An ExpSpace Tableau-based Algorithm for SHOIQ

Chan Le Duc, Myriam Lamolle, Olivier Curé

An EXPSPACE Tableaux-Based Algorithm for $\mathcal{SHOIQ}$

Chan Le Duc[1], Myriam Lamolle[1], and Olivier Curé[2]

[1] LIASD Université Paris 8 - IUT de Montreuil, France
{chan.leduc, myriam.lamolle}@iut.univ-paris8.fr
[2] LIGM Université Paris-Est, France
ocure@univ-mlv.fr

Abstract. In this paper, we propose an EXPSPACE tableaux-based algorithm for $\mathcal{SHOIQ}$. The construction of this algorithm is founded on the standard tableaux-based method for $\mathcal{SHOIQ}$ and the technique used for designing a NEXPTIME algorithm for the two-variable fragment of first-order logic with counting quantifiers $\mathcal{C}^2$.

1 Introduction

The ontology language OWL-DL [1] 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 [2]. An interesting feature of logics with nominals (denoted by $\mathcal{O}$ in $\mathcal{SHOIQ}$) 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.

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 [2] has shown that the consistency problem of a $\mathcal{SHOIQ}$ knowledge base is NEXPTIME-complete. Moreover, tableaux-based algorithms presented in [3] for $\mathcal{SHOIQ}$ have been exploited to implement reasoners such as Pellet [4], which inherit from the success of early Description Logic reasoners such as FaCT [5].

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 $\mathcal{SHOIQ}$ [2]. The work in [6] has indicated that when nominals are allowed in $\mathcal{SHIQ}$, the resolution-based approach yields a triple exponential decision procedure for the consistency problem. The authors have also identified that the interaction between nominals, inverse roles and number restrictions makes termination more difficult to be achieved, and thus, is responsible for this hardness.

Our approach is inspired from a technique that was employed by Ian Pratt-Hartmann in [9] to construct a NEXPTIME algorithm for the logic $\mathcal{C}^2$ including $\mathcal{SHOIQ}$. Unlike the existing tableaux-based algorithms, this technique does not explicitly build a graph for representing a model but it builds a structure, called a *frame*, from *star-types* each of which represents a set of individuals. A result from [9] shows that a model of a $\mathcal{C}^2$ knowledge base can be constructed from a frame tiled by *well selected* star-types.

The present paper is structured as follows. In the next section, we describe the logic $\mathcal{SHOIQ}$ and the consistency problem for a $\mathcal{SHOIQ}$ knowledge base. Section 3 describes a 2EXPSPACE tableaux-based algorithm for checking consistency of a $\mathcal{SHOIQ}$ knowledge base. An advantage of this algorithm is that a tree-like structure can be maintained to obtain termination. Section 4 transfers a result in [9] from $\mathcal{C}^2$ to $\mathcal{SHOIQ}$. Based on the these results, we propose an EXPSPACE tableaux-based algorithm for $\mathcal{SHOIQ}$. Finally, we discuss the results and future work.

For the lack of place, we refer the reader to [10] for examples and full proofs.

2 The Description Logic $\mathcal{SHOIQ}$

In this section, we present the syntax and the semantics 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 $\underline{\circledast}$ is defined as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{R^\ominus \sqsubseteq S^\ominus \mid R \sqsubseteq S \in \mathcal{R}\}$. We define a function* $\mathsf{Trans}(R)$ *which returns* true *iff there is some $Q \in \mathbf{R}_+ \cup \{P^\ominus \mid P \in \mathbf{R}_+\}$ such that $Q \underline{\circledast} R$. A role R is called* simple *w.r.t. $\mathcal{R}$ if* $\mathsf{Trans}(Q) = \mathsf{false}$. *An* interpretation *$\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ (*domain*) and a function $\cdot^\mathcal{I}$ which maps each role name to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that $R^{-\mathcal{I}} = \{\langle x, y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y, x\rangle \in R^\mathcal{I}\}$ for all $R \in \mathbf{R}$, and $\langle x, z\rangle \in S^\mathcal{I}, \langle z, y\rangle \in S^\mathcal{I}$ implies $\langle x, y\rangle \in S^\mathcal{I}$ 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 $\mathcal{I} \models \mathcal{R}$.*

Notice that the simplicity of roles which relies on the function $\mathsf{Trans}(\cdot)$ plays a crucial role in guaranteeing decidability of $\mathcal{SHIQ}$ [11]. 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-neighbour of s.

Definition 2 (terminology). *Let $\mathbf{C}$ be a non-empty set of* concept names *with a non-empty subset $\mathbf{C}_o \subseteq \mathbf{C}$ of nominals. The set of $\mathcal{SHOIQ}$-concepts is inductively defined as the smallest set containing all C in $\mathbf{C}$, $\top$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n\, S.C)$ and $(\geq n\, S.C)$ where n is a positive integer, C and D are $\mathcal{SHOIQ}$-concepts, R is an $\mathcal{SHOIQ}$-role and S is a simple role w.r.t. a role hierarchy. We denote $\bot$ for $\neg\top$. The interpretation function $\cdot^\mathcal{I}$ of an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ maps each concept name to a subset of $\Delta^\mathcal{I}$ such that $\top^\mathcal{I} = \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$, $\mathsf{card}\{o^\mathcal{I}\} = 1$ for all $o \in \mathbf{C}_o$, $(\exists R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}, \langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$, $(\forall R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \forall y \in \Delta^\mathcal{I}, \langle x, y\rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\}$, $(\geq n\, S.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \mathsf{card}\{y \in C^\mathcal{I} \mid \langle x, y\rangle \in S^\mathcal{I}\} \geq n\}$, $(\leq n\, S.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \mathsf{card}\{y \in C^\mathcal{I} \mid \langle x, y\rangle \in S^\mathcal{I}\} \leq n\}$ where* $\mathsf{card}\{S\}$ *is denoted for the cardinality of a set S.*

* $C \sqsubseteq D$ *is called a general concept inclusion (GCI) where* C, D *are* $\mathcal{SHOIQ}$-*concepts (possibly complex), and a finite set of GCIs is called a terminology* $\mathcal{T}$.
* *An interpretation* $\mathcal{I}$ *satisfies a GCI* $C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *and* $\mathcal{I}$ *satisfies a terminology* $\mathcal{T}$ *if* $\mathcal{I}$ *satisfies each GCI in* $\mathcal{T}$. *Such an interpretation is called a* model *of* $\mathcal{T}$, *denoted by* $\mathcal{I} \models \mathcal{T}$.

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* $(\mathcal{T}, \mathcal{R})$.

Thanks to the reductions between unsatisfiability, subsumption of concepts and knowledge base consistency, 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 [11]. 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 can define a set $\mathsf{cl}(\mathcal{T}, \mathcal{R})$. For the sake of brevity, we refer the reader to [7] for a more complete definition.

To prove soundness and completeness of our algorithms, we need a tableau structure that represents a model of a $\mathcal{SHOIQ}$ knowledge base. Regarding the definition of tableaux for $\mathcal{SHOIQ}$ presented in [7], we add a new property, namely P15. This new property imposes an exact number of S-neighbour individuals t of s if $(\leq nS.C) \in \mathcal{L}(s)$. This property makes explicit nondeterminism implied from the semantics of $(\leq nS.C)$ and requires an extra expansion rule, namely $\bowtie$-rule, introduced in Figure 1 (Appendix). 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.

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 $\{(\leq mS.C), (\geq mS.C)\} \subseteq \mathcal{L}(s)$.

It is not hard to prove that there is a tableau with the new property P15 for a $\mathcal{SHOIQ}$ knowledge base $(\mathcal{T}, \mathcal{R})$ iff $(\mathcal{T}, \mathcal{R})$ is consistent. A proof of a similar result for $\mathcal{SHIQ}$ tableaux can be found in [12].

