All properties of Definition 4 except for P15 are taken from [15] and [13]. If we say that an individual s has an 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 role transitivity makes local property of tableaux more difficult to be ensured. (P6) in Definition 4 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, the local property is guaranteed for $\mathcal{SHIQ}$ tableaux. However, it is much more difficult to preserve the local property of tableaux for logics which allow for transitive closure of roles. For instance, tableaux introduced in [18] for $\mathcal{SHI}$ with transitive closure of roles, namely $\mathcal{SHI}_+$, do not have the local property.

According to this argument, P13 in Definition 4 is responsible for the loss of local property in $\mathcal{SHOIQ}$ tableaux since P13 can be applied to two arbitrary individuals with which the same nominal is associated. The local property of tableaux has an important impact on the way by which a tableau-based algorithm can build completion graphs. For instance, each application of the expansion rules presented in [15] for $\mathcal{SHIQ}$ performs some changes to its neighbours or itself. Then, checking clash-freeness can be performed on each node and its neighbors. On the contrary, the application of o-rule presented in [13] for $\mathcal{SHOIQ}$ may change two arbitrary nodes while the algorithm introduced in [18] for concept satisfiability in $\mathcal{SHI}_+$ has to check over the whole tree whether there exists a path for satisfying axioms such as $R \sqsubseteq P^+$. As a consequence, the loss of local property may lead to augment the complexity, e.g., from EXPTIME-complete for $\mathcal{SHIQ}$ to NEXPTIME-complete for $\mathcal{SHOIQ}$. The results from [26] have indicated that the concept satisfiability problems in $\mathcal{ALC}$ (with *GCI*) and $\mathcal{SHIQ}$ are EXPTIME-complete. This implies that the concept satisfiability problem in $\mathcal{SHI}$ is EXPTIME-complete due to $\mathcal{ALC} \subseteq \mathcal{SHI} \subseteq \mathcal{SHIQ}$. It remains an open question whether the complexity of the satisfiability problem in $\mathcal{SHI}_+$ is EXPTIME-complete.

Lemma 1 *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 [14]. From this proof, it is not hard to prove Lemma 1 for $\mathcal{SHOIQ}$ tableaux with the new properties P13, P14 and P15.

5 A 2EXPSPACE decision procedure for $\mathcal{SHOIQ}$

In this section, we introduce $\mathcal{SHOIQ}$-trees with Definition 5. Intuitively, a $\mathcal{SHOIQ}$-tree is defined by considering nominals as non-nominal concepts. The important difference between $\mathcal{SHOIQ}$-trees and completion trees for $\mathcal{SHIQ}$ is the presence of a *blocking condition with two layers* that is governed by *iterated* and *blocked* nodes. An 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 blocked node y in a $\mathcal{SHOIQ}$-tree is a descendant of an iterated node x such that the blocking condition for y is applied from x. That is x is considered as root of the subtree on which the condition blocking is applied to each blocked descendant y of x. The reason for introducing the blocking condition with two layers is to allow for defining *inner* nodes which will be presented in Definition 7.

5.1 Construction of the algorithm

Definition 5 ($\mathcal{SHOIQ}$-tree) 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 $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$, is defined as follows:

$*$ V_o is a set of nodes containing a root node $\widehat{x}_o \in V_o$. Each node $x \in V_o$ is labelled with a function $\mathcal{L}_o$ such that $\mathcal{L}_o(x) \subseteq \mathsf{cl}(\mathcal{T}, \mathcal{R})$ and $o \in \mathcal{L}_o(\widehat{x}_o)$. A node $x \in V_o$ is called *nominal* if $o' \in \mathcal{L}_o(x)$ for some $o' \in \mathbf{C}_o$. In addition, the inequality relation $\neq_o$ 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 $\mathcal{L}_o$ such that $\mathcal{L}_o(\langle x, y\rangle) \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$.

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

$*$ A node y is called a *descendant* of x if $y \in \mathsf{succ}^n(x)$ for some $n > 0$. A node y is called an *ancestor* of x if $y = \mathsf{pred}^n(x)$ for some $n > 0$. To ensure that $\mathbf{T}_o$ is a tree, it is required that (i) x is a descendant of $\widehat{x}_o$ for all $x \in V_o$ with $x \neq \widehat{x}_o$, and (ii) each node $x \in V_o$ with $x \neq \widehat{x}_o$ has a unique predecessor.

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

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

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

$*$ A node x is called *iterated* by y w.r.t. a node x_o if x has no nominal ancestor except for $\widehat{x}_o$ and there are integers $n, m > 0$ and nodes x', y' such that

1. $x_o = \mathsf{pred}^n(y)$, $y = \mathsf{pred}^m(x)$,
2. $x' = \mathsf{pred}^1(x)$, $y' = \mathsf{pred}^1(y)$,
3. $\mathcal{L}_o(x) = \mathcal{L}_o(y)$, $\mathcal{L}_o(x') = \mathcal{L}_o(y')$,
4. $\mathcal{L}_o(\langle x', x\rangle) = \mathcal{L}_o(\langle y', y\rangle)$, and
5. if there are z, z' and $i > 0$ such that $z' = \mathsf{pred}^1(z)$, $\mathsf{pred}^i(z') = x_o$, $\mathcal{L}_o(z) = \mathcal{L}_o(y)$, $\mathcal{L}_o(z') = \mathcal{L}_o(y')$ and $\mathcal{L}_o(\langle z', z\rangle) = \mathcal{L}_o(\langle y', y\rangle)$ then $i \geq n$.

A node x is called *1-iterated* by y if x is iterated by y w.r.t. $\widehat{x}_o$. A node x is called *blocked* by y, denoted by $y = \mathsf{b}(x)$, if x is iterated by y w.r.t. a 1-iterated node x_o.

$*$ In the following, we often use $\mathcal{L}(x)$, $\mathcal{L}(\langle x, y\rangle)$, $S^{\mathbf{T}}(x, C)$ and $\neq$ instead of $\mathcal{L}_o(x)$, $\mathcal{L}_o(\langle x, y\rangle)$, $S^{\mathbf{T}_o}(x, C)$ and $\neq_o$, respectively. This does not cause any confusion since $V_o \cap V_{o'} = \emptyset$ and $E_o \cap E_{o'} = \emptyset$ if $o \neq o'$. In addition, $x \neq_o y$ is never defined for $x \in V_o$ and $y \in V_{o'}$ with $o \neq o'$. $\lhd$

We can remark that the definition of 1-iterated nodes in Definition 5 for $\mathcal{SHOIQ}$-trees is very similar to the standard definition of blocked nodes for $\mathcal{SHIQ}$ completion

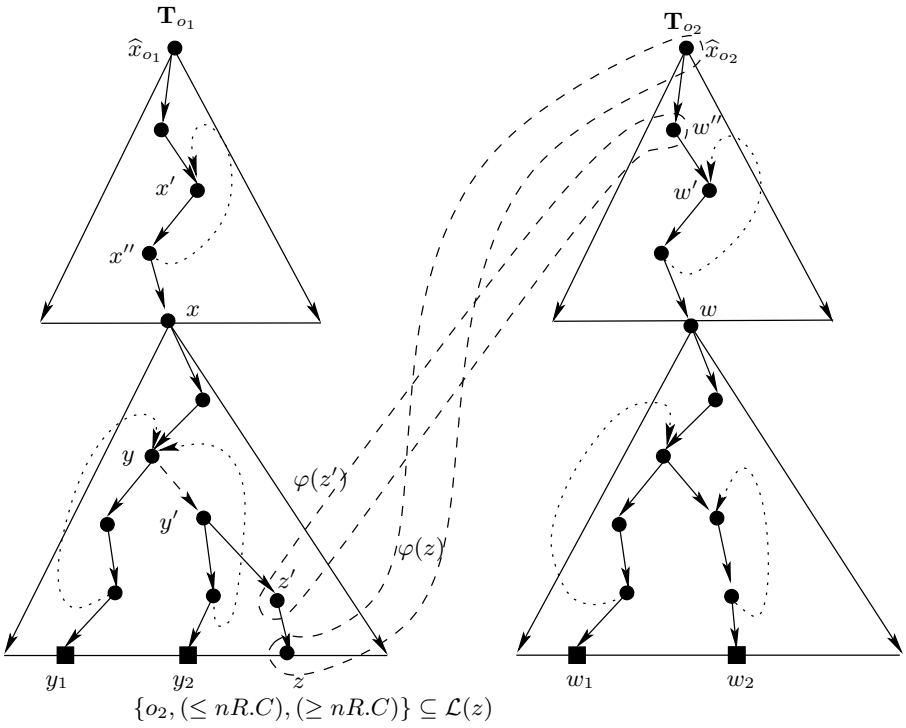

Fig. 1 $\mathcal{SHOIQ}$-trees with a partitioning function φ illustrate the notions presented in Definition 5, 6 and 7. In this figure, two superimposed triangles represent 2 layers of a $\mathcal{SHOIQ}$-tree. Nodes x, w are 1-iterated while y_1, y_2, w_1, w_2 are blocked nodes. Especially, z is a leaf node which is a nominal descendant of the blocking node $y = \mathsf{b}(y_2)$. In addition, we have $\widehat{x}_{o_2} \in \varphi(z)$ since $o_2 \in \mathcal{L}(z) \cap \mathcal{L}(\widehat{x}_{o_2})$. Due to $\{(\leq nR.C), (\geq nR.C)\} \subseteq \mathcal{L}(z)$, $C \in \mathcal{L}(z')$, $R^{\ominus} \in \mathcal{L}(\langle z', z\rangle)$ and $\mathcal{L}(z) = \mathcal{L}(\widehat{x}_{o_2})$, there is a R-successor w'' of $\widehat{x}_{o_2}$ such that $w'' \in \varphi(z')$ and $C \in \mathcal{L}(w'')$. According to Definition 7, z' is inner.

trees (see [15]). Moreover, if we consider the subtree rooted at a 1-iterated node as a $\mathcal{SHIQ}$ completion tree then blocked nodes according to Definition 5 are also blocked nodes according to the standard definition for this $\mathcal{SHIQ}$ completion tree.

In the sequel, following each definition we present some properties which are immediate consequences of the definition, and directly comment on them. When a property is not trivial, a proof can be found in the comment. Additionally, some notions or notations introduced in these properties would be used in proofs thereafter.

Property 1 Each leaf node of a $\mathcal{SHOIQ}$-tree must be either a nominal or blocked node.

Comment: A nominal leaf node either is a descendant of a 1-iterated node or has no 1-iterated ancestor. As we will show, each non-root nominal node can be grouped with a root node that is always nominal. This explains why we do not need to generate successors of a non-root nominal node. In Figure 1, z is a nominal leaf node while y_1, y_2, w_1 and w_2 are blocked nodes.

Property 2 A $\mathcal{SHOIQ}$-tree consists of two layers : the first layer is formed of nodes from the root to 1-iterated nodes or nominal nodes, and the second layer consists of nodes from each 1-iterated node to blocked or nominal nodes. In addition, each node x in the layer 2 has a unique 1-iterated node, denoted $\widehat{\mathsf{b}}(x)$, such that $\widehat{\mathsf{b}}(x)$ is an ancestor of x.

Comment: In Figure 1, nodes x and x' belong to the first layer and x is 1-iterated by x'. Nevertheless, nodes y and y_1 belong to the second layer and y_2 is blocked by y. Moreover, $x = \widehat{\mathsf{b}}(y_1) = \widehat{\mathsf{b}}(y_2) = \cdots$ and $w = \widehat{\mathsf{b}}(w_1) = \widehat{\mathsf{b}}(w_2) = \cdots$. The necessity of the two-layer structure of $\mathcal{SHOIQ}$-trees will be clarified in properties and their comment following Definition 7.

Definition 6 ($\mathcal{SHOIQ}$-forest) Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A $\mathcal{SHOIQ}$-*forest* for $(\mathcal{T}, \mathcal{R})$ is a pair $\mathbf{G} = \langle \mathbf{T}, \varphi \rangle$, where $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ is a set of $\mathcal{SHOIQ}$-trees for $(\mathcal{T}, \mathcal{R})$ with $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$, and φ is a *partitioning* function $\varphi : \mathcal{V} \to 2^{\mathcal{V}}$ with $\mathcal{V} = \bigcup_{o \in \mathbf{C}_o} V_o$. We denote $\mathcal{L}'(\langle x, y \rangle) = \mathcal{L}_o(\langle x, y \rangle)$ if $\langle x, y \rangle \in E_o$, and $\mathcal{L}'_o(\langle x, y \rangle) = \{S^{\ominus} \mid S \in \mathcal{L}_o(\langle y, x \rangle)\}$ if $\langle y, x \rangle \in E_o$ for some $o \in \mathbf{C}_o$. The partitioning function φ satisfies the following conditions:

1. For each $x \in \mathcal{V}$, $\varphi(x)$ is the *partition* of x with $x \in \varphi(x)$. There are $x_0, \cdots, x_n \in \mathcal{V}$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ with $0 \leq i < j \leq n$ and $\bigcup_{0 \leq i \leq n} \varphi(x_i) = \mathcal{V}$;
2. For all $x, x' \in \mathcal{V}$, if $x' \in \varphi(x)$ then $\varphi(x) = \varphi(x')$ and $\mathcal{L}(x) = \mathcal{L}(x')$. We denote $\Lambda(\varphi(x)) = \mathcal{L}(x)$. In addition, an inequality relation over partitions can be described as follows : for $x, x' \in \mathcal{V}$ we define $\varphi(x) \neq \varphi(x')$ if there are two nodes $y \in \varphi(x)$ and $y' \in \varphi(x')$ such that $y \neq_o y'$ for some $o \in \mathbf{C}_o$;
3. For all $\varphi(x)$ and $\varphi(x')$, if there are two edges $\langle y, y' \rangle \in E_o$ and $\langle w, w' \rangle \in E_{o'}$ with $o, o' \in \mathbf{C}_o$ such that $y, w \in \varphi(x)$, $y', w' \in \varphi(x')$ and $\mathcal{L}'(\langle y, y' \rangle) \neq \emptyset, \mathcal{L}'(\langle w, w' \rangle) \neq \emptyset$ then $\mathcal{L}'(\langle y, y' \rangle) = \mathcal{L}'(\langle w, w' \rangle)$.
   We define a function $\Lambda(\langle \cdot, \cdot \rangle)$ for labelling edges ended by two partitions as follows: $\Lambda(\langle \varphi(x), \varphi(x') \rangle) = \mathcal{L}'(\langle z, z' \rangle)$ where $z \in \varphi(x)$, $z' \in \varphi(x')$, $\mathcal{L}'(\langle z, z' \rangle) \neq \emptyset$, and $\{\langle z, z' \rangle, \langle z', z \rangle\} \cap E_{o'} \neq \emptyset$ for some $o' \in \mathbf{C}_o$. We say $\varphi(x')$ is a S-neighbour partition of $\varphi(x)$ if $S \in \Lambda(\langle \varphi(x), \varphi(x') \rangle)$.
4. For all $x, x' \in \mathcal{V}$, if $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$ and $\varphi(x) \neq \varphi(x')$ does not hold then $\varphi(x) = \varphi(x')$; and
5. If $(\leq nR.C) \in \Lambda(\varphi(x))$ for some $x \in \mathcal{V}$ and there exist $n+1$ nodes $x_0, \cdots, x_n \in \mathcal{V}$ such that
   - $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ for all $0 \leq i < j \leq n$, and
   - $C \in \Lambda(\varphi(x_i))$, $R \in \Lambda(\langle \varphi(x), \varphi(x_i) \rangle)$ for all $i \in \{0, \cdots, n\}$

   then $\varphi(x_l) \neq \varphi(x_m)$ for all $0 \leq l < m \leq n$.
6. If $(\geq nR.C) \in \Lambda(\varphi(x))$ for some $x \in \mathcal{V}$ then $\varphi(x)$ has n R-neighbour partitions $\varphi(x_1), \cdots, \varphi(x_n)$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and $C \in \Lambda(\varphi(x_i))$ for all $1 \leq i < j \leq n$.

