A Worst-case Optimal Algorithm for SHOIQ

Chan Le Duc, Myriam Lamolle, Olivier Curé

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

Chan Le Duc LEDUC@IUT.UNIV-PARIS8.FR
Myriam Lamolle LAMOLLE@IUT.UNIV-PARIS8.FR
LIASD Université Paris 8 - IUT de Montreuil

Olivier Curé OCURE@UNIV-MLV.FR
Université Marne La Vallée

Abstract

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

1. Introduction

The ontology language OWL-DL (Patel-Schneider, Hayes, & Horrocks, 2004) is widely used to formalize semantic resources on the Semantic Web. This language is mainly based on the description logic $\mathcal{SHOIQ}$ which is known to be decidable (Tobies, 2000). Recently, another expressive description logic, namely $\mathcal{SROIQ}$ (Horrocks, Kutz, & Sattler, 2006a), 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:

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

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 (Tobies, 2000) has shown that the consistency problem of a $\mathcal{SHOIQ}$ knowledge base is NEXPTIME-complete. Moreover, tableaux-based algorithms presented in (Horrocks & Sattler, 2007) and (Horrocks, Kutz, & Sattler, 2006b) for $\mathcal{SHOIQ}$ have been exploited to implement reasoners such as FaCT++ (Tsarkov & Horrocks, 2006), Pellet (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007), which inherit the success from early Description Logic reasoners such as FaCT (Horrocks, 1998) and RACER (Haarslev & Moller, 2001).

In addition, there are several results which have provided worst-case optimal algorithms for Description Logics (DL) without nominal. The work in (Ortiz, 2008) has introduced the so-called $\mathcal{ALCQIb}^{+}_{reg}$ logic (capturing $\mathcal{SRIQ}$ (Horrocks, Kutz, & Sattler, 2005)) 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 ((Vardi & Wolper,), (Vardi, 1998)), this work has also proposed a worst-case optimal algorithm for the satisfiability problem in $\mathcal{ALCQIb}^{+}_{reg}$. Another approach, namely *resolution-based*, has been used in (Hustadt, Motik, & Sattler, 2005) 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 $\mathcal{SHOIQ}$ (Tobies, 2000). The work in (Kazakov & Motik, 2008) 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 ensured, and thus, is responsible for this hardness. Recently, (Motik, Shearer, & Horrocks, 2009) has introduced a combination of the resolution-based approach with the tableaux-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 (Shearer, Motik, & Horrocks, 2008).

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 (Horrocks & Sattler, 2007) and (Motik et al., 2009) 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 (De Giacomo & Lenzerini., 1994) 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 s must satisfy a term such as $(\leq 1\ a)$ and s has two a-neighbors t_1 and t_2 then t_1, t_2 have to satisfy the same set of semantic constraints, i.e. the label of t_1 is identical to that of t_2. It leads us to develop a technique that replaces merging t_1 into t_2 with (i) *propagating* the label from one to another and (ii) *memorizing* that t_1 and t_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 *partition function* for memorizing nodes that could be merged. As a consequence, it allows us to maintain tree-like structures for representing models without adding nominal nodes with new nominals, and thus, the blocking technique can be reused over these tree-like structures to achieve termination.

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 tableaux-based algorithms and their issues. This section gives also main arguments to show how these issues are not present in our approach introduced in this paper. In Section 4, we provide a definition of tableaux for $\mathcal{SHOIQ}$. Section 5 describes in detail an algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base. At the end of this section, we show that this algorithm is a worst-case optimal decision procedure for $\mathcal{SHOIQ}$. Finally, we discuss the results and future work.

2. The Description Logic $\mathcal{SHOIQ}$

In this section, we present the syntax, 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* $\mathbf{R}_\mathsf{I} = \{P^- \mid P \in \mathbf{R}\}$ *to denote a set of inverse roles. Each element of* $\mathbf{R} \cup \mathbf{R}_\mathsf{I}$ *is called a* $\mathcal{SHOIQ}$*-role. We define a function* $R^\ominus$ *which returns* R^- *if* $R \in \mathbf{R}$*, and returns* R *if* $R \in \mathbf{R}_\mathsf{I}$*.*

$*$ *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* $\sqsubseteq\!\!\!\!\!\!*$ *is defined as the transitive-reflexive closure of* $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\}$*. 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* $Q \sqsubseteq\!\!\!\!\!\!* R$*. A role* R *is called* simple *w.r.t.* $\mathcal{R}$ *if* $\mathsf{Trans}(Q) = \mathsf{false}$ *for all* $Q \sqsubseteq\!\!\!\!\!\!* R$*.*

$*$ *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 $\mathcal{SHIQ}$ (Horrocks, Sattler, & Tobies, 1999b). 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 subset* $\mathbf{C}_o \subseteq \mathbf{C}$ *of nominals.*

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

$*$ *An interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *consists of a non-empty set* $\Delta^\mathcal{I}$ *(*domain*) and a function* $\cdot^\mathcal{I}$ *which maps each concept name to a subset of* $\Delta^\mathcal{I}$ *such that*
$\top^\mathcal{I} = \Delta^\mathcal{I}$*,* $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$*,* $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$*,* $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$*,* $\mathsf{card}\{o^\mathcal{I}\} = 1$ *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 (Horrocks et al., 1999b). According to (Baader & Nutt, 2007), $\mathsf{nnf}(C)$ 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:

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

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 (Fischer & Ladner, 1979). The closure $\mathsf{cl}(\mathcal{T}, \mathcal{R})$ contains not only a set $\mathsf{sub}(\mathcal{T})$ of all sub-concepts syntactically obtained from $\mathcal{T}$ but also sub-concepts that are semantically derived from $\mathsf{sub}(\mathcal{T})$ w.r.t. $\mathcal{R}$. For instance, (i) if $\forall S.C \in \mathsf{sub}(\mathcal{T})$ and $R \sqsubseteq\!\!\!\!\sqsubseteq S \in \mathcal{R}$ then $\forall R.C \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$, or (ii) if $(\leq nS.C) \in \mathsf{sub}(\mathcal{T})$ with $n \geq 1$ then $(\leq mS.C), (\geq mS.C) \in \mathsf{cl}(\mathcal{T}, \mathcal{R})$ 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 tableaux-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 tableaux-based algorithms for DLs with inverse roles, number restrictions and nominals.

3.1 Related Work

Donini and Massacci in (Francesco M. Donini, 2000) have provided a tableaux-based algorithm for the logic $\mathcal{ALC}$ that runs in single exponential time. The authors have proposed a 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* ($\sqcap$,$\exists$, $\forall$) (Donini, 2007).

Goré and Nguyen in (Goré & Nguyen, 2007) 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 (Motik et al., 2009) 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 introduced in (Horrocks & Sattler, 2007) a tableaux-based algorithm for $\mathcal{SHOIQ}$ 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, tableaux-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, tableaux-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 tableaux-based algorithm presented in (Horrocks & Sattler, 2007), from which we have learned the following characteristics of completion graphs for $\mathcal{SHOIQ}$:

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.

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 tableaux-based algorithm. To ensure termination of such an algorithm, (i) we reuse the usual blocking condition for the tree-like part, (ii) we have to control the interaction between these two parts such that if there is a merge between a non-nominal and nominal node the tree-like structure should be maintained in order to take advantage of the blocking condition and 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 "unravelling" 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 the termination, (Horrocks & Sattler, 2007) introduced some new rules which can add nominal nodes with new nominals and push "embarrassing" nodes with their labels into the non-tree part such that all merges related to such nodes are no longer required. It was shown that this process may add a double exponential number of new nominal nodes. This explains why the tableaux-based algorithm presented in (Horrocks & Sattler, 2007) is not worst-case optimal.

3.2 Overview of the novel tableaux-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 $\mathcal{SHOIQ}$. This behaviour is rather originated from a selected structure on which we know how to establish a mechanism to ensure termination. This structure requires to push "embarrassing" nodes into the non-tree part by generating new nominal neighbors, then merging one of them into a non-nominal neighbor. This implies that we could avoid adding new nominal nodes if the tree-like structure can be maintained by some way such that it allows us to avoid merging *explicitly* nominal nodes.