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

In this section, we introduce a structure, called $\mathcal{SHOIQ}$-forest. We will show that such a forest is sufficient to represent a model of a $\mathcal{SHOIQ}$-knowledge base.

Definition 4 ($\mathcal{SHOIQ}$-tree). *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}$ *knowledge base. For each* $o \in \mathbf{C}_o$, *a* $\mathcal{SHOIQ}$-*tree for* $(\mathcal{T}, \mathcal{R})$, *denoted by* $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$, *is defined as follows:*

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

* E_o *is a set of edges. Each edge* $\langle x,y\rangle \in E_o$ *is labelled with a function* $\mathcal{L}_o$ *such that* $\mathcal{L}_o(\langle x,y\rangle) \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$. *If* $\langle x,y\rangle \in E_o$ *then* y *is called a successor of* x, *denoted by* $y \in \mathsf{succ}^1(x)$, *or* x *is called the predecessor of* y, *denoted by* $x = \mathsf{pred}^1(y)$. *In this case, we say that* x *is a neighbour of* y *or* y *is a neighbour of* x. *If* $z \in \mathsf{succ}^n(x)$ *(resp.* $z = \mathsf{pred}^n(x)$*) and* y *is a successor of* z *(resp.* y *is the predecessor of* z*) then* $y \in \mathsf{succ}^{(n+1)}(x)$ *(resp.* $y = \mathsf{pred}^{(n+1)}(x)$*) for all* $n \geq 0$ *where* $\mathsf{succ}^0(x) = \{x\}$ *and* $\mathsf{pred}^0(x) = x$. *A node* y *is called a* descendant *of* x *if* $y \in \mathsf{succ}^n(x)$ *for some* $n > 0$. *A node* y *is called an* ancestor *of* x *if* $y = \mathsf{pred}^n(x)$ *for some* $n > 0$. *To ensure that* $\mathbf{T}_o$ *is a tree, it is required that (i)* x *is a descendant of* $\widehat{x}_o$ *for all* $x \in V_o$ *with* $x \neq \widehat{x}_o$, *and (ii) each node* $x \in V_o$ *with* $x \neq \widehat{x}_o$ *has a unique predecessor. A node* y *is called an* R*-successor of* x, *denoted by* $y \in \mathsf{succ}^1_R(x)$ *(resp.* y *is called the* R*-predecessor of* x, *denoted by* $y = \mathsf{pred}^1_R(x)$*) if there is some role* R' *such that* $R' \in \mathcal{L}_o(\langle x,y\rangle)$ *(resp.* $R' \in \mathcal{L}_o(\langle y,x\rangle)$*) and* $R' \sqsubseteq_* R$. *A node* y *is called a* R*-neighbour of* x *if* y *is either a* R*-successor or* R*-predecessor of* x. *If* z *is an* R*-successor of* y *(resp.* z *is the* R*-predecessor of* y*) and* $y \in \mathsf{succ}^n_R(x)$ *(resp.* $y = \mathsf{pred}^n_R(x)$*) then* $z \in \mathsf{succ}^{(n+1)}_R(x)$ *(resp.* $z = \mathsf{pred}^{(n+1)}_R(x)$*) for* $n \geq 0$ *with* $\mathsf{succ}^0_R(x) = \{x\}$ *and* $x = \mathsf{pred}^0_R(x)$.

* *For a node* x, *a role* S *and* $o \in \mathbf{C}_o$, *we define the set* $S^{\mathbf{T}_o}(x,C)$ *of* x*'s* S*-neighbours as follows: :* $S^{\mathbf{T}_o}(x,C) = \{y \in V_o \mid y$ *is a* S*-neighbour of* x *and* $C \in \mathcal{L}_o(x)\}$.

* *A node* x *is called* iterated *by* y *w.r.t. a node* x_o *if* x *has no nominal ancestor except for* $\widehat{x}_o$ *and there are integers* $n, m > 0$ *and nodes* x', y' *such that : (i)* $x_o = \mathsf{pred}^n(y)$, $y = \mathsf{pred}^m(x)$, *(ii)* $x' = \mathsf{pred}^1(x)$, $y' = \mathsf{pred}^1(y)$, *(iii)* $\mathcal{L}_o(x) = \mathcal{L}_o(y)$, $\mathcal{L}_o(x') = \mathcal{L}_o(y')$, *(iv)* $\mathcal{L}_o(\langle x',x\rangle) = \mathcal{L}_o(\langle y',y\rangle)$, *and (v) if there are* z, z' *and* $i > 0$ *such that* $z' = \mathsf{pred}^1(z)$, $\mathsf{pred}^i(z') = x_o$, $\mathcal{L}_o(z) = \mathcal{L}_o(y)$, $\mathcal{L}_o(z') = \mathcal{L}_o(y')$ *and* $\mathcal{L}_o(\langle z',z\rangle) = \mathcal{L}_o(\langle y',y\rangle)$ *then* $i \geq n$.
A node x *is called* 1-iterated *by* y *if* x *is iterated by* y *w.r.t.* $\widehat{x}_o$. *A node* x *is called* blocked *by* y, *denoted by* $y = \mathsf{b}(x)$, *if* x *is iterated by* y *w.r.t. a 1-iterated node* x_o.

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

We can remark that the definition of 1-iterated nodes in Definition 4 for $\mathcal{SHOIQ}$-trees is very similar to the standard definition of blocked nodes for $\mathcal{SHIQ}$ completion trees (see [11]). Moreover, if we consider the subtree rooted at a 1-iterated node as a $\mathcal{SHIQ}$ completion tree then blocked nodes according to Definition 4 are also blocked nodes according to the standard definition for this $\mathcal{SHIQ}$ completion tree.

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

Definition 5 **($\mathcal{SHOIQ}$-forest).** *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A $\mathcal{SHOIQ}$-*forest *for $(\mathcal{T}, \mathcal{R})$ is a pair $\mathbf{G} = \langle \mathbf{T}, \varphi \rangle$, where $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ is a set of $\mathcal{SHOIQ}$-trees for $(\mathcal{T}, \mathcal{R})$ with $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$, and φ is a* partitioning *function $\varphi : \mathcal{V} \to 2^{\mathcal{V}}$ with $\mathcal{V} = \bigcup_{o \in \mathbf{C}_o} V_o$. We denote $\mathcal{L}'(\langle x, y \rangle) = \mathcal{L}_o(\langle x, y \rangle)$ if $\langle x, y \rangle \in E_o$, and $\mathcal{L}'_o(\langle x, y \rangle) = \{S^{\ominus} \mid S \in \mathcal{L}_o(\langle y, x \rangle)\}$ if $\langle y, x \rangle \in E_o$ for some $o \in \mathbf{C}_o$. The partitioning function φ satisfies the following conditions:*