$*$ **Clashes:** $\mathbf{T}$ is said to contain a *clash* if one of the following conditions holds:

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

The idea behind the notion of partitioning function is to capture the semantics of nominals and its consequences. Such a partitioning function determines a graph structure which is described by the following property:

Property 3 A $\mathcal{SHOIQ}$-forest $\langle \mathbf{T}, \varphi \rangle$ according to Definition 6 determines a graph $\mathbf{G}_\varphi^{\mathbf{T}} = (\mathbf{V}, \mathbf{E}, \Lambda)$ where $\mathbf{V}$ is a set of nodes corresponding to partitions, i.e., $\mathbf{V} = \{\varphi(x) \mid x \in \mathcal{V}\}$ and $\mathbf{E}$ is a set of edges with $\mathbf{E} = \{\langle \varphi(x), \varphi(y) \rangle \mid \langle x', y' \rangle \in E_o, x' \in \varphi(x), y' \in \varphi(y), o \in \mathbf{C}_o\}$. The nodes and edges of $\mathbf{G}_\varphi^{\mathbf{T}}$ are labelled by Λ as described in Definition 6. It allows us to say that $\varphi(y)$ is a neighbour of $\varphi(x)$ if $\langle \varphi(x), \varphi(y) \rangle \in \mathbf{E}$ or $\langle \varphi(y), \varphi(x) \rangle \in \mathbf{E}$, and that $\varphi(y)$ is a R-neighbour of $\varphi(x)$ if $R \in \Lambda(\langle \varphi(x), \varphi(y) \rangle)$. In addition, there is an inequality relation $\neq$ over $\mathbf{V}$, i.e., $\varphi(x) \neq \varphi(y)$ iff there are $x' \in \varphi(x)$ and $y' \in \varphi(y)$ such that $x' \neq_o y'$ for some $o \in \mathbf{C}_o$.

Comment: the conditions 1 and 2 in Definition 6 formulate the properties of an ordinary partitioning function while the conditions 3 and 4 show how to label nodes and edges of $\mathbf{G}_\varphi^{\mathbf{T}}$. More precisely, the condition 2 ensures that all nodes of a partition have the same label. Since the empty edge label is not needed to be propagated, the condition 3 guarantees that all non-empty edges ended by two nodes belonging to two given partitions have the same label. The conditions 4 and 5 take care of consequences resulting from the semantics of nominals. Indeed, if we consider nodes of $\mathbf{G}_\varphi^{\mathbf{T}}$ as individuals of a tableau, and the inequality relation $\neq$ over $\mathbf{V}$ is respected, then the condition 4 ensures that for each $o \in \mathbf{C}_o$ there is a unique partition $\varphi(x)$ such that $o \in \Lambda(\varphi(x))$. Additionally, if $(\leq nR.C) \in \Lambda(\varphi(x))$ and the inequality relation $\neq$ over $\mathbf{V}$ is respected, then the condition 5 guarantees that node $\varphi(x)$ has at most n *distinct* R-neigbours $\varphi(x_0), \cdots, \varphi(x_{n-1})$ in $\mathbf{G}_\varphi^{\mathbf{T}}$ such that $C \in \Lambda(\varphi(x_i))$ for $0 \leq i \leq n-1$. Note that $\varphi(x)$ is different from $\varphi(y)$, denoted $\varphi(x) \neq \varphi(y)$, iff $\varphi(x) \cap \varphi(y) = \emptyset$.

We now describe the tableau-based algorithm, namely Algorithm 3, whose goal is to construct from a knowledge base $(\mathcal{T}, \mathcal{R})$ a $\mathcal{SHOIQ}$-forest $\mathbf{G} = \langle \mathbf{T}, \varphi \rangle$. To do this, Algorithm 3 applies the expansion rules as described in Algorithm 1 and 2, and terminates when none of the rules is applicable. The obtained $\mathbf{G}$ is called *complete*, and if $\mathbf{G}$ contains no clash then $\mathbf{G}$ is called *clash-free*. In this case, we also say $\mathbf{T}_o$ is complete and clash-free for all $\mathbf{T}_o \in \mathbf{T}$. Before presenting these expansion rules, we introduce an operation, namely Propagate, which is used in expansion rules.

Propagation $\mathsf{Propagate}(\varphi(x), \varphi(x'), \varphi(y))$ is an operation which propagates (i) node labels from a partition $\varphi(x)$ to another partition $\varphi(x')$, and vice versa, (ii) edge labels from the edges ended by nodes of $\varphi(x)$ and $\varphi(y)$ to the edges ended by nodes of $\varphi(x')$ and $\varphi(y)$, and vice versa. In other terms, $\mathsf{Propagate}(\cdots)$ merges $\varphi(x)$

into $\varphi(x')$, and $\langle\varphi(x),\varphi(y)\rangle$ into $\langle\varphi(x'),\varphi(y)\rangle$. More precisely, let $\mathbf{G} = \langle\mathbf{T},\varphi\rangle$ be a $\mathcal{SHOIQ}$-forest with $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ and $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$. $\mathsf{Propagate}(\varphi(x),\varphi(x'),\varphi(y))$ updates the label of nodes and edges in $\mathbf{T}$ as follows:

1. $\mathcal{L}(z) = \mathcal{L}(x) \cup \mathcal{L}(x')$ for all $z \in \varphi(x) \cup \varphi(x')$,
2. for all $z, z' \in \varphi(x) \cup \varphi(x')$ and $w, w' \in \varphi(y)$, if z is a S-neighbour of w and $\mathcal{L}'(\langle z', w'\rangle) \neq \emptyset$ then
   (a) if z' is a successor of w' and $S \notin \mathcal{L}(\langle w', z'\rangle)$ then $\mathcal{L}(\langle w', z'\rangle) = \mathcal{L}(\langle w', z'\rangle) \cup \{S\}$,
   (b) if w' is a successor of z' and $S \notin \mathcal{L}(\langle z', w'\rangle)$ then $\mathcal{L}(\langle z', w'\rangle) = \mathcal{L}(\langle z', w'\rangle) \cup \{S^{\ominus}\}$.

$\sqsubseteq$-rule:	if	$C \sqsubseteq D \in \mathcal{T}$ and $\mathsf{nnf}(\neg C \sqcup D) \notin \mathcal{L}(x)$
	then	$\mathcal{L}(x') = \mathcal{L}(x') \cup \{\mathsf{nnf}(\neg C \sqcup D)\}$ 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	$C_1 \sqcup C_2 \in (x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x') = \mathcal{L}_o(x') \cup \{C\}$ 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 blocked, x is not non-root nominal, 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 $\mathsf{Trans}(Q)$ and $Q \sqsubseteq S$, and
	3.	there is an Q-neighbour y of x such that $\forall Q.C \notin \mathcal{L}(y)$
	then	$\mathcal{L}(y') = \mathcal{L}(y') \cup \{\forall Q.C\}$ 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 $\{C, \dot{\neg}C\} \cap \mathcal{L}(y) = \emptyset$
	then	$\mathcal{L}(y') = \mathcal{L}_o(y') \cup \{E\}$ with some $E \in \{C, \dot{\neg}C\}$ for all $y' \in \varphi(y)$.
$\geq$-rule:	if 1.	$(\geq n\ S.C) \in \mathcal{L}(x)$, x is not blocked, x is not non-root nominal, and
	2.	x has no n S-neighbours $y_1, ..., y_n$ 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 $y_1, ..., y_n$ 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.	$\mathsf{card}\{S^{\mathbf{T}}(x, C)\} > n$ and there are two S-neighbours y, z of x with $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, y is not an ancestor of z and not $y \neq z$
	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 $u \neq y$.
$\bowtie$-rule:	if 1.	$(\leq n R.C) \in \mathcal{L}(x)$, $\{(\leq l\ R.C), (\geq l\ R.C)\} \not\subseteq \mathcal{L}(x)$ for all $l \leq n$,
	2.	$(\leq k\ R.C) \notin \mathcal{L}(x)$ for all $k < n$, and
	3.	x has a R-neighbour 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 $x' \in \varphi(x)$.

Figure 1: Expansion rules for $\mathcal{SHIQ}$

In the following, we discuss and stress new aspects of the expansion rules in Algorithm 1 and 2 for $\mathcal{SHOIQ}$.

The rules in Algorithm 1 maintain the tree-like structure of $\mathcal{SHOIQ}$-forest and they are similar to those in [13] except that if a concept C is added to the label of a node x due to application of these rules then C is propagated to the label of each node $y \in \varphi(x)$. Moreover, all rules in Algorithm 1 except for $\exists$- and $\geq$-rule update only the label of nodes or edges and do no change on the partitioning function φ. Especially, when the $\leq$-rule is applied to a node x with two S-neighbours y, z of x, it must propagate the label of $\langle x, y\rangle$ to that of all $\langle x', z'\rangle$ (or $\langle z', x'\rangle$) where $x' \in \varphi(x)$ and $z' \in \varphi(z)$, and set the label of $\langle x, y\rangle$ to empty set. This may change φ only if $\varphi(y)$ is singleton.

A new feature of the generating rules ($\exists$-rule and $\geq$-rule) with respect to those in [13] is that these rules are not applied to non-root nominal nodes. By Definition 6, a non-root nominal node belongs always to the partition $\varphi(\widehat{x}_o)$ where $\widehat{x}_o$ is a root node (note that all root nodes are nominal). As a result, it is not necessary to generate successors of a non-root nominal node since it "inherits" all neighbours of $\widehat{x}_o$.

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. $\mathsf{Propagate}(\varphi(x), \varphi(x'), \varphi(y))$ for each y such that $\{\langle x'', y\rangle, \langle y, x''\rangle\} \cap E_o \neq \emptyset$ for $x'' \in \varphi(x) \cup \varphi(x')$, $o \in \mathbf{C}_o$.
		2. $\varphi(y') = \varphi(x) \cup \varphi(x')$ for all $y' \in \varphi(x) \cup \varphi(x')$.
$\leq_\varphi$-rule:	if 1.	$(\leq nR.C) \in \mathcal{L}(x)$,
	2.	there are nodes $y_0, \cdots, y_n$ with $\varphi(y_i) \cap \varphi(y_j) = \emptyset$, $0 \leq i < j \leq n$, $C \in \Lambda(\varphi(y_i))$, $R \in \Lambda(\langle\varphi(x), \varphi(y_i)\rangle)$ 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-neighbour y', x'' has a R-neighbour y'' s.t. $C \in \mathcal{L}(y') \cap \mathcal{L}(y'')$, $\varphi(y') \cap \varphi(y'') = \emptyset$, and not $\varphi(y') \neq \varphi(y'')$
	then	1. $\mathsf{Propagate}(\varphi(y'), \varphi(y''), \varphi(x))$,
		2. $\varphi(y) = \varphi(y') \cup \varphi(y'')$ for all $y \in \varphi(y') \cup \varphi(y'')$.

Figure 2: New expansion rules for $\mathcal{SHOIQ}$

We would like to stress the difference between the $\leq$-rule in Algorithm 1 and the $\leq_\varphi$-rule in Algorithm 2. Basically, the $\leq$-rule with the others in Algorithm 1 ensures that the semantic constraints (except for nominals) imposed by the label of each node are satisfied. The rule $\leq_\varphi$-rule with o_φ-rule takes care of the semantic constraint of nominals and its consequences, i.e., these rules deal with partitions rather than nodes, and do no change on the structure of trees. The necessity of both rules is a consequence of the two-part structure for models of a $\mathcal{SHOIQ}$ knowledge base : (i) the first part which is finite and contains nominals, will be constructed from partitions, (ii) the second part will be built by unravelling subtrees which contain no nominals and are maintained by $\mathcal{SHIQ}$-rules. Note that the $\leq$-rule and $\leq_\varphi$-rule are complementary, i.e., when one of these rules has been applied to a node the other one may be applicable to the same node. For instance, after applying the $\leq_\varphi$-rule to a node x with $(\leq nS.C) \in \mathcal{L}(x)$, x may have $(n+1)$ S-neighbours containing C since a S-neighbour partition of $\varphi(x)$ may contain two S-neighbours that include C.

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 $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$ be an initial tree such that $V_o = \{\widehat{x}_o\}$, $\mathcal{L}(\widehat{x}_o) = \{o\}$, and there are no $x, y \in V_o$ such that $x \neq y$;
2 A partitioning function φ is initialized with $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$;
3 **while** *there is a non-empty set* S *of expansion rules in Figure 1 and 2 such that each* $\mathsf{r} \in \mathsf{S}$ *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* $\{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ **then**
6 **return** YES ;
7 **else**
8 **return** NO ;

Algorithm 3: A 2EXPSPACE algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base

Another aspect that needs to be clarified is that when two partitions $\varphi(y')$ and $\varphi(y'')$ combine to obtain the new partition $\varphi(y') \cup \varphi(y'')$, the inequality relations are automatically transfered from $\varphi(y')$ and $\varphi(y'')$ to $\varphi(y') \cup \varphi(y'')$ due to the definition of inequality relation over partitions. Indeed, if there is some $\varphi(z)$ such that $\varphi(z) \neq \varphi(y')$ (resp. $\varphi(z) \neq \varphi(y'')$), i.e., there are some $z' \in \varphi(z)$ and $w \in \varphi(y')$ (resp. $w \in \varphi(y'')$) with $z' \neq_o w$ and $o \in \mathbf{C}_o$ then $\varphi(z) \neq \varphi(y') \cup \varphi(y'')$ since $w' \in \varphi(y') \cup \varphi(y'')$.

By the $\bowtie$-rule, each node x containing a term $(\leq\ nS.C)$ has exactly m S-neighbours containing C with some $m \leq n$. As a result, this rule and $\geq$-rule ensure that if there are two nodes $y, y' \in \varphi(x)$ then y and y' have exactly m S-neighbours which contain C in their label.