Although experiments indicate that the tableaux and hypertableaux-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 tableaux-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 to 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 deal with 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 to devise correctly a model by unravelling over tree-like structures, we use a new rule, called NN_{φ}-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.

All properties of Definition 4 except for P15 are taken from (Horrocks et al., 1999b) and (Horrocks & Sattler, 2007). 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 (Le Duc, 2009) 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 makes an important impact on the way by which a tableaux-based algorithm can build completion graphs. For instance, each application of the expansion rules presented in (Horrocks et al., 1999b) for $\mathcal{SHIQ}$ performs some changes to its neighbor 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 (Horrocks & Sattler, 2007) for $\mathcal{SHOIQ}$ may change two arbitrary nodes while the algorithm introduced in (Le Duc, 2009) 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 (Tobies, 2001) have indicated that the concept satisfiability problems in $\mathcal{ALC}$ (with *CGI*) 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.

Regarding the definition of tableaux for $\mathcal{SHOIQ}$ presented in (Horrocks & Sattler, 2007), we add a new property, namely P15. This new property imposes an exact number of S-neighbor individuals t of s if $(\leq nS.C) \in \mathcal{L}(s)$. This property makes explicit the nondeterminism implied from the semantics of $(\leq nS.C)$ and requires an extra expansion rule, namely *NN*$_{\leq}$-rule, introduced in Figure 3. The presence of this rule may have an impact on the so-called "pay-as-you-go" behaviour of the tableaux-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 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:*

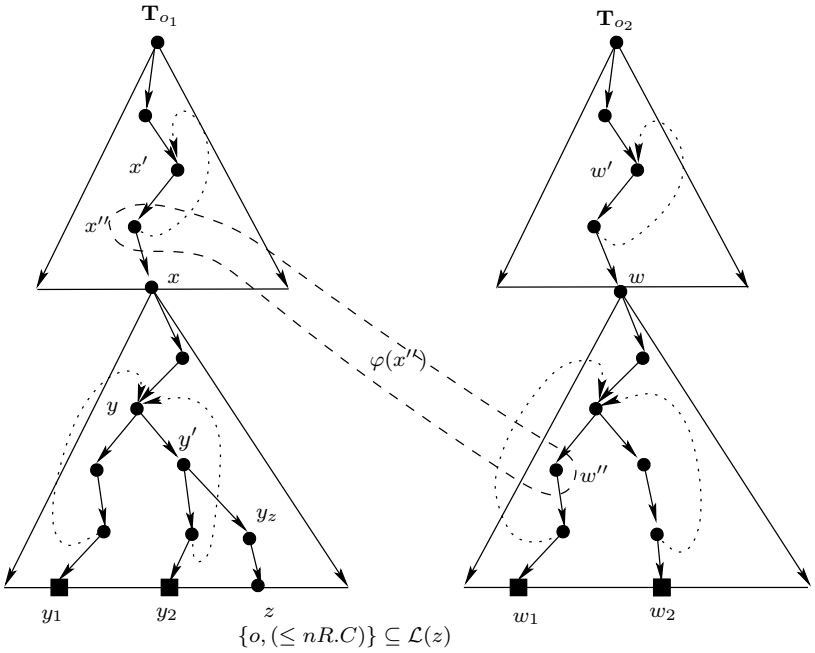

Figure 1: $\mathcal{SHOIQ}$-trees with a partitioning function φ

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* $\mathsf{card}\{S^T(s, C)\} \geq n$,
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* $\doteq C \in \mathcal{L}(t)$ *where* $S^T(s, C) := \{t \in \mathbf{S} | \langle s, t\rangle \in \mathcal{E}(S) \wedge C \in \mathcal{L}(t)\}$,
P13 *If* $\mathrm{o} \in \mathcal{L}(s) \cap \mathcal{L}(t)$ *for some* $\mathrm{o} \in \mathbf{C}_o$ *then* $s = t$,
P14 *For each* $\mathrm{o} \in \mathbf{C}_o$, *if* o *occurs in* $\mathcal{T}$ *then there is* $s \in \mathbf{S}$ *such that* $\mathrm{o} \in \mathcal{L}(s)$.
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* $\{(\leq mS.C), (\geq mS.C)\} \subseteq \mathcal{L}(s)$.

Lemma 5 *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}$ *knowledge base.* $(\mathcal{T}, \mathcal{R})$ *is consistent iff there is a tableau for* $(\mathcal{T}, \mathcal{R})$.

A proof of a similar lemma for $\mathcal{SHIQ}$ tableaux can be found in (Horrocks, Sattler, & Tobies, 1999a). From this proof, it is not hard to prove Lemma 5 for $\mathcal{SHOIQ}$ tableaux with the new properties P13, P14 and P15.

5. A novel decision procedure for $\mathcal{SHOIQ}$

In this section, we introduce $\mathcal{SHOIQ}$-trees with Definition 6. Intuitively, a $\mathcal{SHOIQ}$-tree is defined by considering nominals as non-nominal concepts. The important difference of $\mathcal{SHOIQ}$-trees from completion trees for $\mathcal{SHIQ}$ is the presence of a *blocking condition with two layers* that is governed by *1-iterated* and *2-iterated* nodes. A 1-iterated node in a $\mathcal{SHOIQ}$-tree whose root is a nominal node corresponds exactly to a blocked node in a completion tree for $\mathcal{SHIQ}$. A 2-iterated node y in a $\mathcal{SHOIQ}$-tree is a descendant of a 1-iterated node x such that the blocking condition for y is applied from x, i.e., x is considered as root of the subtree on which the condition blocking is applied for each 2-iterated descendant y of x. The reason for introducing the blocking condition with two layers is to facilitate the definitions of inner and blocked nodes which will be presented in Definition 7.

Definition 6 ($\mathcal{SHOIQ}$-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 $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \widehat{x}_o, \neq)$, 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 $\mathcal{L}$ such that $\mathcal{L}(x) \subseteq \mathsf{cl}(\mathcal{T},\mathcal{R})$ and $o \in \mathcal{L}(\widehat{x}_o)$. A node $x \in V_o$ is called* nominal *if $o' \in \mathcal{L}(x)$ for some $o' \in \mathbf{C}_o$. In addition, $\neq$ is a symmetric binary relation over V_o.*

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

$*$ *If $\langle x,y\rangle \in E_o$ then y is called a successor of x, denoted by $y \in \mathsf{succ}^1(x)$, or x is called the predecessor of y, denoted by $x = \mathsf{pred}^1(y)$. In this case, we say that x is a neighbor of y or y is a neighbor 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 $n > 0$.*

$*$ *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 $R' \in \mathcal{L}(\langle x,y\rangle)$ (resp. $R' \in \mathcal{L}(\langle y,x\rangle)$) and $R' \sqsubseteq^* R$. A node y is called a R-neighbor of x if y is either a R-successor or R-predecessor of x. If z is 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 and a role S, we define the set $S^{\mathbf{T}}(x,C)$ of x's S-neighbors as follows:*

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

$*$ *A node x is called* iterated *by y w.r.t. a node v, denoted by $y = \mathsf{b}(x)$, 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. *$v = \mathsf{pred}^n(y)$, $y = \mathsf{pred}^m(x)$,*
2. *$x' = \mathsf{pred}^1(x)$, $y' = \mathsf{pred}^1(y)$,*
3. *$\mathcal{L}(x) = \mathcal{L}(y)$, $\mathcal{L}(x') = \mathcal{L}(y')$,*
4. *$\mathcal{L}(\langle x',x\rangle) = \mathcal{L}(\langle y',y\rangle)$, and*

5. *if there are* z, z' *such that* $z' = \mathsf{pred}^1(z)$, $\mathsf{pred}^i(z') = v$, $\mathcal{L}(z) = \mathcal{L}(y)$, $\mathcal{L}(z') = \mathcal{L}(y')$ *and* $\mathcal{L}(\langle z', z\rangle) = \mathcal{L}(\langle y', y\rangle)$ *then* $n \leq i$.