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

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

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

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

Propagation Propagate$(\varphi(x), \varphi(x'), \varphi(y))$ is an operation which propagates (i) node labels from a partition $\varphi(x)$ to another partition $\varphi(x')$, and vice versa, (ii) edge labels

from the edges ended by nodes of $\varphi(x)$ and $\varphi(y)$ to the edges ended by nodes of $\varphi(x')$ and $\varphi(y)$, and vice versa. In other terms, $\mathsf{Propagate}(\cdots)$ merges $\varphi(x)$ into $\varphi(x')$, and $\langle\varphi(x), \varphi(y)\rangle$ into $\langle\varphi(x'), \varphi(y)\rangle$. More precisely, let $\mathbf{G} = \langle\mathbf{T}, \varphi\rangle$ be a $\mathcal{SHOIQ}$-forest with $\mathbf{T} = \{\mathbf{T}_o \mid o \in \mathbf{C}_o\}$ and $\mathbf{T}_o = (V_o, E_o, \mathcal{L}_o, \widehat{x}_o, \neq_o)$. $\mathsf{Propagate}(\varphi(x), \varphi(x'), \varphi(y))$ updates the label of nodes and edges in $\mathbf{T}$ as follows:
1. $\mathcal{L}(z) = \mathcal{L}(x) \cup \mathcal{L}(x')$ for all $z \in \varphi(x) \cup \varphi(x')$,
2. for all $z, z' \in \varphi(x) \cup \varphi(x')$ and $w, w' \in \varphi(y)$, if z is a S-neighbour of w and $\mathcal{L}'(\langle z', w'\rangle) \neq \emptyset$ then (i) if z' is a successor of w' and $S \notin \mathcal{L}(\langle w', z'\rangle)$ then $\mathcal{L}(\langle w', z'\rangle) = \mathcal{L}(\langle w', z'\rangle) \cup \{S\}$, (ii) if w' is a successor of z' and $S \notin \mathcal{L}(\langle z', w'\rangle)$ then $\mathcal{L}(\langle z', w'\rangle) = \mathcal{L}(\langle z', w'\rangle) \cup \{S^{\ominus}\}$.

The rules in Figure 1 (Appendix) maintain the tree-like structure of $\mathcal{SHOIQ}$-forest and they are similar to those in [7] except that if a concept C is added to the label of a node x due to application of these rules then C is propagated to the label of each node $y \in \varphi(x)$. Moreover, all rules in Figure 1 except for $\exists$- and $\geq$-rule update only the label of nodes or edges and do no change on the partitioning function φ. Especially, when the $\leq$-rule is applied to a node x with two S-neighbours y, z of x, it must propagate the label of $\langle x, y\rangle$ to that of all $\langle x', z'\rangle$ (or $\langle z', x'\rangle$) where $x' \in \varphi(x)$ and $z' \in \varphi(z)$, and set the label of $\langle x, y\rangle$ to empty set. This may change φ only if $\varphi(y)$ is singleton. By the $\bowtie$-rule in Figure 2, each node x containing a term $(\leq nS.C)$ has exactly m S-neighbours containing C with some $m \leq n$. As a result, this rule and $\geq$-rule ensure that if there are two nodes $y, y' \in \varphi(x)$ then y and y' have exactly m S-neighbours which contain C in their label. Finally, we can avoid infinite sequences of "merging-and-generating" without pruning nodes since all merges due to number restrictions or nominals are performed by updating the partitioning function.

The following lemma establishes correctness and completeness of the algorithm.

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

1. *The tableaux algorithm terminates and builds a $\mathcal{SHOIQ}$-forest whose the size is bounded by a double exponential function in the size of $(\mathcal{T}, \mathcal{R})$.*
2. *If the tableaux algorithm yields a clash-free and complete $\mathcal{SHOIQ}$-forest for $(\mathcal{T}, \mathcal{R})$ then there is a tableau for $(\mathcal{T}, \mathcal{R})$.*
3. *If there is a tableau for $(\mathcal{T}, \mathcal{R})$ then the tableaux algorithm yields a clash-free and complete $\mathcal{SHOIQ}$-forest for $(\mathcal{T}, \mathcal{R})$.*

It is straightforward to show that the size of a $\mathcal{SHOIQ}$-forest is bounded by a double exponential function in the size of $(\mathcal{T}, \mathcal{R})$. To prove soundness of the tableaux algorithm, we can devise a model from a clash-free and complete $\mathcal{SHOIQ}$-forest by considering a partition as an individual and unraveling blocked nodes since we can show that each blocking node $\mathsf{b}(x)$ has no "core path" from $\mathsf{b}(x)$ to every nominal descendant y, i.e., there do not exist terms $(\leq m_i R_i.C_i) \in \mathsf{pred}^i(y)$, roles $R_i \in \mathcal{L}(\langle\mathsf{pred}^{i-1}(y), \mathsf{pred}^i(y)\rangle)$ and concepts $C_i \in \mathcal{L}(\mathsf{pred}^{i+1}(y))$ for $k < i \leq 0$ with $\mathsf{b}(x) = \mathsf{pred}^k(y)$.

The following theorem is a consequence of Lemma 1.

Theorem 1. *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. The tableaux algorithm is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in* 2NEXPTIME *in the size of $(\mathcal{T}, \mathcal{R})$.*

4 An EXPSPACE tableaux-based algorithm for $\mathcal{SHOIQ}$

This section starts by translating from results presented in [9] for $\mathcal{C}^2$ into those for $\mathcal{SHOIQ}$.

Definition 6 (star-type). *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A* star-type *is a triplet $\sigma = \langle \lambda_\sigma, \bar{\nu}_\sigma, \bar{\mu}_\sigma \rangle$, where $\lambda_\sigma \in 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$, $\bar{\nu}_\sigma$ contains at most a pair $\langle r, l \rangle \in 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ and $\bar{\mu}_\sigma = (\langle r_1, l_1 \rangle, \cdots, \langle r_{d_\sigma}, l_{d_\sigma} \rangle)$ is a d-tuple over $2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$. A pair $\langle r', l' \rangle$ is a* ray *of σ if $\langle r', l' \rangle$ is a component of $\bar{\mu}_\sigma$ or $\langle r', l' \rangle \in \bar{\nu}_\sigma$. We define an inequality relation $\neq$ over the set of rays. A ray $\langle r', l' \rangle$ of σ is* primary *w.r.t. a term $(\leq mR.C)$, denoted $\langle r', l' \rangle_{\langle \leq mR.C \rangle}$, if $(\leq mR.C) \in \lambda_\sigma$, $R \in r'$ and $C \in l'$. For a term $(\leq mR.C) \in \lambda_\sigma$, we denote $\mathcal{C}^\sigma_{\langle \leq mR.C \rangle}$ for the set of all rays $\langle r', l' \rangle$ of σ such that $R \in r'$, $C \in l'$.*