Finally, we can avoid infinite sequences of "merging-and-generating", the so-called "yo-yo" effect ([2], [3]), in the tableau algorithm presented in this paper without pruning nodes, since all merges due to number restrictions or nominals are performed by updating the partitioning function. We will clarify how it works by examples in Section 5.2.

In the sequel, we present the specific features of a complete and clash-free $\mathcal{SHOIQ}$-forest. These features, which are related to the notions introduced in Definition 7, are crucial to establish soundness of the tableau algorithm presented in this paper. Basically, the goal of the soundness proof is to build a tableau from a complete and clash-free $\mathcal{SHOIQ}$-forest. For this purpose, we use the technique introduced in [15] to define paths (corresponding to individuals of the tableau) which are originated from the root node of each $\mathbf{T}_o \in \mathbf{T}$, go down through successors of a node and may be infinitely lengthened by "unraveling". The specific features of complete and clash-free $\mathcal{SHOIQ}$-trees make such paths avoid infinitely going through nominal nodes.

Definition 7 (Innerness and unravelling) Let $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ and φ be a set of $\mathcal{SHOIQ}$-trees and a partitioning function, respectively, which are built by Algorithm 3 for a $\mathcal{SHOIQ}$ knowledge base $(\mathcal{T}, \mathcal{R})$ where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$.

$*$ **Core node:**

A node $x \in V_o$ with $o \in \mathbf{C}_o$ is called *core node* if x has a nominal descendant z with $x = \mathsf{pred}^k(z)$ $(k \geq 0)$ such that there are $(\leq n_i R_i.C_i) \in \mathcal{L}(\mathsf{pred}^i(z))$, $R_i^{\ominus} \in \mathcal{L}(\langle \mathsf{pred}^{(i+1)}(z), \mathsf{pred}^i(z)\rangle)$, $C_i \in \mathcal{L}(\mathsf{pred}^{i+1}(z))$ for $0 \leq i \leq (k-1)$.

$*$ **Inner node:** A node $x \in V_o$ with $o \in \mathbf{C}_o$ is called *inner* node if

1. x is a descendant of a 1-iterated node; and
2. x is a core node.

$*$ **Unravellable node:** A node $x \in V_o$ with $o \in \mathbf{C}_o$ is called *unravellable* node if the following conditions are satisfied:

1. x is blocked;
2. $\mathsf{b}(x)$ is not a core node; and
3. if $\mathsf{b}(x)$ has a blocked descendant y' such that $\mathsf{b}(y') = \mathsf{pred}^m(\mathsf{b}(x))$ $(m > 0)$ then $\mathsf{b}(y')$ is not a core node.

Before providing the intuition behind the notions introduced in Definition 7, we formulate and prove an immediate consequence of the definition of inner node.

Property 4 If a node $x \in V_o$ is inner for some $o \in \mathbf{T}_o$ then there is an ancestor x' of a 1-iterated node such that $\varphi(x) = \varphi(x')$.

Comment: By definition, there is a nominal node z with $x = \mathsf{pred}^k(z)$ $(k \geq 0)$. Since $\mathbf{T}_o$ is complete, i.e. $\bowtie$-rule and o-rule are not applicable, we have $o \in \mathcal{L}(\widehat{x}_o)$, $\varphi(z) = \varphi(\widehat{x}_o)$, and thus $\{(\leq n_0 R_0.C_0), (\geq n_0 R_0.C_0)\} \subseteq \mathcal{L}(\widehat{x}_o)$. Due to non-applicability of NN-, $\geq$- and $\leq_\varphi$-rules, it holds that $\widehat{x}_o$ has a $R_0^{\ominus}$-successor w_1 such that $\varphi(\mathsf{pred}^1(z)) = \varphi(w_1)$, and thus $\{(\leq n_1 R_1.C_1), (\geq n_1 R_1.C_1)\} \subseteq \mathcal{L}(w_1)$. This implies that w_1 has exactly n_1 distinct R_1-neighbours $u_1, \cdots, u_{n_1}$ with $C_1 \in \mathcal{L}(u_i)$. Since $z \neq \mathsf{pred}^2(z)$ it holds that w_1 has a $R{\ominus}_1$-successor w_2 such that $\varphi(\mathsf{pred}^2(z)) = \varphi(w_2)$, and thus $\{(\leq n_2 R_2.C_2), (\geq n_2 R_2.C_2)\} \subseteq \mathcal{L}(w_2)$. By the same argument, there are $w_i = \mathsf{succ}^i(\widehat{x}_{\mathbf{T}_o})$ such that $\varphi(\mathsf{pred}^i(z)) = \varphi(\mathsf{succ}^i(\widehat{x}_o))$ for all $0 \leq i \leq k$. Moreover, $\mathsf{succ}^k(\widehat{x}_o)$ must belong to the layer 1 of $\mathbf{T}_o$ since the path from $\mathsf{pred}^k(z)$ to z contains no two identical edges which are necessary to bring $\mathsf{succ}^k(\widehat{x}_o)$ to the layer 2.

The rule $\bowtie$-rule is crucial to establish Property 4. This rule makes paths from a nominal node to its inner ancestors "hard", i.e., nodes of such paths cannot be duplicated by unravelling.

Intuition behind the notion of inner node. A node x belonging to the layer 1 of a complete $\mathcal{SHOIQ}$-tree $\mathbf{T}_o \in \mathbf{T}$ is completely expanded by the rules in Algorithm 1 and 2, i.e., all neighbours of x and thus all neighbour partitions of $\varphi(x)$ are generated. Moreover, if a node y is located in the layer 2 of $\mathbf{T}_o$ and y is inner, i.e., $\varphi(y) = \varphi(x)$ for some x located in the layer 1 (by Property 4), then y can be considered as completely expanded since all neighbour partitions of $\varphi(y)$ are determined from x. It allows us to omit the subtree rooted at y, denoted by $\mathbf{T}_o[y]$, when constructing a tableau in the soundness proof of Algorithm 3. Notice that the two-layer structure of $\mathcal{SHOIQ}$-trees helps avoid "breaking" the inner nodes which "refer" to nodes in $\mathbf{T}_o[y]$ after omitting $\mathbf{T}_o[y]$. Indeed, it is possible to omit the subtree $\mathbf{T}_o[y]$ rooted at an inner node y only if (i) there is a node x such that x is completely expanded with

$y \in \varphi(x)$, and (ii) there is no inner node y' such that $\varphi(y') = \{y'\} \cup X'$ where X' are a subset of nodes of $\mathbf{T}_o[y]$. It is obvious that we cannot obtain this property if we adopt the one-layer structure (which includes nodes from root to 1-iterated nodes) for $\mathcal{SHOIQ}$-trees.

Intuition behind the notion of unravellable node. A blocked node x is called unravellable according to Definition 7 if we can infinitely lenghthen without violating number restrictions the paths (considered as tableau individuals) which go through the predecessor of x, then $\mathsf{b}(x)$ and so on. To attain this goal, Definition 7 requires that (i) if there is a nominal node z between $\mathsf{b}(x)$ and a blocked descendant y of $\mathsf{b}(x)$ then there exists a non-inner node y_z between $\mathsf{b}(x)$ and z, and (ii) otherwise, i.e., if there is no nominal node between $\mathsf{b}(x)$ and a blocked descendant y of $\mathsf{b}(x)$ then y must be unravellable.

It is obvious that if this requirement is met we can infinitely duplicate the edges ended by the nodes from $\mathsf{b}(x)$ to nodes y_z or to unravellable nodes y. As a result, infinite paths can be formed from these duplicated edges without violating number restrictions.

Property 5 If a blocked node x has no inner ancestor then x is unravellable.

Comment: Assume that x is not unravellable. According to the definition of unravellable nodes, there are the two following possibilities:

1. $\mathsf{b}(x)$ has a nominal descendant z with $\mathsf{b}(x) = \mathsf{pred}^k(z)$ and each node $y = \mathsf{pred}^{k'}(z)$ for $k \leq k' \leq 0$ is inner. This implies that $\mathsf{b}(x)$ is inner and $\mathsf{b}(x)$ is an ancestor of x, which is a contradiction.
2. $\mathsf{b}(x)$ has a blocked descendant y' such that $\mathsf{b}(y') = \mathsf{pred}^m(\mathsf{b}(x))$ $(m > 0)$, $\mathsf{b}(y')$ has a nominal descendant z' with $\mathsf{b}(y') = \mathsf{pred}^l(z')$ $(l > 0)$ and each node $w = \mathsf{pred}^{l'}(z')$ for $0 < l' \leq l$ is inner. This implies $\mathsf{b}(y')$ is inner and $\mathsf{b}(y')$ is an ancestor of x, which is a contradiction.

In the following section, we present different examples to illustrate the main features of Algorithm 3 and discuss on how the algorithm deals with the main issues such as "yo-yo" effect and exponential blow-up of new nominals, etc. The section starts by running the algorithm with a simple knowledge base. The goal is to show how a partitioning function can help to avoid merging. Next, we will execute the algorithm with some interesting knowledge bases taken from [13] and [19].

5.2 Examples of applying the algorithm

5.2.1 An example of getting started.

We consider a knowledge base $\mathcal{K}_1$ consisting of the following axioms:

$$\begin{array}{lcl} \mathcal{R}_1 & = & \{F \sqsubseteq R, \mathsf{Trans}(R)\}, \\ \mathcal{T}_1 & = & \{\top \sqsubseteq \exists S.o, \\ & & \ o \sqsubseteq \neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C\} \end{array}$$

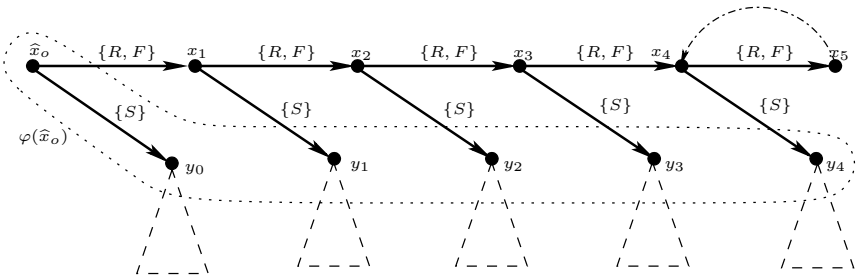

Fig. 2 A $\mathcal{SHOIQ}$-tree with a partitioning function φ for the example

According to Algorithm 3, there is only one $\mathcal{SHOIQ}$-tree $\mathbf{T} = (V, E, \mathcal{L}, \widehat{x}_o, \neq)$ which will be built since there is one nominal o in the knowledge base. Figure 2 illustrates graphically the construction of $\mathbf{T}$. First, it initializes $\mathcal{L}(\widehat{x}_o) = \{o\}$ and $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$. Then, by applying $\sqsubseteq$-rule and $\sqcap$-rule to $\widehat{x}_o$, we have :

$\mathcal{L}(\widehat{x}_o) \supseteq \{o, \exists S.o, (\neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C), \exists S.o, \neg C, \exists F.C, \forall R.\exists F.C\}$

In turn, $\exists$-, $\forall$-, $\forall_+$- and $\sqsubseteq$-rule are applied to $\widehat{x}_o$ to obtain two new nodes y_0 and x_1 such that $\mathcal{L}(\langle \widehat{x}_o, y_0\rangle) = \{S\}$, $\mathcal{L}(\langle \widehat{x}_o, x_1\rangle) = \{R, F\}$, $\varphi(y_0) = \{y_0\}$, $\varphi(x_1) = \{x_1\}$ and

$\mathcal{L}(y_0) \supseteq \{o, \exists S.o, (\neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C)\}$,
$\mathcal{L}(x_1) \supseteq \{C, \exists F.C, \forall R.\exists F.C, \exists S.o, \neg o\}$.

Then, $\sqcap$-rule is applied to y_0 and x_1, we have

$\mathcal{L}(y_0) \supseteq \{o, \exists S.o, (\neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C), \neg C, \exists F.C, \forall R.\exists F.C\}$,
$\mathcal{L}(x_1) \supseteq \{C, \exists F.C, \forall R.\exists F.C, \exists S.o, \neg o\}$.

In the same way, we can continue applying $\exists$-rule, $\forall$-rule, $\forall_+$, $\sqsubseteq$-rule and $\sqcap$-rule to yield x_2, x_3, x_4, x_5 and y_1, y_2, y_3, y_4 such that x_2, y_1 are successors of x_1; x_3, y_2 are successors of x_2; x_4, y_3 are successors of x_3, and x_5, y_4 are successors of x_4. In addition, $\mathcal{L}(y_i) = \mathcal{L}(\widehat{x}_o)$ for $i \in \{1, \cdots, 4\}$ and $\mathcal{L}(x_j) = \mathcal{L}(x_1)$, $\varphi(x_j) = \{x_j\}$ for $j \in \{2, \cdots, 5\}$.

At this stage, o_φ-rule can be applied to nominal nodes $\widehat{x}_o$ and y_i for $i \in \{0, \cdots, 4\}$ in order to obtain $\varphi(\widehat{x}_o) = \varphi(y_0) = \cdots = \varphi(y_4) = \{\widehat{x}_o, y_0, \cdots, y_4\}$. No label propagation is needed. Application of o_φ-rule yields an edge $\langle \varphi(\widehat{x}_o), \varphi(\widehat{x}_o)\rangle$ with $\Lambda(\langle \varphi(\widehat{x}_o), \varphi(\widehat{x}_o)\rangle) = \{S\}$ since there is an edge $\langle \widehat{x}_o, y_0\rangle$ with $\widehat{x}_o, y_0 \in \varphi(\widehat{x}_o)$.

From the definition of 1-iterated and blocked nodes, we obtain that x_3 is 1-iterated by x_2 and x_5 is blocked by x_4. In addition, y_i for $i \in \{0, \cdots, 4\}$ are nominal leaves. It is obvious that the obtained $\mathbf{T}$ is complete and clash-free. Therefore, $\mathcal{K}_1$ is consistent.

We now modify slightly $\mathcal{K}_1$ to obtain $\mathcal{K}_1'$ as follows.

$$\begin{array}{lcl} \mathcal{R}_1 & = & \{F \sqsubseteq R, \mathsf{Trans}(R)\}, \\ \mathcal{T}_1' & = & \{\top \sqsubseteq \exists S.(o \sqcap (\leq 1S^-.\top)), \\ & & \quad o \sqsubseteq \neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C\} \end{array}$$

When the at-most number restriction $(\leq 1S^-.\top)$ occurs in the axiom, $\bowtie$-rule is applicable to y_i for $i \in \{1, \cdots, 4\}$ since $(\leq 1S^-.\top) \in \mathcal{L}(y_i)$. This rule guesses a unique $m = 1$ and adds $(\leq 1S^-.\top), (\geq 1S^-.\top)$ to $\mathcal{L}(y_i)$. In turn, application of o_φ-rule and $\leq_\varphi$-rule must group all nodes to a unique partition $\varphi(\widehat{x}_o)$. This implies that $\{C, \neg C\} \subseteq \Lambda(\varphi(\widehat{x}_o))$. Thus, $\mathcal{K}_1'$ is not consistent. $\lhd$

5.2.2 An example of dealing with the "generate-and-merge" problem.

This problem, known as "yo-yo" effect for DLs with nominals, is discused in [13] and [19]. These works proposed pruning nodes to prevent an infinite sequence of "generating-and-merging". Since Algorithm 3 performs no merge, such infinite sequences of "generating-and-merging" can be avoided by lengthening branches of a $\mathcal{SHOIQ}$ tree until blocked nodes occur.