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

A $\mathcal{SHOIQ}$-tree according to Definition 6 is partitioned into two layers : the first layer is formed of nodes from the root to each 1-iterated node, and the second layer consists of nodes from each 1-iterated node to each its 2-iterated descendant. This is depicted in Figure 1. In this figure, for instance, 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_1 is 2-iterated by y.

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

Definition 7 (set of $\mathcal{SHOIQ}$-trees with a partitioning function) *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}$ *knowledge base. Let* $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ *be a set of* $\mathcal{SHOIQ}$*-trees for* $(\mathcal{T}, \mathcal{R})$ *where* $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq)$.

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

1. *For* $x, x' \in \bigcup_{o\in\mathbf{C}_o} V_o$*, if* $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ *for some* $o \in \mathbf{C}_o$ *and* $x{\neq}x'$ *does not hold then* $\varphi(x) = \varphi(x')$*;*

2. *For* $x, x' \in \bigcup_{o\in\mathbf{C}_o} V_o$*, if* $x' \in \varphi(x)$ *then* $\varphi(x) = \varphi(x')$*, and if* $\varphi(x) = \varphi(x')$ *then* $\mathcal{L}(x) = \mathcal{L}(x')$*. Therefore, we define* $\mathcal{L}(\alpha) = \mathcal{L}(z)$ *for* $\alpha \in 2^{(\bigcup_{o\in\mathbf{C}_o} V_o)}$ *and* $z \in \alpha$*. In addition, for* $x, x' \in \bigcup_{o\in\mathbf{C}_o} V_o$ *we define* $\varphi(x){\neq}\varphi(x')$ *if there are two nodes* y, y' *such that* $\varphi(x) = \varphi(y)$*,* $\varphi(x') = \varphi(y')$ *and* $y{\neq}y'$*;*

3. *For* $x, x', y, y' \in \bigcup_{o\in\mathbf{C}_o} V_o$ *such that* $\langle x, x'\rangle \in E_g$ *(or* $\langle x', x\rangle \in E_g$*) and* $\langle y, y'\rangle \in E_h$ *(or* $\langle y', y\rangle \in E_h$*) with* $g, h \in \mathbf{C}_o$*, if* $\varphi(x) = \varphi(y)$*,* $\varphi(x') = \varphi(y')$ *and* $\mathcal{L}'(\langle x, x'\rangle) \neq \emptyset, \mathcal{L}'(\langle y, y'\rangle) \neq \emptyset$ *then* $\mathcal{L}'(\langle x, x'\rangle) = \mathcal{L}'(\langle y, y'\rangle)$ *where* $\mathcal{L}'(\langle v, v'\rangle) = \mathcal{L}(\langle v, v'\rangle)$ *if* $\langle v, v'\rangle \in E_{o'}$ *and* $\mathcal{L}'(\langle v, v'\rangle) = \{S^{\ominus} \mid S \in \mathcal{L}(\langle v', v\rangle)\}$ *if* $\langle v', v\rangle \in E_{o'}$ *with* $o' \in \mathbf{C}_o$*. Therefore, we define* $\mathcal{L}(\langle\varphi(x), \varphi(x')\rangle) = \mathcal{L}'(\langle z, z'\rangle)$ *with* $z \in \varphi(x)$*,* $z' \in \varphi(x')$*,* $\mathcal{L}'(\langle z, z'\rangle) \neq \emptyset$*, and* $\langle z, z'\rangle \in E_{o'}$ *or* $\langle z', z\rangle \in E_{o'}$*;*

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

   - $C \in \mathcal{L}(\varphi(x_i))$*,* $R \in \mathcal{L}(\langle\varphi(x), \varphi(x_i)\rangle)$ *for* $i \in \{0, \cdots, n\}$*, and*

- *there are two nodes x_l, x_m with $l, m \in \{0, \cdots, n\}$ such that $\varphi(x_l) \neq \varphi(x_m)$ does not hold.*

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

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

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

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

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

1. *There is some node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $\{A, \dot{\neg} A\} \subseteq \mathcal{L}(\varphi(x))$ for some concept name $A \in \mathbf{C}$;*
2. *There are nodes $x, y \in \bigcup_{o \in \mathbf{C}_o} V_o$ such that $x \neq y$ and $o \in \mathcal{L}(\varphi(x)) \cap \mathcal{L}(\varphi(y))$ for some $o \in \mathbf{C}_o$;*
3. *There is a node $x \in \bigcup_{o \in \mathbf{C}_o} V_o$ with $(\leq nR.C) \in \mathcal{L}(\varphi(x))$ and there are $(n+1)$ nodes $x_0, \cdots, x_n \in \bigcup_{o \in \mathbf{C}_o} V_o$ 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 $i \in \{0, \cdots, n\}$.* $\lhd$

Algorithm 1 constructs a set $\mathbf{T}$ of $\mathcal{SHOIQ}$-trees with a partitioning function. It terminates when no rule in Figure 2 and 3 is applicable. In this case, the obtained $\mathbf{T}$ is called *complete*, and if $\mathbf{T}$ contains no clash then $\mathbf{T}$ is called *clash-free*.

Definition 7 exhibits two key notions. The first one is *inner nodes*. An inner node must belong to the second layer of a $\mathcal{SHOIQ}$-tree. In Figure 1, when a node w'' of the second layer is inner we no longer need to take care of satisfying concepts contained in its label since this is done with a node x'' of the first layer such that both w'', x'' belong to the same partition $\varphi(x'')$.

The second notion is *blocking condition* with blocking and blocked nodes located in the second layer of a $\mathcal{SHOIQ}$-tree. The characteristics of $\mathcal{SHOIQ}$-trees with a partition function, described by Lemma 8, allow us to avoid duplicating nominal nodes when constructing a model from these trees. The definition of blocked nodes says that if a 2-iterated node y_1 is blocked by an ancestor y then it is necessary that any nominal descendant z of y does not have a "hard path" from y to z, i.e. the label of each node y_i between z and y including z contains a term such as $(\leq n_i R_i.C_i)$ such that the predecessor y_j of y_i is a R_i-neighbor of y_i and the label of y_j contains C_i. This implies that we can avoid infinitely duplicating the nominal node z when constructing a model. This is also depicted in Figure 1.

The $\mathcal{SHIQ}$ expansion rules described in Figure 2 are similar to those presented in (Horrocks et al., 1999b) except that they may propagate a label update between nodes of a partition or between

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

Figure 2: Expansion rules for $\mathcal{SHIQ}$

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

The $\mathcal{SHOIQ}$ expansion rules described in Figure 3 (except for *NN*$_\varphi$-rule) perform on the partition function φ necessary updates triggered by merging partitions. We adopt a strategy of rule applications as follows:

The NN$_\varphi$-rule and all $\mathcal{SHIQ}$-rules are applied with higher priority.

We can realize that there are some relationships between the expansion rules introduced in (Horrocks & Sattler, 2007) for dealing with nominals and the rules in Figure 3. The *NN*-rule in (Horrocks & Sattler, 2007) was employed to avoid the so-called "yo-yo" effect ((Baader & P., 1991)). This effect as illustrated by an example in (Baader & Sattler, 2001) may lead to an infinite sequence of "merging-and-generating". This is not exhibited in the algorithm presented in this paper since we do not merge explicitly nominal nodes. Whereas, the *NN*$_\varphi$-rule in Figure 3 is aimed at trying to make "hard" the path from each nominal node z to an ancestor y of z if all nodes of the path contain a term such as $(\leq nS.C)$. Unlike *NN*-rule that is applied only to nominal nodes, *NN*$_\varphi$-rule may add terms $(\leq mS.C)$ and $(\geq mS.C)$ to any node such that its label includes a term $(\leq nS.C)$ with $m \leq n$.