- *A star-type σ is* nominal *if $o \in \lambda_\sigma$ for some $o \in \mathbf{C}_o$.*
- *A star-type σ is* isomorph *to a star-type σ' if $\lambda_\sigma = \lambda_{\sigma'}$, and for each term $(\leq mR.C) \in \lambda_\sigma$, there is an injection $\pi : \mathcal{C}^\sigma_{\langle \leq mR.C \rangle} \to \mathcal{C}^{\sigma'}_{\langle \leq mR.C \rangle}$ such that $\pi(\langle r, l \rangle) = (\langle r, l \rangle)$.*
- *Two star-types σ, σ' are* isomorph *if $\lambda_\sigma = \lambda_{\sigma'}$, and for each term $(\leq mR.C) \in \lambda_\sigma$, there is a bijection $\pi : \mathcal{C}^\sigma_{\langle \leq mR.C \rangle} \to \mathcal{C}^{\sigma'}_{\langle \leq mR.C \rangle}$ such that $\pi(\langle r, l \rangle) = (\langle r, l \rangle)$.*
- *A star-type $\sigma = \langle \lambda, \bar{\nu}, \bar{\mu} \rangle$ with $\bar{\mu} = (\langle r_1, l_1 \rangle, \cdots, \langle r_{d_\sigma}, l_{d_\sigma} \rangle)$ and $\lambda = l_0$, $\bar{\nu} = \{\langle r_{d_\sigma+1}, l_{d_\sigma+1} \rangle\}$, is* valid *if the following conditions are satisfied:*
  1. *If $C \sqsubseteq D \in \mathcal{T}$ then $\mathsf{nnf}(\neg C \sqcup D) \in l_i$ for all $0 \leq i \leq d_\sigma + 1$;*
  2. *$\{A, \neg A\} \not\subseteq l_i$ for every concept name A with $0 \leq i \leq d_\sigma + 1$;*
  3. *If $C_1 \sqcap C_2 \in l_i$ then $\{C_1, C_2\} \subseteq l_i$ for all $0 \leq i \leq d_\sigma + 1$;*
  4. *If $C_1 \sqcup C_2 \in l_i$ then $\{C_1, C_2\} \cap l_i \neq \emptyset$ for all $0 \leq i \leq d_\sigma + 1$;*
  5. *If $\exists R.C \in \lambda$ then there is some $1 \leq i \leq d_\sigma + 1$ such that $C \in l_i$ and $R \in r_i$;*
  6. *If $(\leq nS.C) \in \lambda$ and there is some $1 \leq i \leq d_\sigma + 1$ such that $S \in r_i$ then $C \in l_i$ or $\dot{\neg} C \in l_i$;*
  7. *If $(\leq nS.C) \in \lambda$ and there is some $1 \leq i \leq d_\sigma + 1$ such that $C \in l_i$ and $S \in r_i$ then there is some $1 \leq m \leq n$ such that $\{(\leq mS.C), (\geq mS.C)\} \subseteq \lambda$;*
  8. *For each $1 \leq i \leq d_\sigma + 1$, if $R \in r_i$ and $R \sqsubseteq^* S$ then $S \in r_i$;*
  9. *If $\forall R.C \in \lambda$ and $R \in r_i$ for some $1 \leq i \leq d_\sigma + 1$ then $C \in l_i$;*
  10. *If $\forall R.D \in \lambda$, $S \sqsubseteq^* R$, $\mathsf{Trans}(S)$ and $R \in r_i$ for some $1 \leq i \leq d_\sigma + 1$ then $\forall S.D \in l_i$;*
  11. *If $(\geq nS.C) \in \lambda$ then there are $1 \leq i_1 < \cdots < i_n \leq d_\sigma + 1$ such that $C \in l_{i_j}$ and $S \in r_{i_j}$ for all $1 \leq j \leq n$;*
  12. *If $(\leq nS.C) \in \lambda$ and there are no $1 \leq i_1 < \cdots < i_{n+1} \leq d_\sigma + 1$ such that $C \in l_{i_j}$ and $S \in r_{i_j}$ for all $1 \leq j \leq n$;*

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

In the context of a $\mathcal{SHOIQ}$-forest, we can think of a star-type σ as the set of nodes which satisfy λ_σ and have R-neighbours such that R is included in their rays. Moreover, we can merge nodes satisfying homomorph and isomorph star-types without violating semantic constraints imposed by node and edge labels. A star-type σ is valid if no expansion rule is applicable to a node whose label is λ_σ.

Definition 7 (frame). *Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A* frame *for $(\mathcal{T},\mathcal{R})$ is a tuple $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$, where $H \in \mathbb{N}$ is the* dimension *of $\mathcal{F}$, $N_i \subseteq \Sigma$ for all $0 \leq i \leq H$, and all star-types in $\mathcal{N}_0$ are nominal, δ is a function $\delta : \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i \rightarrow \mathbb{N}$, Φ is a function $\Phi : \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i \rightarrow 2^{\bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i}$, and $\widehat{\delta}$ is a function $\delta' : \Phi(\bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i) \rightarrow \mathbb{N}$;*

1. *Two star-types $\sigma, \sigma' \in \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i$ are* mergeable in *$\mathcal{F}$, denoted $\sigma \approx \sigma'$, if, either σ and σ' are homomorph to a star-type σ_0; or σ and σ' are isomorph. The relation of mergeability $\approx$ is an equivalence relation over $\bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i$. We denote $\Phi(\sigma) = \{\sigma' \mid \sigma' \approx \sigma\}$ and $\Phi(\sigma)$ is called* mergeable. *We say that $\Phi(\sigma)$ is* homomorph *w.r.t. a star-type σ_0 if σ' is homomorph to σ_0 for all $\sigma' \in \Phi(\sigma)$. We say that $\Phi(\sigma)$ is* isomorph *if σ', σ'' are isomorph for all $\sigma', \sigma'' \in \Phi(\sigma)$. For each $\Phi(\sigma)$, we define a set of rays of $\Phi(\sigma)$ as follows:*
   - *If $\Phi(\sigma)$ is homomorph w.r.t. a star-type $\sigma_0 \in \Phi(\sigma)$ and $\langle r', l'\rangle$ is a primary ray of σ_0 then we define a primary ray $\langle r, l\rangle$ of $\Phi(\sigma)$ with $r = r'$ and $l = l'$;*
   - *If $\Phi(\sigma)$ is isomorph and $\langle r', l'\rangle$ is a primary ray of some fixed star-type $\sigma_0 \in \Phi(\sigma)$ then we define a primary ray $\langle r, l\rangle$ of $\Phi(\sigma)$ with $r = r'$ and $l = l'$;*
   - *If $\langle r', l'\rangle$ is a non primary ray of $\Phi(\sigma)$ then there is some $\sigma' \in \Phi(\sigma)$ that has a non primary ray $\langle r, l\rangle$ such that $r = r'$ and $l = l'$.*

   We denote $\mathcal{C}^{\Phi(\sigma)}$ for the set of all rays of $\Phi(\sigma)$, and $\mathcal{C}^{\Phi(\sigma)}_{\langle\leq mR.C\rangle} = \{\langle r', l'\rangle \in \mathcal{C}^{\Phi(\sigma)} \mid R \in r, C \in l\}$.
2. *A star-type $\sigma \in \mathcal{N}_k$ ($0 \leq k \leq H$) is called* linkable *with a star-type $\sigma' \in \mathcal{N}_{k-1} \cup \mathcal{N}_{k+1}$ by a ray $\langle r, l\rangle$ of σ if σ' has a ray $\langle r', l'\rangle$ such that $l = \lambda_{\sigma'}$, $l' = \lambda_\sigma$ and $r = r'^-$ where $r'^- = \{R^\ominus \mid R \in r'\}$.*

The frame structure, as introduced in Definition 7, allows us to tile star-types to obtain a forest structure. Such a structure is crucial to obtain termination when designing a tableaux-based algorithm. An important difference between a frame and a $\mathcal{SHOIQ}$-forest is that a frame does not represent nodes corresponding to individuals but store the number of individuals satisfying a star-type. The function $\delta(\sigma)$ is used for this purpose. According to Lemma 1, the number of a $\mathcal{SHOIQ}$-forest's nodes may be double exponential in the size of a $\mathcal{SHOIQ}$ knowledge base $(\mathcal{T},\mathcal{R})$ while the number of distinct star-types is bounded by an exponential function since star-types are built from the signature of $(\mathcal{T},\mathcal{R})$. This implies that $\delta(\sigma)$ may take a double exponential value. In the context of a $\mathcal{SHOIQ}$-forest, we can think of a $\Phi(\sigma)$ as the set of partitions each of which satisfies all $\sigma' \in \Phi(\sigma)$. The function $\widehat{\delta}(\sigma)$ is used to store the number of partitions satisfying all $\sigma' \in \Phi(\sigma)$.