To clarify this, we consider the following KB which is taken from [13]:

$$\begin{array}{rcl} \mathcal{K}_2 & = & \{A \sqsubseteq \exists R.(A \sqcap \exists R.(A \sqcap \forall R^-.o)), \\ & & o \sqsubseteq A\} \end{array}$$

First, a root node $\widehat{x}_o$ is created with $\mathcal{L}(\widehat{x}_o) = \{o\}$ and $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$. Then, by applying $\sqsubseteq$-, $\exists$-, $\sqcap$-rule to $\widehat{x}_o$, we create a R-successor x_1 of $\widehat{x}_o$, and a R-successor x_2 of x_1 with $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$, $\varphi(x_1) = \{x_1\}$, $\varphi(x_2) = \{x_2\}$ and :

$$\begin{array}{lcl} \mathcal{L}(\widehat{x}_o) & \supseteq & \{o, A, \exists R.(A \sqcap \exists R.(A \sqcap \forall R^-.o))\}, \\ \mathcal{L}(x_1) & \supseteq & \{A, \exists R.(A \sqcap \forall R^-.o)\} \\ \mathcal{L}(x_2) & \supseteq & \{A, \forall R^-.o, A \sqcap \forall R^-.o\} \end{array}$$

Then, application of $\forall$-rule to x_2 adds o to $\mathcal{L}(x_1)$, which leads to apply o-rule to x_1 and $\widehat{x}_o$. We now obtain the following partitions : $\varphi(\widehat{x}_o) = \{\widehat{x}_o, x_1\}$ and $\varphi(x_2) = \{x_2\}$ with

$$\begin{array}{lcl} \mathcal{L}(\widehat{x}_o) = \mathcal{L}(x_1) & \supseteq & \{o, A, \exists R.(A \sqcap \forall R^-.o), \exists R.(A \sqcap \exists R.(A \sqcap \forall R^-.o))\}, \\ \mathcal{L}(x_2) & \supseteq & \{A, \forall R^-.o, A \sqcap \forall R^-.o\} \end{array}$$

We continue applying $\sqsubseteq$-rule and $\exists$-rule to x_2 in order to create a R-successor x_3 of x_2, then apply $\exists$-rule to x_3 in order to create a R-successor x_4 of x_3. At this stage, x_4 is blocked by x_2 since $\mathcal{L}(x_4) = \mathcal{L}(x_2)$ and $\mathcal{L}(\widehat{x}_0) = \mathcal{L}(x_1) = \mathcal{L}(x_3)$ with $\varphi(\widehat{x}_o) = \{\widehat{x}_o, x_1, x_3\}$. Therefore, the algorithm could avoid "yo-yo" effect without pruning nodes.

Note that the algorithm will create another R-successor x_1' of $\widehat{x}_0$ with $\mathcal{L}(x_1') = \mathcal{L}(x_2)$ since the term $\exists R.(A \sqcap \forall R^-.o)$ is propagated from x_1 to $\widehat{x}_0$. $\lhd$

5.3 Soundness and completeness of the algorithm

As mentioned in Section 5.1, we establish soundness of the algorithm by construcing a tableau from a complete and clash-free set of $\mathcal{SHOIQ}$-trees with a partitioning function. The proof given in this section is founded on the notions introduced in Definition 7 and Properties 5 and 4.

Lemma 2 (Termination) *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Algorithm 3 terminates and builds $\mathcal{SHOIQ}$-trees whose the size is bounded by a double exponential function in the size of $(\mathcal{T}, \mathcal{R})$.*

Proof First, we compute a upper bound of the $\mathcal{SHOIQ}$ tree's depth from the blocking condition. This upper bound equals $K = 2^{2m+k} \times 2$ where $m = \mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\}$,

and k is the number of roles occurring in $\mathcal{T}$ and $\mathcal{R}$ plus their inverse. Moreover, the number of neighbours of any node is bounded by $M = \sum m_i$ where $M = \sum m_i + E$, m_i occurs in a number restriction term $(\geq m_i R.C)$ appearing in $\mathcal{T}$ and E is the number of distinct terms $\exists R.C$ appearing in $\mathcal{T}$. This implies that the size of $\mathcal{SHOIQ}$-trees is bounded by $M^{2^{2m+k} \times 2}$.

Applications of rules in Algorithm 1 and 2 neither remove concepts from the label of nodes, nor remove roles from the label of edges except for setting the label of some edges to empty. However, when the label of an edge is set to empty it remains to be empty forever. Moreover, no rule attempts to remove nodes and edges from trees. Regarding to the partitioning function φ, it does no change on the structure of trees. Furthermore, partitions are never shrunken following application of the rules.

We have shown that (i) the size of $\mathcal{SHOIQ}$-trees is bounded, (ii) the number of nodes and edges never decreases, (iii) the label of nodes and edges are never reduced, (iv) partitions are never reduced, and (v) each application of the rules either creates a node, or changes node, edge labels or partitions. Therefore, termination of the algorithm is a consequence of these facts. □

Lemma 3 (Soundness) *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. If Algorithm 3 answers "YES" by yielding a clash-free and complete 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 with a function φ which is built by Algorithm 3 for $(\mathcal{T}, \mathcal{R})$ where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \widehat{x}_o, \neq_o)$. First, we show that φ satisfies all conditions of a partitioning function as described in Definition 6.

- The condition 1 is satisfied since the generating rules ($\exists$-rule and $\geq$-rule) always define $\varphi(x)$ for each fresh node x. In addition, the set $\varphi(x)$ is never shrunken by applying expansion rules for all x. Moreover, all expansion rules either define $\varphi(x)$ for a fresh node x or combine $\varphi(x)$ with $\varphi(y)$ to redefine $\varphi(y') = \varphi(x) \cup \varphi(y)$ for all $y' \in \varphi(x) \cup \varphi(y)$.
- The condition 2 holds since (i) the condition 1 implies that if $x' \in \varphi(x)$ then $\varphi(x') = \varphi(x)$, (ii) when the generating rules are applied to a node x, fresh nodes y_i are created with $\varphi(y_i) = \{y_i\}$, (ii) when the other rules are applied to a node x, this may change the label of x and propagate (due to the operation Propagate if φ_o-rule, and $\leq_\varphi$-rule are applied) added concepts to the label of all nodes of the partition containing x, i.e., $\varphi(x)$.
- The condition 3. If $\varphi(x)$ and $\varphi(x')$ are singleton then the condition 3 is obvious. Assume that $\varphi(x')$ (or $\varphi(x)$) is not singleton. This implies that $\varphi(x')$ is formed by an application of o_φ-rule or $\leq$-rule. In this case, the condition 3 holds due to the definition of the operation Propagate.
- The condition 4. Assume that $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ and $x \neq x'$ does not hold. Since $\mathbf{T}$ is clash-free, $\varphi(x) \neq \varphi(x')$ does not hold. Moreover, since $\mathbf{T}$ is complete the o_φ-rule is not applicable. This implies that $\varphi(x) \cap \varphi(x') \neq \emptyset$. Due to the condition 2, we have $\varphi(x) = \varphi(x')$.
- The condition 5 holds due to $\leq_\varphi$-rule.
- The condition 6 holds due to $\geq$-rule. In fact, we never merge two neighbour partitions $\varphi(x_i)$ and $\varphi(x_j)$ such that $\varphi(x_i) \neq \varphi(x_j)$.

Before constructing a tableau from $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$, we introduce some notions and properties that are used in this construction.

Let $\mathbf{I}$ be the set of all inner nodes x such that x has no inner ancestor. Let $\mathbf{B}$ be the set of all blocked nodes y such that y has no ancestor that belongs to $\mathbf{I}$. This implies that each $y \in \mathbf{B}$ is unravellable due to Property 5.

We use $\bar{\mathbf{B}}$ to denote the set of nodes $x \in \mathbf{B}$ such that each $x' \in \varphi(x)$ is not an ancestor of a node $y \in \mathbf{B} \cup \mathbf{I}$. As a consequence, if $x \in \mathbf{B} \setminus \bar{\mathbf{B}}$ then x does not need to be unravelled.

For $y \in \bar{\mathbf{B}}$, if $\mathsf{b}(y)$ has an inner descendant z with $\mathsf{b}(y) = \mathsf{pred}^k(z)$ then there is a node $w = \mathsf{pred}^i(z)$ with $0 < i \leq k$ such that w is not inner (otherwise, y is not unravellable). Let l be the least number such that $w_z = \mathsf{pred}^l(z)$ is not inner and w_z has no inner ancestor. We use $\bar{\mathbf{I}}$ to denote the set of successors of all w_z. This implies that $\bar{\mathbf{I}} \subseteq \mathbf{I}$.

For each $\varphi(x)$ with $x \in \bar{\mathbf{B}} \cap V_o$ and $o \in \mathbf{C}_o$, we define an infinite tree $\mathbf{T}_o^{\varphi(x)}$ and a function p (stand for Projection) from the nodes of $\mathbf{T}_o^{\varphi(x)}$ to V_o as follows :

- We use $\widehat{V}_o^{\varphi(x)}$ to denote the set of $\mathbf{T}_o^{\varphi(x)}$'s nodes. For each node $v \in \widehat{V}_o^{\varphi(x)}$, the set of v's successors in $\mathbf{T}_o^{\varphi(x)}$ is denoted by $\widetilde{\mathsf{succ}}^1(v)$. We initialize $\widehat{V}_o^{\varphi(x)} = \{\mathsf{b}^1(\varphi(x))\}$ and $\mathsf{p}(\mathsf{b}^1(\varphi(x))) = \mathsf{b}(x)$ where $\mathsf{b}^1(\varphi(x))$ is the root of $\mathbf{T}_o^{\varphi(x)}$.
- For each $x' \in \mathsf{succ}^1(\mathsf{p}(v))$ with $v \in \widehat{V}_o^{\varphi(x)}$,
  - If x' is not blocked and $x' \notin \bar{\mathbf{I}}$ then we add to $\widehat{V}_o^x$ and $\widetilde{\mathsf{succ}}^1(v)$ a new node v' that is a successor of v with $\mathcal{L}(\langle v, v'\rangle) = \mathcal{L}(\langle \mathsf{p}(v), x'\rangle)$, and define $\mathsf{p}(v') = x'$.
  - If x' is unravellable (note that each blocked descendant y' of $\mathsf{b}(x)$ such that y' has no inner ancestor, is unravellable) then we add to $\widehat{V}_o^{\varphi(x)}$ and $\widetilde{\mathsf{succ}}^1(v)$ a new node v' with $\mathcal{L}(\langle v, v'\rangle) = \mathcal{L}(\langle \mathsf{p}(v), x'\rangle)$, and define $\mathsf{p}(v') = \mathsf{b}(x')$.

Note that all nodes of $\mathbf{T}_o^{\varphi(x)}$ are duplicated from the non-inner nodes of the subtrees rooted at $\mathsf{b}(x)$ or $\mathsf{b}(x')$ where $\mathsf{b}(x')$ is an ancestor of $\mathsf{b}(x)$. We now define a tableau $T = (\mathbf{S}, \mathcal{L}'', \mathcal{E})$ as follows:

- $\mathbf{S} = \mathbf{S}_1 \cup \mathbf{S}_2$ where
  - $\mathbf{S}_1 = \{\varphi(x) \mid x = \mathsf{pred}^i(x'), x' \in \mathbf{I} \cup \mathbf{B}, i \geq 1\}$, and $\mathcal{L}''(\varphi(x)) = \Lambda(\varphi(x))$
  - $\mathbf{S}_2 = \bigcup_{x \in \bar{\mathbf{B}} \cap V_o,\ o \in \mathbf{C}_o} \widehat{V}_o^{\varphi(x)}$, and $\mathcal{L}''(v) = \mathcal{L}(\mathsf{p}(v))$ 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

$$\mathcal{E}_1(R) = \{\langle \varphi(x), \varphi(y)\rangle \in \mathbf{S}_1 \times \mathbf{S}_1 \mid R \in \Lambda(\langle \varphi(x), \varphi(y)\rangle)\},$$

$$\mathcal{E}_2(R) = \{\langle s, t\rangle \in \mathbf{S}_2 \times \mathbf{S}_2 \mid R \in \mathcal{L}(\langle \mathsf{p}(s), \mathsf{p}(t)\rangle) \vee R^{\ominus} \in \mathcal{L}(\langle \mathsf{p}(t), \mathsf{p}(s)\rangle)\}$$

$$\mathcal{E}_3(R) = \{\langle \varphi(x'), \mathsf{b}^1(\varphi(x))\rangle \in \mathbf{S}_1 \times \mathbf{S}_2 \mid R \in \Lambda(\langle \varphi(x'), \varphi(x)\rangle)\} \cup \{\langle \mathsf{b}^1(\varphi(x)), \varphi(x')\rangle \in \mathbf{S}_2 \times \mathbf{S}_1 \mid R^{\ominus} \in \Lambda(\langle \varphi(x'), \varphi(x)\rangle)\}$$

$$\mathcal{E}_4(R) = \{\langle s, \varphi(w)\rangle \in \mathbf{S}_2 \times \mathbf{S}_1 \mid R \in \mathcal{L}(\langle \mathsf{p}(s), w\rangle), w \in \bar{\mathbf{I}}\} \cup \{\langle \varphi(w), s\rangle \in \mathbf{S}_1 \times \mathbf{S}_2 \mid R^{\ominus} \in \mathcal{L}(\langle \mathsf{p}(s), w\rangle), w \in \bar{\mathbf{I}}\}.$$

From the definition of the tableau T, we remark that $\mathbf{S}_1$ is finite and contains nominals while $\mathbf{S}_2$ is infinite. Moreover, $\varphi(x)$ may belong to $\mathbf{S}_1$ even if $x \in \mathbf{I} \cup \mathbf{B}$ since x may have an ancestor x' which is an ancestor of a node $y \in \mathbf{I} \cup \mathbf{B}$. Regarding role instances, $\mathcal{E}_3(R)$ includes role instances from the predecesor of an unravellable

node x to a copy of $\mathsf{b}(x)$. $\mathcal{E}_4(R)$ contains an infinite number of role instances from non-inner nodes to inner nodes.

We now show that T satisfies all properties in Definition 4.