The o-rule in (Horrocks & Sattler, 2007) is very similar to the o_φ-rule in Figure 3. When the o_φ-rule is applied to two nodes x, y it redefines the partitioning function and propagates node and edge labels between x, y as if they are merged. Finally, while the $\leq_o$-rule in (Horrocks & Sattler, 2007) merges nominal neighbors of a nominal nodes, the $\leq_\varphi$-rule in Figure 3 performs necessary label propagations and modifies the partitioning function such that the number of neighbor partitions of a partition $\varphi(x)$ satisfies a term such as $(\leq nR.C)$ if this term is included in the label of x. Notice that this rule may group $\varphi(y)$ into $\varphi(x)$ by being applied to x with $(\leq nR.C) \in \mathcal{L}(x)$.

Notice that the relation $\neq$ over the set of nodes of all $\mathcal{SHOIQ}$-trees is defined by $\mathcal{SHIQ}$-rules. This relation is not changed by the $\mathcal{SHOIQ}$-rules. Moreover, $x{\neq}y$ implies $\varphi(x){\neq}\varphi(y)$ where x, y are nodes of $\mathcal{SHOIQ}$-trees.

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

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

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

Figure 3: New expansion rules for $\mathcal{SHOIQ}$

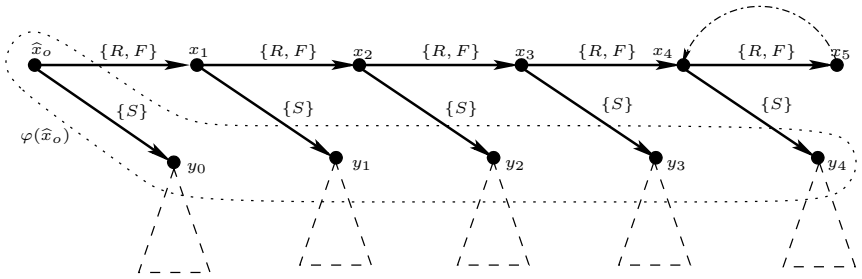

Figure 4: A $\mathcal{SHOIQ}$-tree with a partitioning function φ for the example

5.1 Example application of the algorithm

We start by considering a knowledge base with concept and role axioms as follows.

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

According to Algorithm 1, 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 4 provides 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 :

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

In turn, $\exists$-rule, $\forall$-rule, $\forall_+$ 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
$\mathcal{L}(y_0) = \{o, \exists S.o, (\neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C)\}$, $\mathcal{L}(x_1) = \{C, \exists F.C, \forall R.\exists F.C, \exists S.o, \neg o\}$.

Then, $\sqcap$-rule is applied to y_0 and x_1, we have
$\mathcal{L}(y_0) = \{o, \exists S.o, (\neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C), \neg C, \exists F.C, \forall R.\exists F.C\}$,
$\mathcal{L}(x_1) = \{C, \exists F.C, \forall R.\exists F.C, \exists S.o, \neg o\}$.

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 $i \in \{1, \cdots, 4\}$ in order to obtain $\varphi(\widehat{x}_o) = \varphi(y_0) = \cdots = \varphi(y_4) = \{\widehat{x}_o, y_1, \cdots, y_4\}$. No label propagation is needed. According to the o_φ-rule we have an "edge" connecting $\varphi(\widehat{x}_o)$ to itself since there is an edge $\langle \widehat{x}_o, y_0 \rangle$ with $\widehat{x}_o, y_0 \in \varphi(\widehat{x}_o)$. It is not useful to apply expansion rules to y_i since they are grouped into $\widehat{x}_o$ by φ.

From the definition of 1-iterated and 2-iterated nodes, we can see that x_3 is 1-iterated by x_2 and x_5 is 4-iterated by x_4. In addition, according to the definition of inner and blocked nodes, y_4 is inner, and x_5 is blocked by x_4 since y_4 is a nominal descendant of the blocking node x_4 and the label of y_4 contains no term such as $(\leq n\ S^{\ominus}.E)$ with $E \in \mathcal{L}(x_4)$. It is obvious that the obtained $\mathbf{T}$ is complete and clash-free. Therefore, $(\mathcal{R}_1, \mathcal{T}_1)$ is consistent.

We now modify slightly the knowledge base as follows.

$$\begin{array}{lcl} \mathcal{R}_1 & = & \{F \sqsubseteq R, \mathsf{Trans}(R)\}, \\ \mathcal{T}_2 & = & \{\top \sqsubseteq \exists S.(o \sqcap (\leq 1 S^{-}.C)), \\ & & \ o \sqsubseteq \neg C \sqcap \exists F.C \sqcap \forall R.\exists F.C\} \end{array}$$

When the at-most number restriction $(\leq 1 S^{-}.C)$ occurs in the axiom, NN_φ-rule is applicable to y_i since $(\leq 1 S^{-}.C) \in \mathcal{L}(y_i)$ for $i \in \{1, \cdots, 4\}$. This rule guesses a unique $m = 1$ and adds $(\leq 1 S^{-}.C), (\geq 1 S^{-}.C)$ to $\mathcal{L}(y_i)$. In turn, the application of o_φ-rule and $\leq_\varphi$ must group all nodes to a unique partition $\varphi(\widehat{x}_o)$. This implies that $\{C, \neg C\} \subseteq \mathcal{L}(\varphi(\widehat{x}_o))$. Thus, $(\mathcal{R}_1, \mathcal{T}_2)$ is not consistent. $\lhd$

Lemma 8 presents the most important characteristics of a clash-free set of $\mathcal{SHOIQ}$-trees. They allow us to use a partitioning function φ to govern all fusions of nodes and edges triggered by

merging nominal nodes instead of performing explicitly these fusions on $\mathcal{SHOIQ}$-trees. One of these characteristics says that if a node x does not have any inner ancestor then x is blocked. This simplifies dramatically termination condition for the tableaux-based algorithm.

Lemma 8 *Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Let $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C_o}\}$ be a set of $\mathcal{SHOIQ}$-trees with a function φ which are built by Algorithm 1 where $\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \widehat{x}_o, \neq)$. If $\mathbf{T}$ is clash-free then*

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

Before proving the lemma, let us point out why we need the properties of $\mathcal{SHOIQ}$-trees formulated in Lemma 8. Property 1 affirms that a partitioning function φ can be maintained by the applications of the expansion rules. Property 2 says that grouping any two distinct non-nominal nodes into a partition results from grouping two nominal nodes. This result is necessary to show that if $x \in \varphi(y)$ and y is blocked then x must be a 2-iterated and $\mathcal{L}(x') = \mathcal{L}(y')$, $\mathcal{L}(\langle x', x\rangle) = \mathcal{L}(\langle y', y\rangle)$ where x', y' are the predecessors of x, y respectively. This allows us to use a unique blocking node for all nodes $w \in \varphi(y)$. We refer the readers to the proof of Lemma 10 for more details. Property 3 affirms that if a node y located in the second layer of a $\mathcal{SHOIQ}$-tree and y has a "hard path" leading to a nominal descendant then y is inner. This property is obtained thanks to *NN*$_\varphi$-rule, $\geq$-rule, $\leq$-rule and the characteristics of φ. They impose that if $x \in \varphi(y)$ and $(\leq nS.C) \in \mathcal{L}(x) = \mathcal{L}(y)$ then *NN*$_\varphi$-rule guesses some $m \leq n$ and adds $(\leq mS.C)$ to $\mathcal{L}(x)$ and $\mathcal{L}(y)$. In turn, $\geq$-rule and $\leq$-rule generate exactly m S-neighbors x_i of x containing C, and m S-neighbors y_j of y containing C as well. Properties 3 and 4 ensure that each node located in the second layer of a $\mathcal{SHOIQ}$-tree either belongs a partition with a node located in the first layer, or is blocked. This is crucial for soundness of the tableaux-based algorithm.