Definition 8 (valid frame). *Let $(\mathcal{T},\mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. A frame $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$ is* valid *if the following conditions are satisfied:*

1. *For each $\sigma \in \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i$, if $\delta(\sigma) \geq 1$ then σ is valid;*
2. *For each $o \in \mathbf{C}_o$ there is a unique $\sigma_o \in \mathcal{N}_0$ such that $o \in \lambda_{\sigma_o}$ and $\delta(\sigma_o) = 1$;*
3. *For each $o \in \mathbf{C}_o$, $\Phi(\sigma_o) = \{\sigma \in \bigcup_{i\in\{1,\cdots,H\}} \mathcal{N}_i \mid o \in \lambda_\sigma\}$ and $\widehat{\delta}(\Phi(\sigma_o)) = 1$;*

4. *For each* $0 \leq k < H$ *and* $\langle \lambda, r, \lambda' \rangle \in 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})} \times 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ *with* $r^- = \{R^\ominus \mid R \in r\}$*,*

$$\sum_{\sigma \in \mathcal{N}_k} \delta(\sigma)|\bar{\mu}_\sigma|_{\langle \lambda, r, \lambda' \rangle} = \sum_{\sigma' \in \mathcal{N}_{k+1}} \delta(\sigma')|\bar{\nu}_{\sigma'}|_{\langle \lambda', r^-, \lambda \rangle} \tag{1}$$

where $|\bar{\nu}_\omega|_{\langle \lambda, r, \lambda' \rangle}$ *and* $|\bar{\mu}_\omega|_{\langle \lambda, r, \lambda' \rangle}$ *are denoted for the number of components* $\langle r', l' \rangle$ *of respective* $\bar{\nu}_\omega$ *and* $\bar{\mu}_\omega$ *such that* $\lambda_\omega = \lambda$*,* $r' = r$ *and* $l' = \lambda'$ *for a star-type* $\omega = \langle \lambda_\omega, \bar{\nu}_\omega, \bar{\mu}_\omega \rangle$*;*
5. *For each* $\langle \lambda, r, \lambda' \rangle \in 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})} \times 2^{\mathbf{R}_{(\mathcal{T},\mathcal{R})}} \times 2^{\mathsf{cl}(\mathcal{T},\mathcal{R})}$ *with* $r^- = \{R^\ominus \mid R \in r\}$*,*

$$\sum_{\Phi(\sigma)} \widehat{\delta}(\Phi(\sigma))|\Phi(\sigma)|_{\langle \lambda, r, \lambda' \rangle} = \sum_{\Phi(\sigma')} \widehat{\delta}(\Phi(\sigma'))|\Phi(\sigma'))|_{\langle \lambda', r^-, \lambda \rangle} \tag{2}$$

where $|\Phi(\omega)|_{\langle l, s, l' \rangle}$ *is denoted for the number of rays* $\langle u, h \rangle$ *of* $\Phi(\omega)$ *with some star-type* ω *such that* $\lambda_\omega = l$*,* $u = s$ *and* $h = l'$*.*
6. *For each* $\Phi(\sigma)$ *with* $\sigma \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$*, and for each term* $(\leq mR.C) \in \lambda_\sigma$*,*

$$\mathsf{card}\{\mathcal{C}^{\Phi(\sigma)}_{\langle \leq mR.C \rangle}\} \leq m \tag{3}$$

The notion of validity for a frame is crucial to establish a connection with the tableaux-based algorithm presented in Section 3, i.e., how to build a $\mathcal{SHOIQ}$-forest from a valid frame, and inversely. Condition 1 in Definition 8 requires that every star-type counted by δ must be valid. Condition 2 and 3 ensure that each nominal is counted exactly once. In the context of a $\mathcal{SHOIQ}$-forest, these conditions imply that for each nominal o there is exactly one tree whose root contains o and there is exactly one partition contains o. Condition 4 allows for linking star-types at level k with star-types at level $k-1$ and $k+1$. It ensures that each node x satisfying (or counted for) a star-type σ at level k is linked by its rays to neighbours satisfying star-types at level $k-1$ and $k+1$. The number of these neighbours corresponds exactly to the number of x's rays. Condition 5 guarantees that each partition satisfying $\Phi(\sigma)$ can be linked exactly with another partition via a ray of $\Phi(\sigma)$. Finally, Condition 5 ensures that each partition satisfying $\Phi(\sigma)$ with $(\leq mR.C) \in \lambda_\sigma$ can be linked at most with m partitions containing C via rays that include R.

Lemma 2. *Let* $(\mathcal{T}, \mathcal{R})$ *be a* $\mathcal{SHOIQ}$ *knowledge base.*

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

Lemma 2 points out the equivalence between a clash-free and complete $\mathcal{SHOIQ}$-forest and a valid frame for $(\mathcal{T}, \mathcal{R})$. The following lemma affirms that there is an exponential structure, a valid frame, which can represent a $\mathcal{SHOIQ}$-forest whose size may be double exponential.

Lemma 3. *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. The size of a valid frame $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$ is bounded by an exponential function in the size of $(\mathcal{T}, \mathcal{R})$.*

We can sketch a proof of the lemma here. We have $H \leq K$ where $K = 2^{2m+k} \times 2$ with $m = \mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\}$ and $k = \mathsf{card}\{\mathbf{R}_{(\mathcal{T},\mathcal{R})}\}$. $\mathsf{card}\{\Sigma\} \leq (\mathsf{card}\{\mathsf{cl}(\mathcal{T}, \mathcal{R})\})^2 \times \mathsf{card}\{\mathbf{R}_{(\mathcal{T},\mathcal{R})}\}$ $\delta(\sigma) \leq M^{2^{2m+k} \times 2}$ where $M = \sum m_i + E$, m_i occurs in a number restriction term $(\geq m_i R.C)$ appearing in $\mathcal{T}$, and E is the number of distinct terms $\exists R.C$ appearing in $\mathcal{T}$ for $\sigma \in \Sigma$. If $\delta(\sigma)$ is represented as a binary number then it takes an exponential number of bits.

Based on Lemma 3 and 2, we can present straightforwardly an optimal worst-case algorithm for checking the consistency of a $\mathcal{SHOIQ}$ knowledge base. However, such an algorithm cannot be used in practice since there are tremendously non-determinisms which must be dealt with when constructing a valid frame. In the sequel, based on the results obtained so far, we try to design an algorithm which has more goal-directed behaviours.

Blocking condition for a frame Let $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$ be a frame. A star-type $\sigma_k \in \mathcal{N}_k$ with $0 < k \leq H$ is *blocked* if there are $\sigma_i \in \mathcal{N}_i$ with $0 \leq i \leq k$ such that σ_i is linkable with σ_{i-1} for all $i \in \{1, \cdots, k\}$, then there are $0 < k_1 < k_2 < k_3 < k_4 \leq k$ such that:

1. $\lambda_{\sigma_{k_1}} = \lambda_{\sigma_{k_2}}$, $\bar{\nu}_{\sigma_{k_1}} = \bar{\nu}_{\sigma_{k_2}}$, and there is no $0 < j < k_2$ such that $j \neq k_1$, $\lambda_{\sigma_j} = \lambda_{\sigma_{k_2}}$ and $\bar{\nu}_{\sigma_j} = \bar{\nu}_{\sigma_{k_2}}$;
2. $\lambda_{\sigma_{k_3}} = \lambda_{\sigma_{k_4}}$, $\bar{\nu}_{\sigma_{k_3}} = \bar{\nu}_{\sigma_{k_4}}$, and there is no $k_2 < j < k_4$ such that $j \neq k_3$, $\lambda_{\sigma_j} = \lambda_{\sigma_{k_4}}$ and $\bar{\nu}_{\sigma_j} = \bar{\nu}_{\sigma_{k_4}}$.

Notice that this blocking condition is looser than the blocking condition introduced in Definition 5 for a $\mathcal{SHOIQ}$-forest. Since we can not determine the path from root to a node satisfying a star-type over a frame, it not possible to check blocking condition in the same way as for a $\mathcal{SHOIQ}$-forest. The blocking condition for a frame, as described above, implies that a node satisfying a blocked star-type must have an ancestor which is blocked according to the blocking condition for a $\mathcal{SHOIQ}$-forest.