- P1-P4 hold due to the non-applicability of $\sqsubseteq$-rule, $\sqcap$-rule and $\sqcup$-rule in Algorithm 1 and the facts that $\mathbf{T}$ is clash-free.
- For P5, assume that $s, t \in \mathbf{S}$ and $\forall S.C \in \mathcal{L}''(s)$. By the definition of T, we consider the following cases:
  - $\langle s, t\rangle \in \mathcal{E}_1(S)$. This implies that there are x, y such that $s = \varphi(x), t = \varphi(y)$ and $\forall S.C \in \Lambda(\varphi(x))$, $Q \in \Lambda(\langle\varphi(x), \varphi(y)\rangle)$. By the definition of $\Lambda(\langle ., .\rangle)$, there are $x' \in \varphi(x)$ and $y' \in \varphi(y)$ such that $Q \in \mathcal{L}'(\langle x, y\rangle)$ (note that $\mathcal{L}'(\langle ., .\rangle)$ is defined for edges of $\mathbf{T}$ in Definition 6). Due to the non-applicability of $\forall$-rule, it follows that $C \in \mathcal{L}(y')$. By the definition of T and $\mathbf{T}$, we have $C \in \mathcal{L}''(t)$.
  - $\langle s, t\rangle \in \mathcal{E}_2(S)$. This implies that $\forall S.C \in \mathcal{L}(\mathsf{p}(s))$, $Q \in \mathcal{L}'(\langle\mathsf{p}(s), \mathsf{p}(t)\rangle)$. Due to the non-applicability of $\forall$-rule, it follows that $C \in \mathcal{L}(\mathsf{p}(t))$. By the definition of T, we have $C \in \mathcal{L}''(t)$.
  - $\langle s, t\rangle \in \mathcal{E}_3(S)$. This implies that there are x, y such that (i) either $s = \varphi(x), t = \mathsf{b}^1(\varphi(y)) \in \widehat{V}_o^{\varphi(y)}$ with some $o \in \mathbf{C}_o$ and $\forall S.C \in \mathcal{L}(x')$, $Q \in \mathcal{L}'(\langle x', y'\rangle)$ with $x' \in \varphi(x)$ and $y' \in \varphi(y)$. Due to the non-applicability of $\forall$-rule, $C \in \mathcal{L}(y')$, or (ii) $t = \varphi(x)$, $s = \mathsf{b}^1(\varphi(y))$ with $\forall S.C \in \mathcal{L}(y')$, $Q \in \mathcal{L}'(\langle y', x'\rangle)$, $x' \in \varphi(x)$ and $y' \in \varphi(y)$. Due to the non-applicability of $\forall$-rule, it follows that $C \in \mathcal{L}(x')$. By the definition of T, we have $C \in \mathcal{L}''(t)$.
  - $\langle s, t\rangle \in \mathcal{E}_4(S)$. This implies that either (i) $S \in \mathcal{L}(\langle\mathsf{p}(s), w\rangle)$ with $\forall S.C \in \mathcal{L}(\mathsf{p}(s))$ and $t = \varphi(w)$ for some $w \in \bar{\mathbf{I}}$. Due to the non-applicability of $\forall$-rule, $C \in \mathcal{L}(w)$, or (ii) $S^{\ominus} \in \mathcal{L}(\langle\mathsf{p}(t), w\rangle)$ with $\forall S.C \in \mathcal{L}(w)$, $s = \varphi(w)$. Due to the non-applicability of $\forall$-rule, $C \in \mathcal{L}(\mathsf{p}(t))$. By the definition of T, we have $C \in \mathcal{L}''(t)$.
- P6. Analogously.
- P7. Assume that $s \in \mathbf{S}_1$ and $\exists S.C \in \mathcal{L}''(s)$. Let $s = \varphi(x)$ with $\exists S.C \in \mathcal{L}(x)$. This implies that there is a node $x' \in \mathbf{I} \cup \mathbf{B}$ such that $x = \mathsf{pred}^i(x')$ with $i \geq 1$. That means that x is not a leaf node. Due to the non-applicability of $\exists$-rule, x has a S-neighbour y with $C \in \mathcal{L}(y)$. We consider the following cases:
  - $y \in \mathbf{I}$ or $y \in \mathbf{B} \setminus \bar{\mathbf{B}}$. By definition, there is some node $y' \in \varphi(y)$ such that y' is an ancestor of a node $z \in \mathbf{I} \cup \mathbf{B}$. This implies that $\varphi(y) \in \mathbf{S}_1$ with $C \in \mathcal{L}''(\varphi(y))$, and $\langle\varphi(x), \varphi(y)\rangle \in \mathcal{E}_1(R)$.
  - $y \in \bar{\mathbf{B}}$. We have $\mathsf{b}^1(\varphi(y)) \in \mathbf{S}_2$ with $C \in \mathcal{L}''(\mathsf{b}^1(\varphi(y)))$, and $\langle\varphi(x), \mathsf{b}^1(\varphi(y))\rangle \in \mathcal{E}_3(R)$.
  - Otherwise, i.e., $y \notin \mathbf{I} \cup \mathbf{B}$. By definition, we have $\varphi(y) \in \mathbf{S}_1$ with $C \in \mathcal{L}''(\varphi(y))$, and $\langle\varphi(x), \varphi(y)\rangle \in \mathcal{E}_1(R)$.

  Assume that $s \in \mathbf{S}_2$ with $s \in \widehat{V}_o^{\varphi(x)}$ and $\exists S.C \in \mathcal{L}''(s)$. We consider the following cases:
  - $s = \mathsf{b}^1(\varphi(y))$ for some $y \in \bar{\mathbf{B}}$. This implies that $\exists S.C \in \mathcal{L}(\mathsf{b}(y))$. Due to the non-applicability of $\exists$-rule, $\mathsf{b}(y)$ has a S-neighbour y' with $C \in \mathcal{L}(y')$. If y' is a successor of $\mathsf{b}(y)$, by the definition of $\widehat{V}_o^{\varphi(x)}$, s has a S-neighbour

$v' \in \widehat{V}_o^{\varphi(x)}$ such that $C \in \mathcal{L}(v')$. If y' is the predecessor of $\mathsf{b}(y)$ then $C \in \Lambda(\varphi(x))$ and $\langle s, \varphi(x)\rangle \in \mathcal{E}_4(S)$.
  - Otherwise, due to the definition of $\widehat{V}_o^{\varphi(x)}$, s has a S-neighbour $v' \in \widehat{V}_o^{\varphi(x)}$ such that $C \in \mathcal{L}(v')$.
- P8 and P9 holds due to the construction of T.
- For P10, assume that $s \in \mathbf{S}$ with $(\leq nS.C) \in \mathcal{L}''(s)$. We consider the following cases:

  • Assume $s \in \mathbf{S}_1$ i.e. $s = \varphi(x)$ such that $x = \mathsf{pred}^i(x')$, $i \geq 1$ for some $x' \in \mathbf{I} \cup \mathbf{B}$. Assume that there are distinct $s_0, \cdots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}''(s_i)$ and $\langle s, s_i\rangle \in \mathcal{E}(S)$ for $0 \leq i \leq n$.

  Assume that $s_0, \cdots, s_n \in \mathbf{S}_1$. This implies that there are nodes $x_0, \cdots, x_n$ such that $\varphi(x_i) \cap \varphi(x_j) = \emptyset$, $S \in \Lambda(\varphi(x), \varphi(x_i))$ and $C \in \Lambda(\varphi(x_i))$ for $0 \leq i < j \leq n$. Due to the non-applicability of $\leq$-rule, there do not exist y', y'' such that $\varphi(y') \cap \varphi(y'') = \emptyset$, not $\varphi(y') \neq \varphi(y'')$, $S \in \Lambda(\varphi(x), \varphi(y'))$, $S \in \Lambda(\varphi(x), \varphi(y''))$ and $C \in \Lambda(\varphi(y')) \cap \Lambda(\varphi(y''))$. This implies $\varphi(x_i) \neq \varphi(x_j)$ for all $0 \leq i < j \leq n$, which contradicts clash-freeness of $\mathbf{T}$.

  Assume that $s_0, \cdots, s_l \in \mathbf{S}_2$ and $s_{l+1}, \cdots, s_n \in \mathbf{S}_1$ with $0 \leq l < n$. Due to the definition of $\mathbf{S}_1$ there are nodes $w_{l+1}, \cdots, w_n$ such that $\varphi(w_i) \cap \varphi(w_j) = \emptyset$, $C \in \Lambda(\varphi(w_i))$ and $S \in \Lambda(\langle \varphi(x), \varphi(w_i)\rangle)$ for $(l+1) \leq i < j \leq n$.

  Assume that $s_0 \in \widehat{V}_o^{\varphi(y)}$ and $\langle \varphi(x), s_0\rangle \in \mathcal{E}_4(S)$. By the definition of T, this means that $\mathsf{p}(s_0)$ is a $S^{\ominus}$-predecessor of a node $w \in \bar{\mathbf{I}}$ with $w \in \varphi(x)$. According to Property 4 (with $(\leq nS.C) \in \mathcal{L}(w)$, $C \in \mathcal{L}(\mathsf{p}(s_0))$), $\mathsf{p}(s_0)$ is inner since $\mathsf{p}(s_0)$ belongs to the layer 2. It contradicts the definition of $\bar{\mathbf{I}}$ which contains no node whose predecessor is inner. We have shown that if $s_i \in \mathbf{S}_2$ then it is not possible that $\langle \varphi(x), s_i\rangle \in \mathcal{E}_4(S)$ for all $0 \leq i \leq l$. This implies that $\langle \varphi(x), s_i\rangle \in \mathcal{E}_3(S)$ for all $0 \leq i \leq l$.

  According to the definition of T, there nodes $w_0, \cdots, w_l \in \bar{\mathbf{B}}$ such that $\varphi(w_i) \cap \varphi(w_j) = \emptyset$, $C \in \Lambda(\varphi(w_i))$ and $S \in \Lambda(\langle \varphi(x), \varphi(w_i)\rangle)$ for $0 \leq i < j \leq l$. Due to the definition of $\bar{\mathbf{B}}$, we have $\varphi(w_i) \notin \mathbf{S}_1$ for all $0 \leq i \leq l$.

  Since $\mathbf{S}_1 \cap \mathbf{S}_2 = \emptyset$, we have $\varphi(w_i) \cap \varphi(w_j) = \emptyset$ for all $0 \leq i < j \leq n$, which contradicts clash-freeness of $\mathbf{T}$.

  • Assume $s \in \mathbf{S}_2$, i.e., $s \in \widehat{V}_o^{\varphi(x)}$ for some $x \in \bar{\mathbf{B}}$. Assume that there are distinct $s_0, \cdots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}''(s_i)$ and $\langle s, s_i\rangle \in \mathcal{E}(S)$ for $0 \leq i \leq n$.

  Due to the non-applicability of $\leq$-rule, $\mathsf{p}(s)$ has at most n S-neighbours $x_1, \cdots, x_n$ with $C \in \mathcal{L}(x_i)$. Due to the definition of T, there are at most n individuals $s_1, \cdots, s_n$ such that $\langle s, s_i\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}''(s_i)$.
- To show P11, assume that $s \in \mathbf{S}_1$ with $(\geq nR.C) \in \mathcal{L}''(s)$ i.e. $s = \varphi(x)$ with $x = \mathsf{pred}^i(x')$, $i \geq 1$ for some $x' \in \mathbf{I} \cup \mathbf{B}$. Due to the non-applicability of $\geq$-rule, x has n S-neighbours $x_1, \cdots, x_n$ such that $x_i \neq x_j$ with $C \in \mathcal{L}(x_i)$ for $1 \leq i < j \leq n$. This implies $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and for all $1 \leq i < j \leq n$.

  Moreover, if $x_i \in \mathbf{I}$, $x_i \in \mathbf{B} \setminus \bar{\mathbf{B}}$ or $x_i = \mathsf{pred}^k(x')$, $k \geq 1$ for some $x' \in \mathbf{I} \cup \mathbf{B}$ then $\varphi(x_i) \in \mathbf{S}_1$, $\langle \varphi(x), \varphi(x_i)\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}''(\varphi(x_i))$. Otherwise, i.e., if $x_i \in \bar{\mathbf{B}}$, then $\mathsf{b}^1(\varphi(x_i)) \in \mathbf{S}_2$, $\langle \varphi(x), \mathsf{b}^1(\varphi(x_i))\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}''(\mathsf{b}^1(\varphi(x_i)))$. Due to $\mathsf{b}^1(\varphi(x_i)) \neq \mathsf{b}^1(\varphi(x_j))$ if $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and $\mathbf{S}_1 \cap$

$\mathbf{S}_2 = \emptyset$, s has n distinct individuals $s_1, \cdots, s_n$ correponding to $\varphi(x_1), \cdots, \varphi(x_n)$ such that $\langle s, s_i\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}''(s_i)$.

Assume that $s \in \mathbf{S}_2$ with $(\geq nR.C) \in \mathcal{L}''(s)$, i.e., $s \in \widehat{V}_o^{\varphi(x)}$ for some $x \in \bar{\mathbf{B}}$. Due to the non-applicability of $\geq$-rule, x has n S-neighbours $x_1, \cdots, x_n$ such that $x_i \neq x_j$ with $C \in \mathcal{L}(x_i)$ for $1 \leq i < j \leq n$. This implies $\varphi(x_i) \cap \varphi(x_j) = \emptyset$ and for all $1 \leq i < j \leq n$. (we have to distinguish two rules for at most).

– For P12 assume that $s, t \in \mathbf{S}$ with $(\leq nS.C) \in \mathcal{L}''(s)$ and $\langle s, t\rangle \in \mathcal{E}(S)$. Assume $s \in \mathbf{S}_2$, i.e., $s \in \widehat{V}_o^{\varphi(x)}$ for some $x \in \bar{\mathbf{B}}$.
If $s = \mathsf{b}^1(\varphi(x))$ and $t = \varphi(y)$ with $\varphi(y) \in \mathbf{S}_1$ then $\mathsf{b}(x)$ has a S-predecessor y' with $\mathcal{L}(y') = \mathcal{L}(y)$. Due to the non-applicability of ch-rule, $C \in \mathcal{L}(y')$. This implies that $C \in \mathcal{L}''(t)$.
If $s \neq \mathsf{b}^1(\varphi(x))$ and $t \in \mathbf{S}_2$ then $\mathsf{p}(t)$ is a S-neighbour of $\mathsf{p}(s)$. Due to the non-applicability of ch-rule, $C \in \mathcal{L}(\mathsf{p}(t)) = \mathcal{L}''(t)$.
If $s \neq \mathsf{b}^1(\varphi(x))$ and $t \in \mathbf{S}_1$ with $t = \varphi(y)$ and $y \in \bar{\mathbf{I}}$ then y is a S-neighbour of $\mathsf{p}(s)$. Due to the non-applicability of ch-rule, $C \in \mathcal{L}(y) = \mathcal{L}''(t)$.
Assume $s \in \mathbf{S}_1$, i.e., $s = \varphi(x)$ with $x = \mathsf{pred}^i(x')$, $i \geq 1$ for some $x' \in \mathbf{I} \cup \mathbf{B}$. Similarly.
– P13 holds due to the non-applicability 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-applicability of $\bowtie$-rule. □

Lemma 4 (Completeness) *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then Algorithm 3 answers "YES" by yielding a clash-free and complete 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 $\{\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq) \mid o \in \mathbf{C}_o\}$ be a set of $\mathcal{SHOIQ}$-trees. We show that there exists a sequence of expansion rule applications such that it generates a set of clash-free $\mathcal{SHOIQ}$-trees with a partitioning function φ (**).