Proof:[Proof of Lemma 8]

1. o_φ-rule ensures that if $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and $x{\neq}x'$ does not hold then $\varphi(x) = \varphi(x')$ i.e. the condition 1 in Definition 7 holds. Since $\exists$-rule and $\geq$-rule extend φ by adding new nodes or edges, it is obvious that the conditions 1 and 2 in Definition 7 hold. In addition, applications of $\sqsubseteq$-rule, $\sqcap$-rule, $\sqcup$-rule, $\forall$-rule, $\forall_+$-rule, ch-rule, *NN*$_\varphi$-rule to a node x change the label of nodes and propagate each added concept to the label of all nodes of a partition $\varphi(x)$. $\leq$-rule changes the label of nodes and edges. In particular, it sets the label of an edge to empty and does not propagate the emptiness to the other edges that connect the two related partitions. Therefore, the conditions 2 and 3 in Definition 7 hold.

   Applications of o_φ-rule and $\leq_\varphi$-rule to a node x change the label of nodes and propagate each added concept to the label of all nodes of a partition $\varphi(x)$. In addition, they propagate also

each role from an edge $\langle x, y\rangle$ to the label of all edges that connects a node $x' \in \varphi(x)$ to a node $y' \in \varphi(y)$. This implies that the conditions 2 and 3 in Definition 7 hold.

Finally, $\leq_\varphi$-rule ensures that the condition 4 in Definition 7 holds.

2. Since $x \neq y$ are non-nominal and $x \in \varphi(y)$, x, y must be grouped by applying $\leq_\varphi$-rule to a node z such that $x_1 \neq y_1 \in \varphi(z)$, $\mathcal{L}'(\langle x, x_1\rangle) = \mathcal{L}'(\langle y, y_1\rangle)$, $(\leq n_0 R_0.C_0) \in \mathcal{L}(x)$, $C_0 \in \mathcal{L}(x_1)$. According to Lemma 9, each $\mathbf{T}_o$ is built by a finite number of applications of expansion rules and a first merge is started by the application of o_φ-rule. By induction, it follows that there are nodes $x_0, \cdots, x_k = x$ and $y_0, \cdots, x_k = y$ with $x_i \neq y_i$ for $0 \leq i \leq k$ such that $\varphi(x_i) = \varphi(y_i)$, $\mathcal{L}'(\langle x_j, x_{j+1}\rangle) = \mathcal{L}'(\langle y_j, y_{j+1}\rangle)$, $(\leq n_j R_j.C_j) \in \mathcal{L}(x_j)$, $C_j \in \mathcal{L}(x_{j+1})$,

   $\mathcal{L}(\langle x_j, x_{j+1}\rangle)$ for all $0 \leq i \leq k$, $0 \leq j < k$, and $o \in \mathcal{L}(x_0)$ for some $o \in \mathbf{C}_o$.

3. By construction, we have $\varphi(\widehat{x}_o) = \varphi(z)$ with $o \in \mathcal{L}(z)$, and thus z is inner. Assume that nodes $w_1 = z, \cdots, w_k = w$ between z and w (i.e. w_{i+1} is the predecessor of w_i) are inner with $\varphi(w_i) = \varphi(y_i)$, $\mathcal{L}'(\langle w_j, w_{j+1}\rangle) = \mathcal{L}'(\langle y_j, y_{j+1}\rangle)$ where y_1 is a neighbor of $\widehat{x}_o$, y_{j+1} is a neighbor of y_j for all $0 < i \leq k$, $0 < j < k$. By the definition of 1-iterated and 2-iterated nodes, y is a descendant of a 1-iterated node v and y_i is an ancestor of a 1-iterated node for all $1 \leq i \leq k$. This implies that the predecessor w' of w is a descendant of v. Since $(\leq nR.C) \in \mathcal{L}(w)$ and $\mathcal{L}(w) = \mathcal{L}(y_k)$, y_k has $m \leq n$ R-neighbors $v_1, \cdots, v_m$ such that $C \in \mathcal{L}(v_i)$ and w has m R-neighbors $u_1, \cdots, u_m$ such that $C \in \mathcal{L}(u_i)$ for $i \in \{1, \cdots, m\}$ due to NN_φ-rule and $\geq$-rule. This implies that $w' \in \{u_1, \cdots, u_m\}$ since $C \in \mathcal{L}(w')$, $R^\ominus \in \mathcal{L}(\langle w', w\rangle)$, the non-applicability of $\leq$-rule and clash-freeness. Due to the non-applicability of $\leq_\varphi$-rule there is some $l \in \{1, \cdots, m\}$ such that $\varphi(v_l) = \varphi(w')$. Since y_k is an ancestor of a 1-iterated, v_l must be either a 1-iterated node or an ancestor of a 1-iterated node. That means that w' is inner. Thus, y is inner.

4. Let x be a node that is 2-iterated by y. Assume that x is not blocked. According to the definition of blocked nodes, there are the two following possibilities:

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

   (b) y has a descendant y' that is 2-iterated by an ancestor z' of y and z' has a nominal descendant z such that for each node w which is a descendant of z', and w is an ancestor of z' or $w = z'$, there is $(\leq nR.C) \in \mathcal{L}(w)$ with $C \in \mathcal{L}(w')$ and $R^\ominus \in \mathcal{L}(\langle w', w\rangle)$, $\langle w', w\rangle \in E_o$ for some $o \in \mathbf{C}_o$. By Item 3, we obtain z' is inner. □

□

Lemma 9 (Termination) *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Algorithm 1 terminates.*

Proof: First, we compute a upper bound of the $\mathcal{SHOIQ}$ tree's height from the condition for 1-iteration and 2-iteration. This upper bound equals $K = 2^{2m+k} \times 2$ where $m = \mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\}$, and k is the number of roles occurring in $\mathcal{T}$ and $\mathcal{R}$ plus their inverse. Moreover, the number of

neighbors of any node is bounded by $M = \sum m_i$ where m_i occurs in a number restriction term $(\geq m_i R.C)$ that appears in $\mathcal{T}$.

Applications of rules in Figure 2 and 3 do not remove concepts from the label of nodes. Moreover, applications of these rules do not remove roles from the label of edges except that they may set the label of edges to an empty set. In fact, when $\leq$-rule is applied to a node x to merge the label of a successor y of x to that of a node z, and the label of $\langle x, z\rangle$ to that of $\langle x, z\rangle$ such that relation $\neq$ is preserved i.e. this does not lead to apply $\geq$-rule to x more once. If y is a S-neighbor of x created by an application of $\geq$-rule then y is now replaced with z. When this rule makes the label of an edge become empty it remains to be empty forever. Notice that x cannot be created by applying $\geq$-rule to y since y is a successor of x.

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

Lemma 10 affirms that the success in constructing a clash-free set $\mathbf{T}$ of $\mathcal{SHOIQ}$-trees with a partitioning function φ for a knowledge base $(\mathcal{T}, \mathcal{R})$ guarantees the existence of a model of $(\mathcal{T}, \mathcal{R})$. In other terms, $\mathbf{T}$ and φ capture sufficiently information related to the consistency of $(\mathcal{T}, \mathcal{R})$. The proof of Lemma 10, which is presented thereafter, consists in building a model from $\mathbf{T}$ with a partitioning function φ. Figure 5 visualizes this construction with two $\mathcal{SHOIQ}$-trees $\mathbf{T}_{o_1}$ and $\mathbf{T}_{o_2}$.

The construction of a model is started at each nominal root. Such a model can be considered as a possibly infinite graph each node of which corresponds to either a partition or a duplication of x such that x is a blocking node or x is a descendant of a blocking node. The non-tree part of the model consists of nodes that are arbitrarily interconnected. The tree-like parts of the model consist of non-nominal nodes that would be obtained by "unravelling" the nodes of the tree-like parts.