We are now ready to propose an EXPSPACE tableaux-based algorithm for $\mathcal{SHOIQ}$. Before applying the frame rules described in Figure 3 (Appendix), we initialise a frame $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$ from a $(\mathcal{T}, \mathcal{R})$ knowledge base as follows:
$\mathcal{N}_0 := \{\langle\{o\}, \emptyset, \emptyset\rangle \mid o \in \mathbf{C}_o\}$; $\delta(\sigma_o) := 1$, $\Phi(\sigma_o) = \{\sigma_o\}$ and $\delta'(\Phi(\sigma_o)) = 1$ for all $\sigma_o \in \mathcal{N}_0$.

If no frame rule is applicable to all star-types of $\mathcal{F}$ then we say that $\mathcal{F}$ is complete. If we obtain a valid and complete $\mathcal{F}$ by applying the frame rules from a $(\mathcal{T}, \mathcal{R})$, then we conclude that $(\mathcal{T}, \mathcal{R})$ is consistent. Otherwise, $(\mathcal{T}, \mathcal{R})$ is not consistent.

Soundness of the tableaux-based algorithm for building a frame can be established thanks to Lemma 2. Since each frame rule has its counterpart in the expansion rules, completeness of the algorithm can be shown by using the same arguments as those employed to prove Lemma 1. From these results and Lemma 3, we obtain the following main result of the section:

Theorem 2. *Let $(\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOIQ}$ knowledge base. The tableaux algorithm for contructing a frame is a decision procedure for consistency of $(\mathcal{T}, \mathcal{R})$ and it runs in* EXPSPACE *in the size of $(\mathcal{T}, \mathcal{R})$.*

5 Conclusion and Discussion

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

The algorithms proposed in the present paper have used several nondeterministic rules, e.g., $\bowtie$ or $\leq_o$-rules. We think that these rules should be improved in some way such that, for instance, they would take advantage of information from the part of the frame which has already built.

References

1. Patel-Schneider, P., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax. In: W3C Recommendation. (2004)
2. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. Journal of Artificial Intelligence Research **12** (2000) 199–217
3. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
4. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a pratical owl-dl reasoner. Journal of Web Semantics **5**(2) (2007) 51–53
5. Horrocks, I.: The FaCT system. In de Swart, H., ed.: Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98). Volume 1397 of Lecture Notes in Artificial Intelligence., Springer (1998) 307–312
6. Kazakov, Y., Motik, B.: A Resolution-Based Decision Procedure for SHOIQ. Journal of Automated Reasoning **40**(2–3) (2008) 89–116
7. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal Of Automated Reasoning **39**(3) (2007) 249–276
8. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research **36** (2009) 165–228
9. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language and Information **14**(3) (2005) 369–395
10. Le Duc, C., Lamolle, M., Curé, O.: An EXPSPACE tableaux-based algorithm for $\mathcal{SHOIQ}$. In: Technical Report, http://www.iut.univ-paris8.fr/∼leduc/papers/RR2012a.pdf (2012)
11. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999), Springer (1999)
12. Horrocks, I., Sattler, U., Tobies, S.: A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. In: LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen. (1999)
13. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica **69** (2001) 5–40

Appendix

The rules in Figure 3 for building a frame calls the algorithms described in Figure 1, 2, 3 and 4. Basically, these algorithms update the frame by adding a new star-type or modifying the functions $\delta\sigma$ and $\widehat{\delta}\sigma$.

Notation Let $\mathcal{F} = \langle(\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta}\rangle$ be a frame.

- For each $\sigma \in \mathcal{N}_k$, if $k = 0$ then we define $\mathcal{N}(\sigma) = \emptyset$ and $\widehat{\mathcal{N}}(\sigma, \langle e, h\rangle) = \emptyset$;
- For each $\sigma \in \mathcal{N}_k$ with $k > 0$, we denote $\mathcal{N}(\sigma) \subseteq \mathcal{N}_{k-1}$ for the non-empty set with $\mathcal{N}(\sigma) = \mathcal{N}'(\sigma) \cup \{\omega_0\}$ such that $\sum\limits_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega') < \delta(\sigma)$, $\sum\limits_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega') + \delta(\omega_0) \geq \delta(\sigma)$, and for all $\sigma' \in \mathcal{N}(\sigma)$ it holds that $\lambda_{\sigma'} = l_0$, σ' has a ray $\langle r'', l''\rangle \notin \bar{\nu}_{\sigma'}$ with $r'' = r_0^-$ and $l'' = \lambda_\sigma$.
- For each ray $\langle r, h\rangle$ of σ with $\langle r, h\rangle \notin \bar{\nu}_\sigma$, we denote $\widehat{\mathcal{N}}(\sigma, \langle e, h\rangle) \subseteq \mathcal{N}_{k+1}$ for the non-empty set with $\widehat{\mathcal{N}}(\sigma, \langle e, h\rangle) = \widehat{\mathcal{N}'}(\sigma, \langle e, h\rangle) \cup \{\bar{\omega}_0\}$ such that $\sum\limits_{\omega' \in \widehat{\mathcal{N}}'(\sigma, \langle e, h\rangle)} \delta(\omega') < \delta(\sigma)$, $\sum\limits_{\omega' \in \widehat{\mathcal{N}}'(\sigma, \langle e, h\rangle)} \delta(\omega') + \delta(\bar{\omega}_0) \geq \delta(\sigma)$, and for all $\sigma' \in \widehat{\mathcal{N}}(\sigma, \langle e, h\rangle)$ it holds that $\lambda_{\sigma'} = h$, σ' has a ray $\langle r'', l''\rangle \in \bar{\nu}_{\sigma'}$ with $r'' = r^-$ and $l'' = \lambda_\sigma$.