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

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

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

According to P14 in Definition 4, for each $o \in \mathbf{C}_o$ there is some $s_o \in \mathbf{S}$ such that $o \in \mathcal{L}'(s_o)$. We initialize $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$ and $\pi(\widehat{x}_o) = s_o$ for each $o \in \mathbf{C}_o$.

1. $\sqsubseteq$-rule. Due to P1, we have $\mathsf{nnf}(\neg C \sqcup D) \in \mathcal{L}'(\pi(x))$ for all x. Therefore, if $\mathsf{nnf}(\neg C \sqcup D) \in \mathcal{L}(x')$ for each $x' \in \varphi(x)$ then $\mathsf{nnf}(\neg C \sqcup D) \in \mathcal{L}'(\pi(x))$ due to ($\mathbf{C1}$). Moreover, $\sqsubseteq$-rule adds $\mathsf{nnf}(\neg C \sqcup D)$ to $\mathcal{L}(x')$ for all $x' \in \varphi(x)$ i.e. ($\mathbf{C1}$)

is preserved. Since $(\mathbf{C2})$ and $(\mathbf{C3})$ are trivially preserved, (*) 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. $(\mathbf{C1})$ is preserved. Since $(\mathbf{C2}) - (\mathbf{C4})$ are trivially preserved, (*) 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. $(\mathbf{C1})$ is preserved. Since $(\mathbf{C2}) - (\mathbf{C4})$ are trivially preserved, (*) 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 $(\mathbf{C2}) - (\mathbf{C4})$ 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, $\mathsf{card}\{S^T(\pi(x), C)\} \leq n$. The $\leq$-rule is applicable if x has $(n+1)$ S-neighbours $y_0, \cdots, y_n$ with $C \in \mathcal{L}(y_i)$. Due to $(\mathbf{C3}) - (\mathbf{C4})$ there are $y, z \in \{y_0, \cdots, y_n\}$ such that $\varphi(y) = \varphi(z)$ and $y \neq z$ does not hold. Therefore, y, z can be chosen to merge by $\leq$-rule where y is a successor of x.
   It is obvious that $(\mathbf{C1})$ is preserved. If y was a R-neighbour of x then z is now a R-neighbour of x and $\langle \pi(x), \pi(z) \rangle \in \mathcal{E}(R)$ due to $\pi(y) = \pi(z)$ and (*). This implies that $(\mathbf{C1}) - (\mathbf{C2})$ hold. Moreover, $(\mathbf{C3})$ is preserved since $\varphi(y) \neq \varphi(z)$ does not hold. Due to (*) and $\pi(y) = \pi(z)$, $(\mathbf{C4})$ 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 $(\mathbf{C1})$. 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 $(\mathbf{C1})$. 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, $(\mathbf{C1})$ is preserved. It is obvious that $(\mathbf{C2})$ and $(\mathbf{C4})$ hold since $\pi(x) = \pi(y)$. The condition $(\mathbf{C3})$ holds since $\varphi(y) \neq \varphi(z)$ does not hold. Due to $\pi(x) = \pi(y)$ and (*), $(\mathbf{C4})$ is preserved as well.
10. $\leq_\varphi$-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, $\mathsf{card}\{S^T(\pi(x), C)\} \leq n$. The rule $\leq_\varphi$-rule is applicable if (i) there are $(n+1)$ nodes $y_0, \cdots, y_n$ such that $C \in \Lambda(\varphi(y_i))$, $S \in \Lambda(\langle \varphi(x), \varphi(y_i) \rangle)$ and $\varphi(y_i) \cap \varphi(y_j) = \emptyset$ for all $0 \leq i < j \leq n$. This implies that there are two nodes $y, z \in \{y_0, \cdots, y_n\}$ such that $\pi(y) = \pi(z)$. Thus, $\varphi(y) \neq \varphi(z)$ cannot hold due to (*), and (ii) there are $x', x'' \in \varphi(x)$ with

$x' \neq x''$ such that x' has a S-neighbour $y' \in \varphi(y)$ and x'' has a S-neighbour $z' \in \varphi(z)$ with $C \in \mathcal{L}(y') \cap \mathcal{L}(z')$.
Therefore, $\varphi(y), \varphi(z)$ and x', x'' can be chosen to merge by $\leq_\varphi$-rule. It is obvious that $(\mathbf{C1})$ is preserved. If w' is a R-neighbour 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 $(\mathbf{C2})$ holds. Moreover, $(\mathbf{C3})$ is preserved since $\varphi(y) \neq \varphi(z)$ does not hold. Due to (*) and $\pi(y) = \pi(z)$, $(\mathbf{C4})$ is preserved as well.

It holds that φ is a partitioning function since the non-applicability of $\leq_\varphi$-rule.

We now show that if the $\mathcal{SHOIQ}$-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 $(\mathbf{C1})$.
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 $(\mathbf{C3})$ 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 $x_0, \cdots, x_n \in \bigcup_{o \in \mathbf{C}_o} V_o$ 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 $i \in \{0, \cdots, n\}$. By P10, there are $x_i, x_j \in \{x_0, \cdots, x_n\}, i \neq j$ 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 $(\mathbf{C4})$. This implies that $x' \neq x''$ does not hold due to $(\mathbf{C3})$, which contradicts $\varphi(x_i) \neq \varphi(x_j)$. □

The following theorem is the main result of the present section which is an immediate consequence of Lemmas 2, 3 and 4.

Theorem 6 *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Algorithm 3 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in* 2NEXPTIME *in the size of $(\mathcal{T}, \mathcal{R})$.*

6 A worst-case optimal algorithm for $\mathcal{SHOIQ}$

This section starts by translating results presented in [22] for $\mathcal{C}^2$ into those for $\mathcal{SHOIQ}$.

Definition 8 (star-type) Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A *star-type* is a triplet $\sigma = \langle \lambda_\sigma, \bar{\nu}_\sigma, \bar{\mu}_\sigma \rangle$, where $\lambda_\sigma \in 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$, $\bar{\mu}_\sigma = (\langle r_1, l_1 \rangle, \cdots, \langle r_d, l_d \rangle)$ is a d-tuple over $2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$, and $\bar{\nu}_\sigma = (\langle r', l' \rangle)$ with $\langle r', l' \rangle \in 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$. A pair $\langle r, l \rangle$ is a *ray* of σ if $\langle r, l \rangle$ is a component of $\bar{\mu}_\sigma$ or $\bar{\nu}_\sigma$. In particular, $\langle r, l \rangle$ is a *predecessor ray* if $(\langle r, l \rangle) = \bar{\nu}_\sigma$, and $\langle r, l \rangle$ is a *successor ray* if $\langle r, l \rangle$ is a component of $\bar{\mu}_\sigma$. We denote $\bar{\xi}_\sigma = (\langle r_1, l_1 \rangle, \cdots, \langle r_d, l_d \rangle, \langle r_{d+1}, l_{d+1} \rangle)$ if $\bar{\nu}_\sigma = (\langle r', l' \rangle)$ where $r' = r_{d+1}$, $l' = l_{d+1}$, and $\bar{\xi}_\sigma = \bar{\mu}_\sigma$ if $\bar{\nu}_\sigma$ is empty.

- A ray $\langle r', l' \rangle$ of σ is *primary* w.r.t. a term $(\leq mR.C)$ if $(\leq mR.C) \in \lambda_\sigma$, $R \in r'$ and $C \in l'$. For a term $(\leq mR.C) \in \lambda_\sigma$, we denote $\mathcal{C}^\sigma_{\langle \leq mR.C \rangle}$ for the set of all rays $\langle r', l' \rangle$ of σ such that $R \in r'$, $C \in l'$.
- A star-type σ is *nominal* if $o \in \lambda_\sigma$ for some $o \in \mathbf{C}_o$.

- A star-type σ is *chromatic* if there is a term $(\geq nS.D) \in \lambda_\sigma$ and σ has n rays $\langle r'_1, l'_1\rangle, \cdots, \langle r'_n, l'_n\rangle$ such that $S \in l'_i$, $D \in l'_i$ for all $1 \leq i \leq n$, and $l'_i \neq l'_j$ for all $0 \leq i < j \leq n$ with $l'_0 = \lambda_\sigma$.
- A star-type σ is *homomorphic* (resp. *isomorphic*) to a star-type σ' if $\lambda_\sigma = \lambda_{\sigma'}$, and for each term $(\leq mR.C) \in \lambda_\sigma$, there is an injection (resp. a bijection) $\pi : \mathcal{C}^{\sigma}_{\langle \leq mR.C\rangle} \rightarrow \mathcal{C}^{\sigma'}_{\langle \leq mR.C\rangle}$ such that $\pi(\langle r, l\rangle) = \langle r', l'\rangle$ implies $r' = r$ and $l' = l$.
- Two star-types σ, σ' are *equivalent* if $\lambda_\sigma = \lambda_{\sigma'}$, and there is a bijection π between $\bar{\xi}_\sigma$ and $\bar{\xi}_{\sigma'}$ such that $\pi(\langle r, l\rangle) = \langle r', l'\rangle$ implies $r' = r$ and $l' = l$.

We denote Σ for the set of all star-types for $(\mathcal{T}, \mathcal{R})$. $\lhd$

In the context of a $\mathcal{SHOIQ}$-forest, we can think of a star-type σ as the set of nodes x such that $\mathcal{L}(x) = \lambda_\sigma$, and each ray $\langle r_i, l_i\rangle$ of σ corresponds to a neighbour x_i of x such that $\mathcal{L}'(\langle x, x_i\rangle) = r_i$ and $\mathcal{L}(x_i) = l_i$. In this case, we say that x *satisfies* σ.

Remark 1 The notion of chromaticity introduced in Definition 8 implies an inequality relation $\neq$ over nodes. That stronger notion is needed to prevent "distinct" star-types from including nodes x, y which are neighbours or $x \not\doteq y$. In order to make star-types chromatic, it is necessary to add to knowledge bases some new concepts and axioms as follows. Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. For each term $(\geq nS.D) \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$, we add to $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ n new concept names $C^0_{(\geq nS.D)}, \cdots, C^n_{(\geq nS.D)}$, and to $\mathcal{T}$ the following axioms: $C^i_{(\geq nS.D)} \sqcap C^j_{(\geq nS.D)} \sqsubseteq \bot$ for all $0 \leq i < j \leq n$. It is straightforward to prove that the terminology $(\mathcal{T}', \mathcal{R})$ is consistent iff $(\mathcal{T}, \mathcal{R})$ is consistent where $\mathcal{T}'$ is obtained from $\mathcal{T}$ by adding these new axioms. Thanks to these new concepts and axioms, the following definition points out how to build chromatic star-types.

Definition 9 (valid star-type) Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Let σ be a star-type for $(\mathcal{T}, \mathcal{R})$ where $\sigma = \langle \lambda_\sigma, \bar{\nu}, \bar{\mu}\rangle$ with $\bar{\mu} = (\langle r_1, l_1\rangle, \cdots, \langle r_d, l_d\rangle)$ and $\lambda_\sigma = l_0$, $\bar{\nu} = \{\langle r_{d+1}, l_{d+1}\rangle\}$. σ is *valid* if the following conditions are satisfied:

1. If $C \sqsubseteq D \in \mathcal{T}$ then $\mathsf{nnf}(\neg C \sqcup D) \in l_i$ for all $0 \leq i \leq d+1$;
2. $\{A, \neg A\} \not\subseteq l_i$ for every concept name A with $0 \leq i \leq d+1$;
3. If $C_1 \sqcap C_2 \in l_i$ then $\{C_1, C_2\} \subseteq l_i$ for all $0 \leq i \leq d+1$;
4. If $C_1 \sqcup C_2 \in l_i$ then $\{C_1, C_2\} \cap l_i \neq \emptyset$ for all $0 \leq i \leq d+1$;
5. If $\exists R.C \in \lambda_\sigma$ then there is some $1 \leq i \leq d+1$ such that $C \in l_i$ and $R \in r_i$;
6. If $(\leq nS.C) \in \lambda_\sigma$ and there is some $1 \leq i \leq d+1$ such that $S \in r_i$ then $C \in l_i$ or $\dot{\neg} C \in l_i$;
7. If $(\leq nS.C) \in \lambda_\sigma$ and there is some $1 \leq i \leq d+1$ such that $C \in l_i$ and $S \in r_i$ then there is some $1 \leq m \leq n$ such that $\{(\leq mS.C), (\geq mS.C)\} \subseteq \lambda$;
8. For each $1 \leq i \leq d+1$, if $R \in r_i$ and $R \boxtimes S$ then $S \in r_i$;
9. If $\forall R.C \in \lambda_\sigma$ and $R \in r_i$ for some $1 \leq i \leq d+1$ then $C \in l_i$;
10. If $\forall R.D \in \lambda_\sigma$, $S \boxtimes R$, $\mathsf{Trans}(S)$ and $R \in r_i$ for some $1 \leq i \leq d+1$ then $\forall S.D \in l_i$;
11. If $(\geq nS.C) \in \lambda_\sigma$ then $C^0_{(\geq nS.C)} \in \lambda_\sigma$ and there are $1 \leq i_1 < \cdots < i_n \leq d+1$ such that $\{C, C^j_{(\geq nS.C)}\} \subseteq l_{i_j}$, $S \in r_{i_j}$ for all $1 \leq j \leq n$.

12. If $(\leq nS.C) \in \lambda_\sigma$ and there are no $1 \leq i_1 < \cdots < i_{n+1} \leq d+1$ such that $C \in l_{i_j}$ and $S \in r_{i_j}$ for all $1 \leq j \leq n+1$;

We denote $\bar{\Sigma}$ for the set of all valid star-types for $(\mathcal{T}, \mathcal{R})$. $\lhd$

Notice that a valid star-type according to Definition 9 is chromatic. If we think of a star-type σ as a node x satisfying σ in a $\mathcal{SHOIQ}$-forest then σ is valid if no expansion rule is applicable to x. Moreover, due to the conditions 7, 11 and 12 in Definition 9, if there is a term $(\geq nS.D) \in \lambda_\sigma$ for a valid star-type σ then σ has exactly n primary rays $\langle r_i, l_i\rangle, \cdots, \langle r_n, l_n\rangle$ w.r.t. $(\leq nS.D)$.