But what is the border between these two parts of the model ? To answer to this question, we consider two partitions $\varphi(y_1)$ and $\varphi(y_z)$ in Figure 5 where y_1 is blocked by y, and y_z contains no term as such $(\leq mR.D)$ if $R^{\ominus} \in \mathcal{L}(\langle y', y_z\rangle)$ and $D \in \mathcal{L}(y')$. In this case, we can see that z, y_z are inner. In addition, as affirmed in the proof of Lemma 10, $\varphi(y_1)$ includes only 2-iterated nodes such as w with $\mathcal{L}(w) = \mathcal{L}(y_1)$ and $\mathcal{L}(\langle w', w\rangle) = \mathcal{L}(\langle y'_1, y_1\rangle)$ where w', y'_1 are the predecessors of w, y_1 respectively. This allows us to determine an infinite tree-like part of the model which goes through individuals corresponding to y and the descendants of y except for y_z, z, y_1 and y_2. This infinite tree-like part has an infinite number of individuals corresponding to duplications of y', and all of them have an S instance with only one individual $\varphi(y_z)$ of the non-tree part.

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

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

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

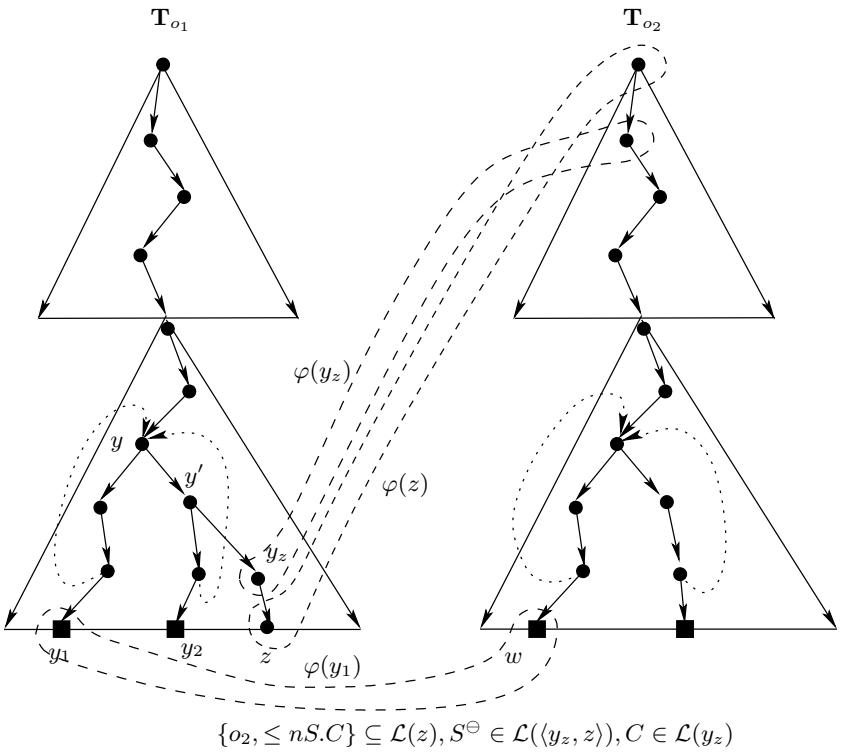

Figure 5: How a model can be built from a clash-free set of $\mathcal{SHOIQ}$-trees with a partition function φ.

For each blocking node y of a blocked node x, if y has a descendant nominal z then, according to the definition of blocked nodes and Lemma 8, y has a descendant y_z such that (i) y_z is not an ancestor of x, (ii) $y_z = z$ or y_z is an ancestor of z, and (iii) if $C \in \mathcal{L}(w')$ and $R^{\ominus} \in \mathcal{L}(\langle w', y_z\rangle)$ with $\langle w', y_z\rangle \in E_o$, $o \in \mathbf{C}_o$ then $(\leq nR.C) \notin \mathcal{L}(y_z)$, and (iv) y_z cannot be an ancestor of any blocked node (*).

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

In fact, according to Lemma 8, there are nodes $w_0, \cdots, w_l = w$ and $y_0, \cdots, y_l = y$ such that $\varphi(w_i) = \varphi(y_i)$, $\mathcal{L}'(\langle w_j, y_{j+1}\rangle) = \mathcal{L}'(\langle w_j, y_{j+1}\rangle)$, $(\leq n_j R_j.C_j) \in \mathcal{L}(w_j)$, $C_j \in \mathcal{L}(w_{j+1})$, $\mathcal{L}(\langle w_j, w_{j+1}\rangle)$ for all $0 \leq i \leq l$, $0 \leq j < l$, and $o \in \mathcal{L}(w_0) = \mathcal{L}(y_0)$ for some $o \in \mathbf{C}_o$. Since y is blocked by $\mathsf{b}(y)$, such a nominal node y_0 must be an ancestor of $\mathsf{b}(y)$ or a descendant of an ancestor of $\mathsf{b}(y)$. This implies that the path from w_0 to w_l have at least two edges whose labels are identical. By the construction of trees with the blocking condition, w_l must be either inner or 2-iterated. It is not possible that w_l is inner since $y \in \varphi(w_l)$ and y is blocked. This implies that w_l is 2-iterated and $w_{l-1} \in \varphi(y_{l-1})$ where $w_{l-1} = w'$ and $y_{l-1} = y'$ and $\mathcal{L}'(\langle w_{l-1}, w_l\rangle) = \mathcal{L}'(\langle y_{l-1}, y\rangle)$.

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

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

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

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

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 Figure 2 and the facts that $\mathbf{T}$ is clash-free.
- For P6, assume that $s, t \in \mathbf{S}$ with $\forall S.C \in \mathcal{L}'(s)$, $Q \sqsubseteq S$, $\mathsf{Trans}(Q)$ and $\langle s, t\rangle \in \mathcal{E}(Q)$. By the definition of T and and $\mathbf{T}$, we consider the following cases:
  - $s \in \mathbf{S}_1$ and $t \in \mathbf{S}_1$. This implies that there are x, y such that x, y have no inner ancestor, $s = \varphi(x), t = \varphi(y)$ and $\forall S.C \in \mathcal{L}(x)$, $Q \in \mathcal{L}'(\langle x, y\rangle)$. Due to the non-applicability of $\forall_+$-rule, it follows that $\forall Q.C \in \mathcal{L}(y)$. By the definition of T and $\mathbf{T}$, we have $\forall Q.C \in \mathcal{L}'(t)$.
  - $s \in \mathbf{S}_2$ and $t \in \mathbf{S}_2$. This implies that $\forall S.C \in \mathcal{L}(\mathsf{p}(s))$, $Q \in \mathcal{L}'(\langle \mathsf{p}(s), \mathsf{p}(t)\rangle)$. Due to the non-applicability of $\forall_+$-rule, it follows that $\forall Q.C \in \mathcal{L}(\mathsf{p}(t))$. By the definition of T, we have $\forall Q.C \in \mathcal{L}'(t)$.
  - $s \in \mathbf{S}_1$ and $t \in \mathbf{S}_2$. This implies that either (i) there are x, y such that (i) either $s = \varphi(x), \widehat{v}_o^{\varphi(y)} \in \widehat{V}_o^{\varphi(y)}$ with some $o \in \mathbf{C}_o$ and $\forall S.C \in \mathcal{L}(x)$, $Q \in \mathcal{L}'(\langle x, y\rangle)$, or (ii) $s = \varphi(x)$, $\mathsf{p}(t) \in \widehat{v}_o^{\varphi(y)}$ with some $o \in \mathbf{C}_o$ and $\forall S.C \in \mathcal{L}(x)$, $Q \in \mathcal{L}'(\langle x, \mathsf{p}(t)\rangle)$. Due to the non-applicability of $\forall_+$-rule, it follows that $\forall Q.C \in \mathcal{L}(y)$ or $\forall Q.C \in \mathcal{L}(\mathsf{p}(t))$. By the definition of T, we have $\forall Q.C \in \mathcal{L}'(t)$.
  - $s \in \mathbf{S}_2$ and $t \in \mathbf{S}_1$. Analogously.
- P5. Analogously.
- P7 holds due to the non-applicability of $\exists$-rule and the construction of T.
- For P10, assume that $s \in \mathbf{S}$ with $(\leq n'S.C) \in \mathcal{L}'(s)$. We consider the following cases:
  - $s \in \mathbf{S}_1$ i.e. $s = \varphi(x)$ for some $x \in V_o$ where x has no inner ancestor and $o \in \mathbf{C}_o$. Assume that x is not 2-iterated. Due to the non-applicability of NN_φ-rule, $\leq$-rule, $\geq$-rule and the clash-freeness, x has $n \leq n'$ nodes $x_1, \cdots, x_n$ such that $x_i \neq x_j$, x_i is S-neighbor of x with $C \in \mathcal{L}(x_i)$ for $1 \leq i < j \leq n$. Due to the definition of φ we have $\varphi(x_i) \neq \varphi(x_j)$ for $1 \leq i < j \leq n$. This implies that $\leq_\varphi$-rule is never applied to $\varphi(x_1), \cdots, \varphi(x_n)$. According to the definition of $\mathbf{S}_2$, it is not possible that there is some $v \in \widehat{V}_o^{\varphi(y)}$ such that $\mathsf{p}(v) = x_i$, $i \in \{1, \cdots, n\}$ since $(\leq nS.C) \in \mathcal{L}(x), C \in \mathcal{L}(\mathsf{p}(x_i)), S \in \mathcal{L}'(\langle \mathsf{p}(x_i), x\rangle)$.