$\sqsubseteq$-rule:	if $C \sqsubseteq D \in \mathcal{T}$ and $\mathsf{nnf}(\neg C \sqcup D) \notin \mathcal{L}(x)$
	then $\mathcal{L}(x') = \mathcal{L}(x') \cup \{\mathsf{nnf}(\neg C \sqcup D)\}$ for all $x' \in \varphi(x)$.
$\sqcap$-rule:	if $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then $\mathcal{L}(x') = \mathcal{L}(x') \cup \{C_1, C_2\}$ for all $x' \in \varphi(x)$.
$\sqcup$-rule:	if $C_1 \sqcup C_2 \in (x)$ and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then $\mathcal{L}(x') = \mathcal{L}_o(x') \cup \{C\}$ with some $C \in \{C_1, C_2\}$ for all $x' \in \varphi(x)$.
$\exists$-rule:	if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked, x is not non-root nominal, and
	2. x has no S-neighbour y with $C \in \mathcal{L}(y)$
	then create a new node y with $\mathcal{L}(\langle x, y\rangle)$=$\{S\}$, $\mathcal{L}(y)$=$\{C\}$ and $\varphi(y) = \{y\}$.
$\forall$-rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, and
	2. there is a S-neighbour y of x such that $C \notin \mathcal{L}(y)$
	then $\mathcal{L}(y') = \mathcal{L}(y') \cup \{C\}$ for all $y' \in \varphi(y)$.
$\forall_+$-rule:	if 1. $\forall S.C \in \mathcal{L}(x)$, and
	2. there is some Q with $\mathsf{Trans}(Q)$ and $Q \underline{\boxast} S$, and
	3. there is an Q-neighbour y of x such that $\forall Q.C \notin \mathcal{L}(y)$
	then $\mathcal{L}(y') = \mathcal{L}(y') \cup \{\forall Q.C\}$ for all $y' \in \varphi(y)$.
ch-rule:	if 1. $(\leq n\ S.C) \in \mathcal{L}(x)$, and
	2. there is an S-neighbour y of x with $\{C, \dot{\neg} C\} \cap \mathcal{L}(y) = \emptyset$
	then $\mathcal{L}(y') = \mathcal{L}_o(y') \cup \{E\}$ with some $E \in \{C, \dot{\neg} C\}$ for all $y' \in \varphi(y)$.
$\geq$-rule:	if 1. $(\geq n\ S.C) \in \mathcal{L}(x)$, x is not blocked, x is not non-root nominal, and
	2. x has no n S-neighbours $y_1, ..., y_n$ such that $C \in \mathcal{L}(y_i)$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$,
	then create n new nodes $y_1, ..., y_n$ with $\mathcal{L}(\langle x, y_i\rangle)$=$\{S\}$, $\mathcal{L}(y_i)$=$\{C\}$, $\varphi(y_i) = \{y_i\}$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\leq$-rule:	if 1. $(\leq n\ S.C) \in \mathcal{L}(x)$, and
	2. $\mathsf{card}\{S^{\mathbf{T}}(x, C)\} > n$ and there are two S-neighbours y, z of x with $C \in \mathcal{L}(y) \cap \mathcal{L}(z)$, y is not an ancestor of z and not $y \neq z$
	then 1. for all $z' \in \varphi(z)$, $x' \in \varphi(x)$ such that $\mathcal{L}'(\langle x', z'\rangle) \neq \emptyset$,
	if x' is an ancestor of z' then $\mathcal{L}(\langle x', z'\rangle) = \mathcal{L}(\langle x', z'\rangle) \cup \mathcal{L}(\langle x, y\rangle)$
	else $\mathcal{L}(\langle z', x'\rangle) = \mathcal{L}(\langle z', x\rangle) \cup \{R^{\ominus} \mid R \in \mathcal{L}(\langle x, y\rangle)\}$
	2. $\mathcal{L}(z') = \mathcal{L}(z') \cup \mathcal{L}(y)$ for all $z' \in \varphi(z)$ and $\mathcal{L}(\langle x, y\rangle) = \emptyset$
	3. add $u \neq z$ for all u such that $u \neq y$.
$\bowtie$-rule:	if 1. $(\leq nR.C) \in \mathcal{L}(x)$, $\{(\leq l\ R.C), (\geq l\ R.C)\} \not\subseteq \mathcal{L}(x)$ for all $l \leq n$,
	2. $(\leq k\ R.C) \notin \mathcal{L}(x)$ for all $k < n$, and
	3. x has a R-neighbour y such that $C \in \mathcal{L}(y)$
	then 1. guess m with $1 \leq m \leq n$, and
	2. $\mathcal{L}(x') = \mathcal{L}(x') \cup \{\leq mR.C, \geq mR.C\}$ for all $x' \in \varphi(x)$.

Fig. 1. Expansion rules for $\mathcal{SHIQ}$

o_φ-rule: if 1. there are nodes x, x' with $o \in \mathcal{L}(x) \cap \mathcal{L}(x')$ for some $o \in \mathbf{C}_o$,
2. $\varphi(x) \cap \varphi(x') = \emptyset$ and $\varphi(x) \neq \varphi(x')$ does not hold,
then 1. $\mathsf{Propagate}(\varphi(x), \varphi(x'), \varphi(y))$ for each y such that $\{\langle x'', y\rangle, \langle y, x''\rangle\} \cap E_o \neq \emptyset$ for $x'' \in \varphi(x) \cup \varphi(x')$, $o \in \mathbf{C}_o$.
2. $\varphi(y') = \varphi(x) \cup \varphi(x')$ for all $y' \in \varphi(x) \cup \varphi(x')$.

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

Fig. 2. New expansion rules for $\mathcal{SHOIQ}$

Input : $\sigma = \langle \lambda_\sigma, \bar{\mu}_\sigma, \bar{\nu}_\sigma \rangle \in \mathcal{N}_k$, $r \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$, $l \subseteq \mathsf{cl}(\mathcal{T}, \mathcal{R})$ and
$\mathcal{F} = \langle (\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$

Output: the frame obtained by updating $\mathcal{F}$

Build a star-type ω with $\lambda_\omega = \lambda_\sigma$, $\bar{\nu}_\omega = \bar{\nu}_\omega$ and $\bar{\mu}_\omega = (\bar{\mu}_\sigma, \langle r, l\rangle)$;
Build a star-type ω' with $\lambda_\omega = l$, $\bar{\nu}_\omega = \{\langle r^-, \lambda_\sigma\rangle\}$ and $\bar{\mu}_\omega = \emptyset$;
$\mathcal{N}_k := \mathcal{N}_k \cup \{\omega\}$;
$\delta(\omega) := \delta(\sigma)$;
$\delta(\sigma) := 0$;
if $\omega' \in \mathcal{N}_{k+1}$ **then**
    $\delta(\omega') := \delta(\omega') + \delta(\sigma)$;
else
    $\mathcal{N}_{k+1} := \mathcal{N}_{k+1} \cup \{\omega'\}$;
    $\delta(\omega') := \delta(\sigma)$;
if $\omega \not\approx \sigma'$ *for all* $\sigma' \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$ **then**
    $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma)) := \widehat{\delta}(\Phi(\sigma)) - 1$;
else
    if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
        $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;
if $\omega' \not\approx \sigma'$ *for all* $\sigma' \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$ **then**
    $\Phi(\omega') := \{\omega'\}$, $\widehat{\delta}(\Phi(\omega')) := 1$;
else
    if $o \notin \lambda_{\omega'}$ *for all* $o \in \mathbf{C}_o$ **then**
        $\widehat{\delta}(\Phi(\omega')) := \widehat{\delta}(\Phi(\omega')) + 1$;

Algorithm 1: $\mathsf{addRay}(\sigma, r, l)$ updates frame when adding a new ray $\langle r, l\rangle$ to a star-type $\sigma \in \mathcal{N}_k$.