Definition 10 (frame) Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A *frame* for $(\mathcal{T}, \mathcal{R})$ is a tuple $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi\rangle$, where

- $H \in \mathbb{N}$ is the *dimension* of $\mathcal{F}$;
- $\mathcal{N}_i \subseteq \Sigma$ for all $0 \leq i \leq H$, and all star-types in $\mathcal{N}_0$ are nominal. We denote $\mathfrak{N} = \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i$;
- δ is a function $\delta : \mathfrak{N} \to \mathbb{N}$;
- Φ is a function $\Phi : \mathfrak{N} \to 2^{\mathfrak{N}}$

such that the following conditions are satisfied:

1. For each star-type $\sigma \in \mathcal{N}_h$ with $\delta(\sigma) > 0$, $0 \leq h \leq H$, and for each ray $\langle r, l\rangle$ of σ, there are star-types $\sigma_1, \cdots, \sigma_k \in \mathcal{N}_l$, and integers denoted by $0 < \delta(\sigma, \langle r, l\rangle, \sigma_i) \leq \delta(\sigma_i)$ $(1 \leq i \leq k)$ where $l = h - 1$ if $\langle r, l\rangle$ is a predecessor ray of σ and $l = h + 1$ if $\langle r, l\rangle$ is a successor ray of σ, such that
   - $\delta(\sigma) = \sum_{1\leq i\leq k} \delta(\sigma, \langle r, l\rangle, \sigma_i)$, and
   - each σ_i $(1 \leq i \leq k)$ has a ray $\langle r_i, l_i\rangle$ with $l = \lambda_{\sigma_i}$, $l_i = \lambda_\sigma$ and $r = r_i^-$ where $r_i^- = \{R^\ominus \mid R \in r_i\}$.

   We denote $\Omega(\sigma, \langle r, l\rangle) = \{\sigma_1, \cdots, \sigma_k\}$ and say that σ is *linkable* with $\sigma_1, \cdots, \sigma_k$ by $\langle r, l\rangle$.
2. For two star-types $\sigma, \sigma' \in \mathfrak{N}$, if $\sigma' \in \Phi(\sigma)$ then either σ is isomorphic to σ', or there is a star-type $\omega \in \mathfrak{N} \setminus \mathcal{N}_H$ such that σ and σ' are homomorphic to ω.

Remark 2 For $\sigma, \sigma' \notin \mathcal{N}_H$ such that σ, σ' are valid, if σ is homomorphic to σ' then σ is isomorphic to σ'. In fact, if there is a term $(\leq mR.C) \in \lambda_\sigma$ then both σ, σ' have exactly m primary rays w.r.t. $(\leq mR.C)$. If there is a homomorphism between the two sets of primary rays w.r.t. $(\leq mR.C)$ then it is an isomorphism as well.

The frame structure, as introduced in Definition 10, allows us to tile a forest structure by star-types. Such a structure is crucial to obtain termination when designing a tableau-based algorithm. An important difference between a frame and a $\mathcal{SHOIQ}$-forest is that a frame does not represent nodes corresponding to individuals but stores the number of individuals satisfying a star-type. The function $\delta(\sigma)$ is used for this purpose. The condition 1 in Definition 10 introduces the relation of linkability for star-types while the condition 2 provides a characteristic of Φ which regroups "specific" homomorphic star-types for representing a partition in the context of a $\mathcal{SHOIQ}$-forest. Notice that the chromaticity of star-types prevents chromatically linakble star-types from collapsing into a unique star-type by the function Φ.

Definition 11 (valid frame) Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A frame $\mathcal{F}=\langle(\mathcal{N}_0,\cdots,\mathcal{N}_H),\delta,\Phi\rangle$ is *valid* if the following conditions are satisfied:

1. For each $\sigma\in\mathfrak{N}$, if $\delta(\sigma)\geq 1$ then σ is valid and $\sigma\in\Phi(\sigma)$;
2. For each $o\in\mathbf{C}_o$ there is a star-type $\sigma\in\mathcal{N}_0$ such that $o\in\lambda_\sigma$ and $\delta(\sigma)=1$;
3. For two star-types $\sigma,\sigma'\in\mathfrak{N}$ with $\delta(\sigma)\geq 1$, $\delta(\sigma')\geq 1$, if $o\in\lambda_\sigma\cap\lambda_{\sigma'}$ with some $o\in\mathbf{C}_o$ then $\sigma'\in\Phi(\sigma)$;
4. For each $0\leq k<H$ and $\langle\lambda,r,\lambda'\rangle\in 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}\times 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}}\times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ with $r^-=\{R^\ominus\mid R\in r\}$,
$$\sum_{\sigma\in\mathcal{N}_k}\delta(\sigma)|\bar{\mu}_\sigma|_{\langle\lambda,r,\lambda'\rangle}=\sum_{\sigma'\in\mathcal{N}_{k+1}}\delta(\sigma')|\bar{\nu}_{\sigma'}|_{\langle\lambda',r^-,\lambda\rangle}$$
where $|\bar{\nu}_\sigma|_{\langle\lambda,r,\lambda'\rangle}$ and $|\bar{\mu}_\sigma|_{\langle\lambda,r,\lambda'\rangle}$ are denoted for the number of components $\langle r',l'\rangle$ of respective $\bar{\nu}_\sigma$ and $\bar{\mu}_\sigma$ such that $\lambda_\sigma=\lambda$, $r'=r$ and $l'=\lambda'$;
5. For two star-types $\sigma,\sigma'\in\Phi(\sigma')$ which have respective primary ray $\langle r,l\rangle$ and $\langle r',l'\rangle$ w.r.t. $(\leq m.R.C)$ such that $r=r'$ and $l=l'$, if there are two star-types $\omega,\omega'\in\mathfrak{N}$ such that σ and σ' are linkable with respective ω and ω' by $\langle r,l\rangle$ and $\langle r',l'\rangle$ then $\omega\in\Phi(\omega')$.

The notion of validity for a frame is crucial to establish a connection with the tableau-based algorithm presented in Section 5, i.e., how to build a $\mathcal{SHOIQ}$-forest from a valid frame, and inversely. The condition 1 in Definition 11 requires that every star-type satisfied by at least one node must be valid. The condition 2 ensures that each nominal is counted exactly once by δ while the condition 3 imposes that all nominal star-types containing some $o\in\mathbf{C}_o$ are regrouped into a unique partition of star-types by Φ. In the context of a $\mathcal{SHOIQ}$-forest, these conditions imply that for each nominal $o\in\mathbf{C}_o$ there is exactly one tree whose root contains o and there is exactly one partition containing o. The condition 4 allows for linking star-types at level k with star-types at level $k-1$ and $k+1$. It ensures that each node x satisfying (or counted for) a star-type σ at level k is linked by its rays to neighbours satisfying star-types at level $k-1$ and $k+1$. The number of these neighbours corresponds exactly to the number of σ's rays. Finally, the condition 5 deals with partitions. The definition of Φ is based on the two following notions : homomorphism and linkability. That means that from a partition $\Phi(\sigma)$ of star-types we can build a "neighbour" partition $\Phi(\sigma')$ including homomorphic star-types which are linkable with star-types included in $\Phi(\sigma)$. Notice that in the context of a $\mathcal{SHOIQ}$-forest, the definition of Φ does not imply that two nodes satisfying a star-type belong to a partition of nodes.

Lemma 5 *Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base.*

1. *If Algorithm 3 can build a clash-free $\mathcal{SHOIQ}$-forest for $(\mathcal{T},\mathcal{R})$ then there is a valid frame for $(\mathcal{T},\mathcal{R})$.*
2. *If there is a valid frame $\mathcal{F}=\langle(\mathcal{N}_0,\cdots,\mathcal{N}_H),\delta,\Phi\rangle$ for $(\mathcal{T},\mathcal{R})$ then Algorithm 3 can build a clash-free $\mathcal{SHOIQ}$-forest for $(\mathcal{T},\mathcal{R})$.*

Proof • Assume that Algorithm 3 builds a clash-free $\mathcal{SHOIQ}$-forest $\mathbf{G}=\langle\mathbf{T},\varphi\rangle$ for $(\mathcal{T},\mathcal{R})$. To obtain a frame whose dimension $H=\mathsf{max}\{\mathsf{h}(\mathsf{pred}(x))\mid x\text{ is blocked}\}$

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

1 Guess a number H with $H \leq K$ where $K = 2^{2m+k} \times 2$ with $m = \mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\}$ and $k = \mathsf{card}\{\mathbf{R}_{(\mathcal{T},\mathcal{R})}\}$;
2 For each $\sigma \in \Sigma$, guess a number $\delta(\sigma) < M^{2^{2m+k}\times 2}$ where $M = \sum m_i + E$, m_i occurs in a number restriction term $(\geq m_i R.C)$ appearing in $\mathcal{T}$, and E is the number of distinct terms $\exists R.C$ appearing in $\mathcal{T}$;
3 Build a frame $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi\rangle$ for $(\mathcal{T}, \mathcal{R})$ with the guessed H and $\delta(\sigma)$ for $\sigma \in \Sigma$;
4 **if** $\mathcal{F}$ *is valid* **then**
5 | **return** YES ;
6 **else**
7 | **return** NO ;

Algorithm 4: An optimal worst-case algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base

where $\mathsf{h}(y)$ is denoted for the level of a node y, we can extend each tree $\mathbf{T}_o \in \mathbf{T}$ by unravelling each blocked node x until the level H (cf. the proof of Lemma 3).

For each level, we regroup nodes x which satisfy a same star-type σ (i.e. all x have a same label, and for each ray $\langle r, l\rangle$ of σ, x has a unique neighbour y such that $\mathcal{L}'(\langle x, y\rangle) = r$ and $l = \mathcal{L}(y)$). We denote $\sigma(x) = \sigma$ if x is regrouped into σ. This allows us to determine $\delta(\sigma)$ for each $\sigma \in \mathcal{N}_k$ where $\mathcal{N}_k$ is the set of all star-types at level k of $\mathbf{G}$, $k \leq 2^{(2m+k)} \times 2$. Moreover, we can define that $\sigma(x)$ is linkable with $\sigma(x')$ by a ray $\langle r, l\rangle$ iff x' is a neighbour of x such that $\mathcal{L}'(\langle x, x'\rangle) = r$ and $\mathcal{L}(x') = l$. From the defined star-types $\sigma(x)$, we can define a function Φ as follows:

- We initialize $\Phi(\sigma(x)) := \{\sigma(x)\}$ for each node x;
- For each node x with $o \in \mathcal{L}(x) \cap \mathbf{C}_o$, we define $\Phi(\sigma(x)) = \bigcup_{x' \in \varphi(x)} \Phi(\sigma(x'))$
- For each $\Phi(\sigma(w))$ and for each $\langle s, \lambda\rangle \in 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ such that there is a star-type $\sigma(x) \in \Phi(\sigma(w))$ which has a primary ray $\langle r, l\rangle$ with $r = s$ and $l = \lambda$, we define $\Phi(\sigma(y))$ to be the union of all $\Phi(\sigma(y'))$ such that there is a star-type $\sigma(x') \in \Phi(\sigma(w))$ which is linkable with $\sigma(y')$ by a primary ray $\langle r', l'\rangle$ with $r' = s$ and $l' = \lambda$.

By the definition of Φ and the construction of $\mathbf{G}$, it holds that $x' \in \varphi(x)$ implies $\sigma(x') \in \Phi(\sigma(x))$. Moreover, Φ satisfies the condition 2 in Definition 10 since $\sigma(x') \in \Phi(\sigma(x))$ implies that either $\sigma(x')$ is isomorphic to $\sigma(x)$ (if x, x are simultaneously blocked or non-blocked), or $\sigma(x)$ is homomorphic to $\sigma(x')$ (if x is blocked). It is trivial to show that Φ satisfies the conditions 1, 3 and 5 in Definition 11.
• Assume that $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi\rangle$ is a valid frame for $(\mathcal{T}, \mathcal{R})$. Due to the condition 2 in Definition 11, for each $o \in \mathbf{C}_o$ there is a unique star-type $\sigma_o \in \mathcal{N}_0$ such that $o \in \lambda_{\sigma_o}$ and $\delta(\sigma_o) = 1$. For each σ_o we add a root node $\widehat{x}_o$ to a tree $\mathbf{T}_o$ with $\mathcal{L}_o(\widehat{x}_o) = \lambda_{\sigma_o}$. We initialize $\varphi(\widehat{x}_o) = \{\widehat{x}_o\}$. We denote $\sigma(x) := \sigma$ for each node x added to $\mathbf{T}_o$ such that x satisfies σ.

Let $0 \leq k < H$ such that for each $i < k$ and for each star-type $\sigma' \in \mathcal{N}_i$ there are $\delta(\sigma')$ nodes added to $\mathbf{T}_o$ with $\delta(\sigma') > 0$. Due to the condition 2 in Definition 10 and the condition 4 in Definition 11, there are $\omega_1, \cdots, \omega_h \in \mathcal{N}_{k+1}$ and

integers $0 < \widehat{\alpha}(\sigma, \langle r, l\rangle, \omega_i) \leq \delta(\omega_i)$ such that $\lambda_{\omega_i} = l$, $\langle r^-, \lambda_{\sigma_k}\rangle \in \bar{\nu}(\omega_i)$ and $\sum_{1\leq i\leq h} \widehat{\alpha}(\sigma, \langle r, l\rangle, \omega_i) = \delta(\sigma(x))$. For each ω_i with $1 \leq i \leq h$, we add $\widehat{\alpha}(\sigma, \langle r, l\rangle, \omega_i)$ successors y_j of x such that $\sigma(y_j) = \omega_i$ and define $\mathcal{L}(y_j) = \lambda_{\omega_i}$, $\mathcal{L}(\langle x, y_j\rangle) = r$ for $1 \leq j \leq \widehat{\alpha}(\sigma, \langle r, l\rangle, \omega_i)$.

Let Ψ be a set of all nodes x added from level 0 to level k. We define a partitioning function φ over Ψ as follows:

- $x \in \varphi(x)$;
- If $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$ then $x' \in \varphi(x)$;
- For $x', x'' \in \varphi(x)$ which have neighbours y', y'' such that $\Phi(\sigma(x))$ has a primary ray $\langle r, l\rangle$ with $\mathcal{L}'(\langle x', y'\rangle) = r$, $\mathcal{L}'(\langle x'', y''\rangle) = r$, $\mathcal{L}(y') = l$ and $\mathcal{L}(y'') = l$, if $\Phi(\sigma(y')) = \Phi(\sigma(y''))$ then $y' \in \varphi(y'')$.