    Assume that there is a x_i with $i \in \{1, \cdots, n\}$ and a blocked node y such that $x_i \in \varphi(y)$. According to (**), for any node $w' \in \varphi(x)$, if a node w is a neighbor of w' and $w \in \varphi(y)$ then w' is a 2-iterated node and $\mathcal{L}'(\langle w', w\rangle) = \mathcal{L}'(\langle y', y\rangle)$ where w', y' are the predecessor of w and y respectively. This implies that $\varphi(x_i)$ is replaced with $\widehat{v}_o^{\varphi(y)}$, and $\varphi(x)$ has exactly n distinct elements $s_1, \cdots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}'(s_i)$ and $S \in \mathcal{E}(\langle \varphi(x), s_i\rangle)$

    Assume that there is a x_i with $i \in \{1, \cdots, n\}$ such that $x_i = \mathsf{p}(v)$ with $v \in \widehat{V}_o^{\varphi(z)}$ for some blocked node z and $o \in \mathbf{C}_o$. By the definition of $\widehat{V}_o^{\varphi(z)}$, $(\leq nSC) \notin \mathcal{L}(x)$, which is a contradiction.

    Assume that x is 2-iterated. According to Lemma 8, x is blocked and $\varphi(x) \notin \mathbf{S}_1$.

– $s \in \mathbf{S}_2$ i.e. $s \in \widehat{V}_o^{\varphi(y)}$ for some blocked node y with $o \in \mathbf{C}_o$. P10 holds due to the definition of $\widehat{V}_o^{\varphi(y)}$ and the non-applicability of $\leq$-rule.

- To show P11, assume that $s \in \mathbf{S}_1$ i.e. $s = \varphi(x)$ for some $x \in V_o$ where x has no inner ancestor with $o \in \mathbf{C}_o$. Due to the non-applicability of *NN*$_\varphi$-rule, $\geq$-rule and the clash-freeness, x has n nodes $x_1, \cdots, x_n$ such that $x_i \neq x_j$, x_i is S-neighbor of x with $C \in \mathcal{L}(x_i)$ for $1 \leq i < j \leq n$. If there is a blocked node $y \in \varphi(x_i)$ then $\varphi(x_i)$ is replaced with $\widehat{v}_o^{\varphi(y)}$. Therefore, $\varphi(x)$ has at least n distinct elements $s_1, \cdots, s_n \in \mathbf{S}$ such that $C \in \mathcal{L}'(s_i)$ and $S \in \mathcal{E}(\langle \varphi(x), s_i \rangle)$.

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

- For P12 assume that $s, t \in \mathbf{S}$ with $(\leq nS.C) \in \mathcal{L}'(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$. By the definition of T, x_t is a S-neighbor of x_s and $(\leq nS.C) \in \mathcal{L}(x_s)$ where $x_s, x_t \in \varphi(x) \cup \varphi(y) \cup \{\mathsf{p}(s), \mathsf{p}(t)\}$, $\varphi(x), \varphi(y) \in \mathbf{S}_1$ and $\{\mathsf{p}(s), \mathsf{p}(t)\} \subseteq \widehat{V}_o^{\varphi(z)} \in \mathbf{S}_2$. Due to the non-applicability of ch-rule, we have $C \in \mathcal{L}(x_t)$ or $\doteq\neg C \in \mathcal{L}(x_t)$. Due to the definition of T, this implies $C \in \mathcal{L}'(t)$ or $\doteq\neg C \in \mathcal{L}'(t)$.

- 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 *NN*$_\varphi$-rule. □

□

Lemma 11 (Completeness) *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then Algorithm 1 can build a clash-free set of $\mathcal{SHOIQ}$-trees for $(\mathcal{T}, \mathcal{R})$.*

Proof: Let $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ be a tableau for $(\mathcal{T}, \mathcal{R})$. Let $\{\mathbf{T}_o = (V_o, E_o, \mathcal{L}, \widehat{x}_o, \neq) \mid o \in \mathbf{C}_o\}$ be a set of $\mathcal{SHOIQ}$-trees. We show that there exists a sequence of expansion rule applications such that it generates a 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-neighbor of x' for some $x' \in \varphi(x)$ and $y' \in \varphi(y)$ then $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S)$,

($\mathbf{C3}$) $x \neq y$ 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 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 preserved trivially, (*) holds if $\sqsubseteq$-rule is applied.

2. $\sqcap$-rule. Due to P3, if $(C \sqcap D) \in \mathcal{L}'(\pi(x))$ then $\{C, D\} \subseteq \mathcal{L}'(\pi(x))$. Due to (*) if $(C \sqcap D) \in \mathcal{L}(x')$ then $(C \sqcap D) \in \mathcal{L}'(\pi(x))$ for all $x' \in \varphi(x)$. Moreover, $\sqcap$-rule adds C, D to $\mathcal{L}(x')$ for all $x' \in \varphi(x)$ if $(C \sqcap D) \in \mathcal{L}(x')$ i.e. $(\mathbf{C1})$ 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, $\mathsf{card}\{S^T(\pi(x), C)\} \leq n$. For some $x' \in \varphi(x)$, $\leq$-rule is applicable if x' has $(n+1)$ S-neighbors $y_0, \cdots, y_n$ 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-neighbor of x then z is now a R-neighbor of x and $\langle \pi(x), \pi(z) \rangle \in \mathcal{E}(R)$ due to $\pi(y) = \pi(z)$ and (*). This implies 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, $\mathsf{card}\{S^T(\pi(x), C)\} \leq n$. $\leq_\varphi$-rule is applicable if there are $(n+1)$

nodes $y_0, \cdots, y_n$ with $C \in \mathcal{L}(y_i)$ and $\varphi(y_i) \cap \varphi(y_j)$ for all $0 \leq i < j \leq n$. This implies that there are two nodes $y, z \in \{y_0, \cdots, y_n\}$ such that $\pi(y) = \pi(z)$. Thus, $\varphi(y) \not\equiv \varphi(z)$ cannot hold due to (*). In addition, if there are $x', x'' \in \varphi(x)$ such that x' has a S-neighbor $y' \in \varphi(y)$ and x'' has a S-neighbor $z' \in \varphi(z)$ with $C \in \mathcal{L}(y') \cap \mathcal{L}(z')$ then $x' \neq x''$ since *NN*$_\varphi$-rule and all $\mathcal{SHIQ}$-rules are applied with higher priority (otherwise, $\varphi(y) \not\equiv \varphi(z)$ holds).