We show that $x' \in \varphi(x)$ implies $\Phi(\sigma(x')) = \Phi(\sigma(x))$ (*). Assume that $x' \in \varphi(x)$, $\Phi(\sigma(x')) = \Phi(\sigma(x))$ and $(\leq mR.C) \in \lambda_{\sigma(x)}$. Let y, y' be two R-neighbours of respective x, x' such that $C \in \mathcal{L}(y) \cap \mathcal{L}(y')$, $\mathcal{L}(y) = \mathcal{L}(y')$ and $\mathcal{L}'(\langle x, y\rangle) = \mathcal{L}'(\langle x', y'\rangle)$. Since $\sigma(x'), \sigma(x)$ are linkable with $\sigma(y'), \sigma(y)$, by the the condition 5 in Definition 11, we have $\Phi(\sigma(y)) = \Phi(\sigma(y'))$.

Let x be a non-blocked node such that $\sigma(x) \in \mathcal{N}_k$ and $\sigma(\mathsf{pred}^1(x)) \in \mathcal{N}_{k-1}$. Let $\langle r, l\rangle$ be a component of $\bar{\mu}(\sigma(x))$. We will define successors of x.

- Assume that there is no term $(\leq mR.C) \in \lambda_{\sigma(x)}$ with $R \in r$, $C \in l$.
  - Assume that $\mathbf{C}_o \cap l = \emptyset$. We define $\varphi(y_j) := \{y_j\}$. Assume that there is a term $(\leq mR.C) \in l$ with $R \in r^-$, $C \in \mathcal{L}(x)$. Since $\varphi(y_j)$ has a unique neighbour partition $\varphi(x)$ and $m > 0$, the condition 5 in Definition 6 holds.
  - Assume that $o' \in \mathbf{C}_o \cap l$. We define $\varphi(y') := \varphi(\widehat{x}_{o'}) \cup \{y_j\}$ for each $y' \in \varphi(\widehat{x}_{o'}) \cup \{y_j\}$.
    Assume that there is a term $(\leq mR.C) \in l$ with $R \in r^-$, $C \in \mathcal{L}(x)$. We will show that $\varphi(\widehat{x}_{o'})$ has m R neighbour partitions containing C. By construction, $\widehat{x}_{o'}$ has m R-successors $w_1, \cdots, w_m$ containing C. Due to the chromaticity, it follows that $\Phi(\sigma(w_i)) \neq \Phi(\sigma(w_{i'}))$ for all $1 \leq i < i' \leq m$. Moreover, due to the condition 3 in Definition 11, we have $\Phi(\sigma(y_j)) = \Phi(\sigma(\widehat{x}_{o'}))$. By the condition 2 in Definition 10, $\sigma(y_j)$ is homomorphic to $\sigma(\widehat{x}_{o'})$. This implies that there is some $1 \leq k \leq m$ such that $\mathcal{L}(w_k) = \mathcal{L}(x)$ and $\mathcal{L}'(\langle \widehat{x}_{o'}, w_k\rangle) = \mathcal{L}'(\langle y_j, x\rangle)$. By the condition 5 in Definition 11, we have $\Phi(\sigma(x)) = \Phi(\sigma(w_k))$. This implies $x \in \varphi(w_k)$ according to the definition of φ as described above. Thus, the condition 5 in Definition 6 holds.
- Assume that there is a term $(\leq mR.C) \in \lambda_{\sigma(x)}$ with $R \in r$, $C \in l$.
  - Assume that $\mathbf{C}_o \cap l = \emptyset$. Assume that there is no node $w \neq x$ that was added to $\mathbf{T}_o$ at a level $i \leq k$ such that $w \in \varphi(x)$. We define $\varphi(y_j) := \{y_j\}$. It follows that $\varphi(x)$ has at most m R-neighbour partitions containing C since $\sigma(x)$ has m primary rays w.r.t. $(\leq mR.C)$. Assume that there is a term $(\leq nS.D) \in \mathcal{L}(y_j)$ with $R \in r^-$, $C \in \mathcal{L}(x)$. In this case, $\varphi(y_j)$ has at most $n > 0$ S-neighbour partitions containing D since $\varphi(y_j)$ has a unique neighbour partition $\varphi(x)$.
    Assume that there is a node $w \neq x$ that was added to $\mathbf{T}_o$ at a level $i \leq k$ such that $w \in \varphi(x)$ and w has a R-neighbour w' such that containing C

such that $\mathcal{L}(w') = \mathcal{L}(y_j)$ and $\mathcal{L}'(\langle w, w'\rangle) = \mathcal{L}'(\langle x, y_j\rangle)$. This implies that $\sigma(w)$ and $\sigma(x)$ are linkable with respective $\sigma(w')$ and $\sigma(y_j)$. According to (*) we have $\sigma(w) \in \Phi(\sigma(x))$. By the condition 5 in Definition 11, we have $\sigma(w') \in \Phi(\sigma(y_j))$. According to the definition of φ as described above, we have $\varphi(y_j) = \varphi(w') \cup \{y_j\}$. This implies that the condition 5 in Definition 6 holds.

– Assume that $o' \in \mathbf{C}_o \cap l$. Due to the condition 3 in Definition 11, we have $\sigma(y_j) \in \Phi(\sigma(\widehat{x}_{o'}))$ where $\widehat{x}_{o'}$ is the root of $\mathbf{T}_{o'}$. We define $\varphi(y') := \varphi(\widehat{x}_{o'}) \cup \{y_j\}$ for each $y' \in \varphi(\widehat{x}_{o'}) \cup \{y_j\}$.
  Assume that there is no node $w \neq x$ that was added to $\mathbf{T}_o$ at a level $i \leq k$ such that $w \in \varphi(x)$. It follows that $\varphi(x)$ has at most m R-neighbour partitions containing C since $\sigma(x)$ has m primary rays w.r.t. ($\leq mR.C$). Assume that there is a term $(\leq nS.D) \in \mathcal{L}(y_j)$ with $R \in r^-$, $D \in \mathcal{L}(x)$. By the condition 5 in Definition 11, we have $\sigma(x) \in \Phi(\sigma(w_i))$ where w_i is successor of $\widehat{x}_{o'}$ such that $\mathcal{L}(w_i) = \mathcal{L}(x)$ and $\mathcal{L}'(\langle y_j, x\rangle) = \mathcal{L}'(\langle \widehat{x}_{o'}, w_i\rangle)$. According to the definition of φ as described above, we have $\varphi(x) = \varphi(w_i)$. This implies that the condition 5 in Definition 6 holds.
  Assume that there is a node $w \neq x$ that was added to $\mathbf{T}_o$ at a level $i \leq k$ such that $w \in \varphi(x)$ and w has a R-neighbour w' such that containing C such that $\mathcal{L}(w') = \mathcal{L}(y_j)$ and $\mathcal{L}'(\langle w, w'\rangle) = \mathcal{L}'(\langle x, y_j\rangle)$. According to (*) we have $\sigma(w) \in \Phi(\sigma(x))$. By the condition 5 in Definition 11, we have $\sigma(w') \in \Phi(\sigma(y_j))$. According to the definition of φ as described above, we have $\varphi(y_j) = \varphi(w') \cup \{y_j\}$. This implies that the condition 5 in Definition 6 holds.

This process of construction terminates at a node x when the blocking condition for a $\mathcal{SHOIQ}$-forest is satisfied for x. Notice that the blocking condition for a frame implies the blocking condition for $\mathcal{SHOIQ}$-forests. By the construction of $\mathbf{T}_o$ and φ as described above, the conditions 1-5 in Definition 6 hold. In particular, the condition 6 in Definition 6 is a consequence of the chromaticity of star-types, □

Lemma 6 *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. The size of a valid frame $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi\rangle$ is bounded by an exponential function in the size of $(\mathcal{T}, \mathcal{R})$.*

Proof We have $H \leq K$ where $K = 2^{2m+k} \times 2$ with $m = \mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\}$ and $k = \mathsf{card}\{\mathbf{R}_{(\mathcal{T},\mathcal{R})}\}$. Moreover, each valid star-type has at most M distinct rays where $M = \sum m_i + E$, m_i occurs in a number restriction term ($\geq m_i R.C$) appearing in $\mathcal{T}$, and E is the number of distinct terms $\exists R.C$ appearing in $\mathcal{T}$. If we denote Σ for the set of all star-types then $\mathsf{card}\{\Sigma\} \leq (\mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\})^2 \times \mathsf{card}\{\mathbf{R}_{(\mathcal{T},\mathcal{R})}\})^M$. Since $\delta(\sigma)$ is bounded by $M\delta(\sigma')$ where σ', σ are respective star-types at level $k-1$ and k, it holds that $\delta(\sigma) \leq M^{2^{2m+k}\times 2}$. If $\delta(\sigma)$ is represented as a binary number then it takes an exponential number of bits. In addition, since the function Φ partitions $\mathfrak{N}$ we have the number of partitions is bounded by $\mathsf{card}\{\mathfrak{N}\}$ where $\mathfrak{N} = \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i$.

In addition, for each $\sigma \in \mathfrak{N}$ and for each ray $\langle r, l\rangle$ of σ, the cardinality of two sets $\mathcal{N}(\sigma, \langle r, l\rangle)$ and $\widehat{\mathcal{N}}(\sigma, \langle r, l\rangle)$ of linkable star-types is bounded by $\mathsf{card}\{\mathfrak{N}\}$. □

The following theorem affirms that we can obtain a worst-case optimal algorithm for $\mathcal{SHOIQ}$ which is founded on Lemma 5 and 6.

Theorem 7 *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Algorithm 4 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in* NEXPTIME.

Proof According to Theorem 7, $(\mathcal{T}, \mathcal{R})$ is consistent iff there is a clash-free $\mathcal{SHOIQ}$-forest $\langle \mathbf{T}, \varphi \rangle$. Due to Lemma 5 and 6, it suffices to show that all conditions in Definition 10 and Definition 11 can be checked exponentially. In fact, checking the condition 1 in Definition 10 for linkability is exponentially since the cardinality of two sets $\mathcal{N}(\sigma, \langle r, l \rangle)$ and $\widehat{\mathcal{N}}(\sigma, \langle r, l \rangle)$ of linkable star-types is bounded by $\mathsf{card}\{\mathfrak{N}\}$. Moreover, since Φ partitions the set of all star-types, the number of distinct $\Phi(\sigma)$ is bounded by $\mathsf{card}\{\mathfrak{N}\}$ where $\mathfrak{N} = \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$.

Since we have at most an exponential number of distinct star-types, an exponential number of distinct triplet $\langle \lambda, r, \lambda' \rangle \in 2^{\mathsf{cl}(\mathcal{T}, \mathcal{R})} \times 2^{\mathbf{R}_{(\mathcal{T}, \mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T}, \mathcal{R})}$, checking the conditions in Definition 11 is exponential. □

7 Conclusion and Discussion

We have presented in this paper a practical EXPSPACE decision procedure for the logic $\mathcal{SHOIQ}$. The construction of this algorithm is founded on the well-known results for $\mathcal{SHOIQ}$ and $\mathcal{C}^2$. First, we have based our approach on a technique that constructs tree-like structures for representing a model. This allows us to reuse the standard blocking technique over these tree-like structures to obtain termination. Second, we have transferred to $\mathcal{SHOIQ}$ the method used for constructing a NEXPTIME algorithm for $\mathcal{C}^2$. This enables us to represent a double exponential $\mathcal{SHOIQ}$-forest by an exponential structure. This result allows us to devise a worst-case optimal algorithm for deciding consistency of a $\mathcal{SHOIQ}$ knowledge base.

References

1. Baader, F., Nutt, W.: Basic description logics. In: The Description Logic Handbook: Theory, Implementation and Applications (2nd edition), pp. 47–104. Cambridge University Press (2007)
2. Baader, F., P., H.: A scheme for integrating concrete domains into concept languages. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91, pp. 452–457. Sydney (Australia) (1991)
3. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica **69**, 5–40 (2001)
4. De Giacomo, G., Lenzerini., M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proceedings of the 12th National conference on Artificial Intelligence, pp. 205–212. The MIT Press (1994)
5. Donini, F.: Complexity of reasoning. In: The Description Logic Handbook: Theory, Implementation and Applications (2nd edition), pp. 105–148. Cambridge University Press (2007)
6. Fischer, M.J., Ladner, R.I.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences **18**(18), 174–211 (1979)
7. Francesco M. Donini, F.M.: Exptime tableaux for alc. Artificial Intelligence **124**(1), 87–138 (2000)
8. Goré, R., Nguyen, L.A.: Exptime tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In: TABLEAUX, pp. 133–148 (2007)
9. Haarslev, V., Moller, R.: Racer system description. In: Proc. of the Int. Joint on Automated Reasoning (IJCAR 2001), pp. 701–705. Springer (2001)
10. Horrocks, I.: The FaCT system. In: H. de Swart (ed.) Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98), *Lecture Notes in Artificial Intelligence*, vol. 1397, pp. 307–312. Springer (1998). URL download/1998/Horr98b.pdf

11. Horrocks, I., Kutz, O., Sattler, U.: The irresistible $\mathcal{SRIQ}$. In: Proc. of the First Int. Workshop on OWL Experiences and Directions (OWLED 2005) (2005). URL download/2006/HoKS05a.pdf
12. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006). URL download/2006/HoKS06a.pdf
13. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal Of Automated Reasoning **39**(3), 249–276 (2007)
14. Horrocks, I., Sattler, U., Tobies, S.: 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 (1999)
15. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999). Springer (1999)
16. Hustadt, U., Motik, B., Sattler, U.: A Decomposition Rule for Decision Procedures by Resolution-based Calculi. In: F. Baader, A. Voronkov (eds.) Proc. of the 11th Int. Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2004), *LNAI*, vol. 3452, pp. 21–35. Springer, Montevideo, Uruguay (2005)
17. Kazakov, Y., Motik, B.: A Resolution-Based Decision Procedure for SHOIQ. Journal of Automated Reasoning **40**(2–3), 89–116 (2008)
18. Le Duc, C.: Decidability of $\mathcal{SHI}$ with transitive closure of roles. In: Proceedings of the European Semantic Web Conference. Springer-Verlag (2009)
19. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research **36**, 165–228 (2009). URL download/2009/MoSH09a.pdf
20. Ortiz, M.: An automata-based algorithm for description logics around $\mathcal{SRIQ}$. In: Proceedings of the fourth Latin American Workshop on Non-Monotonic Reasoning 2008. CEUR-WS.org (2008)
21. Patel-Schneider, P., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax. In: W3C Recommendation (2004)
22. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language and Information **14**(3), 369–395 (2005)
23. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-Efficient OWL Reasoner. In: A. Ruttenberg, U. Sattler, C. Dolbear (eds.) Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU). Karlsruhe, Germany (2008)
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a pratical owl-dl reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)
25. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. Journal of Artificial Intelligence Research **12**, 199–217 (2000)
26. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. In: Ph.D thesis, RWTH Aachen, Germany (2001)
27. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), *Lecture Notes in Artificial Intelligence*, vol. 4130, pp. 292–297. Springer (2006). URL download/2006/TsHo06a.pdf
28. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Proceedings of ICALP 2002, *LNCS*, vol. 1443, pp. 628–641. Springer (1998)
29. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. In: Journal of Computer and System Science, 32, p. 183221. Springer (1986)