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-neighbor of w with $w \in \varphi(x') \cup \varphi(x'')$ and $w' \in \varphi(y') \cup \varphi(z')$ then $\langle \pi(w), \pi(w') \rangle \in \mathcal{E}(R)$ due to $\pi(y) = \pi(z)$ and (*). This implies $(\mathbf{C2})$ holds. Moreover, $(\mathbf{C3})$ is preserved since $\varphi(y) \not\equiv \varphi(z)$ 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 $x \not\equiv y$ 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 $x \not\equiv y$ 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 $\varphi(x_i) \cap \varphi(x_j) = \emptyset$, $\varphi(x_i) \not\equiv \varphi(x_j)$ 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 $x_i \not\equiv x_j$ 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 $x' \not\equiv x''$ does not hold due to $(\mathbf{C3})$, which contradicts $\varphi(x_i) \not\equiv \varphi(x_j)$. □

□

Theorem 12 *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. Algorithm 1 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it is worst-case optimal.*

Proof: From Lemma 9, 10 and 11, it follows that Algorithm 1 is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$. According to Tobies in (Tobies, 2000), the consistency problem of $\mathcal{SHOIQ}$ is NEXPTIME-complete. Algorithm 1 is worst-case optimal if we can show that it is NEXPTIME. In fact, Lemma 9 affirms that Algorithm 1 creates a set of $\mathcal{SHOIQ}$-trees whose size is exponential in the size of input $(\mathcal{T}, \mathcal{R})$. Given a set of $\mathcal{SHOIQ}$-trees with a partitioning function φ that defines an exponential number of partitions over the set of nodes, this algorithm can check whether (i) each expansion rule can be applied to each node of the trees and the partitions $\varphi(x)$ related to this node, and (ii) this set of trees is clash-free. From the pre-condition of the rules and the definition of clashes, it follows that the check process can run in polynomial time in the number of nodes and partitions. □ □

6. Conclusion and Discussion

We have presented in this paper a practical and worst-case optimal decision procedure for the logic $\mathcal{SHOIQ}$. We have based our approach on a new technique that constructs tree-like structures for representing a model without adding nominal nodes with new nominals. This technique is founded on the fact that fusions of nodes triggered by merging nominal nodes can be replaced with governing a partition function which would simulate this merging process. This allows us to reuse the blocking technique over these tree-like structures to obtain termination.

The expansion rules for the tableaux-based algorithm presented in this paper have managed to avoid breaking "materially" tree-like structures for representing models of a knowledge base. This behaviour of the algorithm can be straightforwardly ensured if the expansion rules manipulate only one node and its neighbors for each rule application, e.g., the expansion rules for $\mathcal{SHIQ}$. As discussed in Section 4, this behaviour is also related to the local property of tableaux. This makes us think that there is some relationship between the local property of tableaux and *the tree model property* that is formulated in (Vardi, 1997). Intuitively, we can see that the local property of tableaux could imply the tree model property since each expansion rule corresponding to a tableau property performs only changes to a node or its neighbors of a completion graph. We conjecture that these two properties are equivalent.

As discussed in Section 6, the rule NN_{φ} does not have a "pay-as-you-go" behaviour, i.e., it may add unnecessarily number restrictions to the label of a node. We think that this rule NN_{φ} should be improved in some way such that, for instance, it may be applied only to ancestors of a nominal node.

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

We would like to obtain by experiments real behaviours of our algorithm in practice. This may help us to know whether there is some benefit obtained from governing a partition function instead of adding new nominals. For this purpose, we plan to implement the algorithm presented in this paper and perform experiments with large ontologies. Since the structure of $\mathcal{SHOIQ}$-trees presented in this paper is similar to completion trees for $\mathcal{SHIQ}$, such an implementation can take advantage of the optimization techniques (Horrocks, 2007) available to tableaux-based algorithms for $\mathcal{SHIQ}$.

References

Baader, F., & Nutt, W. (2007). Basic description logics. In *The Description Logic Handbook: Theory, Implementation and Applications (2nd edition)*, pp. 47–104. Cambridge University Press.

Baader, F., & P., H. (1991). 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).

Baader, F., & Sattler, U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, *69*, 5–40.

De Giacomo, G., & Lenzerini., M. (1994). 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.

Donini, F. (2007). Complexity of reasoning. In *The Description Logic Handbook: Theory, Implementation and Applications (2nd edition)*, pp. 105–148. Cambridge University Press.

Fischer, M. J., & Ladner, R. I. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, *18*(18), 174–211.

Francesco M. Donini, F. M. (2000). Exptime tableaux for alc. *Artificial Intelligence*, *124*(1), 87–138.

Goré, R., & Nguyen, L. A. (2007). Exptime tableaux with global caching for description logics with transitive roles, inverse roles and role hierarchies. In *TABLEAUX*, pp. 133–148.

Haarslev, V., & Moller, R. (2001). Racer system description. In *Proc. of the Int. Joint on Automated Reasoning (IJCAR 2001)*, pp. 701–705. Springer.

Horrocks, I. (1998). The FaCT system. In de Swart, H. (Ed.), *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, Vol. 1397 of *Lecture Notes in Artificial Intelligence*, pp. 307–312. Springer.

Horrocks, I. (2007). Implementation and optimization techniques. In *The Description Logic Handbook: Theory, Implementation and Applications (2nd edition)*, pp. 329–373. Cambridge University Press.

Horrocks, I., Kutz, O., & Sattler, U. (2005). The irresistible $\mathcal{SRIQ}$. In *Proc. of the First Int. Workshop on OWL Experiences and Directions (OWLED 2005)*.

Horrocks, I., Kutz, O., & Sattler, U. (2006a). The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*. Springer-Verlag.

Horrocks, I., Kutz, O., & Sattler, U. (2006b). 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.

Horrocks, I., & Sattler, U. (2007). A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal Of Automated Reasoning*, *39*(3), 249–276.

Horrocks, I., Sattler, U., & Tobies, S. (1999a). A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. In *LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen*.

Horrocks, I., Sattler, U., & Tobies, S. (1999b). Practical reasoning for expressive description logics. In *Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999)*. Springer.

Hustadt, U., Motik, B., & Sattler, U. (2005). A Decomposition Rule for Decision Procedures by Resolution-based Calculi. In Baader, F., & Voronkov, A. (Eds.), *Proc. of the 11th Int. Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2004)*, Vol. 3452 of *LNAI*, pp. 21–35, Montevideo, Uruguay. Springer.

Kazakov, Y., & Motik, B. (2008). A Resolution-Based Decision Procedure for SHOIQ. *Journal of Automated Reasoning*, *40*(2–3), 89–116.

Le Duc, C. (2009). Decidability of $\mathcal{SHI}$ with transitive closure of roles. In *Proceedings of the European Semantic Web Conference*. Springer-Verlag.

Motik, B., Shearer, R., & Horrocks, I. (2009). Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research*, *36*, 165–228.

Ortiz, M. (2008). 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.

Patel-Schneider, P., Hayes, P., & Horrocks, I. (2004). Owl web ontology language semantics and abstract syntax. In *W3C Recommendation*.

Shearer, R., Motik, B., & Horrocks, I. (2008). HermiT: A Highly-Efficient OWL Reasoner. In Ruttenberg, A., Sattler, U., & Dolbear, C. (Eds.), *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, Karlsruhe, Germany.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: a pratical owl-dl reasoner. *Journal of Web Semantics*, *5*(2), 51–53.

Tobies, S. (2000). The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, *12*, 199–217.

Tobies, S. (2001). Complexity results and practical algorithms for logics in knowledge representation. In *Ph.D thesis, RWTH Aachen, Germany*.

Tsarkov, D., & Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297. Springer.

Vardi, M. Y. (1997). Why is modal logic so robustly decidable ?. *Discrete Mathematics and Theoretical Computer Science*, *31*(1), 149–184.

Vardi, M. Y. (1998). Reasoning about the past with two-way automata. In *Proceedings of ICALP 2002*, Vol. 1443 of *LNCS*, pp. 628–641. Springer.

Vardi, M. Y., & Wolper, P. Automata-theoretic techniques for modal logics of programs. In *Journal of Computer and System Science*, Vol. 1.