$\sqsubseteq$-rule: if $C \sqsubseteq D \in \mathcal{T}$ and $\mathsf{nnf}(\neg C \sqcup D) \notin \lambda_\sigma$
then $\mathsf{updateLabel}(\sigma, \lambda_\sigma \cup \{\mathsf{nnf}(\neg C \sqcup D)\})$.
$\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and $\{C_1, C_2\} \not\subseteq \lambda_\sigma$
then $\mathsf{updateLabel}(\sigma, \lambda_\sigma \cup \{C_1, C_2\})$.
$\sqcup$-rule: if $C_1 \sqcup C_2 \in \lambda_\sigma$ and $\{C_1, C_2\} \cap \lambda_\sigma = \emptyset$
then $\mathsf{updateLabel}(\sigma, \lambda_\sigma \cup \{C\})$ with some $C \in \{C_1, C_2\}$.
$\exists$-rule: if 1. $\exists S.C \in \lambda_\sigma$, λ_σ is not blocked, and
2. σ has no ray $\langle r, l\rangle$ with $S \in r$ and $C \in l$
then $\mathsf{addRay}(\sigma, r, l)$.
$\forall$-rule: if 1. $\forall S.C \in \lambda_\sigma$, and
2. σ has a ray $\langle r, l\rangle$ such that $S \in r$ and $C \notin l$
then if $\langle r, l\rangle \in \bar{\nu}_\sigma$ then $\mathsf{updatePredRay}(\sigma, \langle r, l\rangle, r, l \cup \{C\})$
else $\mathsf{updateSuccRay}(\sigma, \langle r, l\rangle, r, l \cup \{C\})$
$\forall_+$-rule: if 1. $\forall S.C \in \lambda_\sigma$, and
2. there is some Q with $\mathsf{Trans}(Q)$ and $Q \sqsubseteq^{*} S$, and
3. σ has a ray $\langle r, l\rangle$ such that $Q \in r$ and $\forall Q.C \notin l$
then if $\langle r, l\rangle \in \bar{\nu}_\sigma$ then $\mathsf{updatePredRay}(\sigma, \langle r, l\rangle, r, l \cup \{\forall Q.C\})$.
else $\mathsf{updateSuccRay}(\sigma, \langle r, l\rangle, r, l \cup \{\forall Q.C\})$
ch-rule: if 1. $(\leq n\ S.C) \in \lambda_\sigma$, and
2. σ has a ray $\langle r, l\rangle$ such that $S \in r$ and $\{C, \dot{\neg} C\} \cap l = \emptyset$
then if $\langle r, l\rangle \in \bar{\nu}_\sigma$ then $\mathsf{updatePredRay}(\sigma, \langle r, l\rangle, r, l \cup \{E\})$.
else $\mathsf{updateSuccRay}(\sigma, \langle r, l\rangle, r, l \cup \{E\})$ with some $E \in \{C, \dot{\neg} C\}$.
$\geq$-rule: if 1. $(\geq n\ S.C) \in \lambda_\sigma$, σ is not blocked, and
2. σ has no n rays $\langle r_1, l_1\rangle, ..., \langle r_n, l_n\rangle$ such that $R \in r_i$, $C \in l_i$, and
$\langle r_i, l_j\rangle \neq \langle r_j, l_j\rangle$ for $1 \leq i < j \leq n$,
then call $\mathsf{addRay}(\sigma, \{R\}, \{C\})$ n times to create n rays $\langle r_1, l_1\rangle, ..., \langle r_n, l_n\rangle$
with $r_i = \{R\}$ and $l_i = \{C\}$ for $1 \leq i \leq n$, and
$\langle r_i, l_i\rangle \neq \langle r_j, l_j\rangle$ for $1 \leq i < j \leq n$.
$\leq$-rule: if 1. $(\leq n\ S.C) \in \sigma$, and
2. σ has $(n+1)$ rays $\langle r_0, l_0\rangle, ..., \langle r_n, l_n\rangle$ such that $R \in r_i, C \in l_i$ for all
$0 \leq i \leq n$ and there are $0 \leq i < j \leq n$
such that $\langle r_i, l_i\rangle \neq \langle r_j, l_j\rangle$ does not hold
then 1. For each $\langle r, l\rangle \in \{\langle r_i, l_i\rangle, \langle r_j, l_j\rangle\}$, if $\langle r, l\rangle \in \bar{\nu}_\sigma$,
then, $\mathsf{updatePredRay}(\omega, \langle r, l\rangle, r \cup r', l \cup l')$,
else, $\mathsf{updateSuccRay}(\omega, \langle r, l\rangle, r \cup r', l \cup l')$
where $\langle r', l'\rangle \in \{\langle r_i, l_i\rangle, \langle r_j, l_j\rangle\}$ with $\langle r', l'\rangle \neq \langle r, l\rangle$.
2. add $\langle r', l'\rangle \neq \langle r_i, l_i\rangle$ for all ray $\langle r', l'\rangle$ such that $\langle r', l'\rangle \neq \langle r_j, l_j\rangle$.
o-rule: if 1. there are star-types $\sigma_1, \cdots, \sigma_k$ such that $o \in \lambda_{\sigma_i}$ for some $o \in \mathbf{C}_o$
then $\mathsf{updateLabel}(\sigma_1, \cdots, \sigma_k)$,
$\leq_o$-rule: if 1. there are star-types $\sigma_1, \cdots, \sigma_k \in \Phi(\sigma)$ and $(\leq mR.C)\} \in \lambda_{\sigma_i}$ for all
$1 \leq i \leq k$, and $\sigma_1, \cdots, \sigma_k$ have $(m+1)$ distinct primary rays
$\langle r_0, l_0\rangle, \cdots \langle r_m, l_m\rangle$ such that $R \in r_i$ and $C \in l_i$ for all $0 \leq i \leq m$
then 1. Choose two rays $\langle r_j, l_j\rangle, \langle r_{j'}, l_{j'}\rangle$ of respective $\sigma_i \in \mathcal{N}_h$ and $\sigma_{i'} \in \mathcal{N}_{h'}$
with $0 \leq j < j' \leq m$ and $1 \leq i < i' \leq k$ such that $\langle r_j, l_j\rangle \neq \langle r_{j'}, l_{j'}\rangle$
2. For each $\langle r, l\rangle \in \{\langle r_j, l_j\rangle, \langle r_{j'}, l_{j'}\rangle\}$, if $\langle r, l\rangle \in \bar{\nu}_\omega$ with $\omega \in \{\sigma_i, \sigma_{i'}\}$,
then, $\mathsf{updatePredRay}(\omega, \langle r, l\rangle, r \cup r', l \cup l')$,
else, $\mathsf{updateSuccRay}(\omega, \langle r, l\rangle, r \cup r', l \cup l')$
where $\langle r', l'\rangle \in \{\langle r_j, l_j\rangle, \langle r_{j'}, l_{j'}\rangle\}$ with $\langle r', l'\rangle \neq \langle r, l\rangle$.
$\bowtie$-rule: if 1. $(\leq nR.C) \in \lambda_\sigma$, $\{(\leq l\ R.C), (\geq l\ R.C)\} \not\subseteq \lambda_\sigma$ for all $l \leq n$,
2. $(\leq k\ R.C) \notin \lambda_\sigma$ for all $k < n$, and
3. σ has a ray $\langle r, l\rangle$ such that $R \in r, C \in l$
then 1. guess m with $1 \leq m \leq n$, and
2. $\mathsf{updateLabel}(\sigma, \lambda_\sigma \cup \{\leq mR.C, \geq mR.C\})$.

Fig. 3. Expansion rules for constructing a frame.

Input : $\sigma = \langle \lambda_\sigma, \bar{\mu}_\sigma, \bar{\nu}_\sigma \rangle \in \mathcal{N}_k$; $l \subseteq \mathsf{cl}(\mathcal{T}, \mathcal{R})$ and $\mathcal{F} = \langle (\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$
Output: the frame obtained by updating $\mathcal{F}$

Let $\bar{\nu}(\sigma) = \{\langle r, l \rangle\}$;
Let $\mathcal{N}(\sigma) = \mathcal{N}'(\sigma) \cup \{\omega_0\}$;
foreach $\sigma' \in \mathcal{N}(\sigma)$ **do**
  Build a star-type ω with $\lambda_\omega = \lambda_{\sigma'}$, $\bar{\nu}_\omega = \bar{\nu}_{\sigma'}$ and $\bar{\mu}_\omega := (\bar{\mu}_{\sigma'}, \langle r^-, l_0 \rangle)$;
  $\mathcal{N}_{k-1} := \mathcal{N}_{k-1} \cup \{\omega\}$;
  if $\sigma' = \omega_0$ **then**
    $\delta(\omega) := \delta(\sigma) - \sum_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega')$;
    $\delta(\sigma') := \delta(\sigma') - (\delta(\sigma) - \sum_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega'))$;
  else
    $\delta(\omega) := \delta(\sigma'), \delta(\sigma') := 0$;
  if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
    $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma')) := \widehat{\delta}(\Phi(\sigma')) - 1$;
  else
    if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
      $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;

Let $\bar{\mu}_\sigma = (\langle r_1, l_1 \rangle, \cdots, \langle r_i, l_j \rangle, \cdots, \langle r_{k'}, l_{k'} \rangle)$;
foreach $\langle r_i, l_i \rangle$ *with* $1 \leq i \leq k'$ **do**
  Let $\widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle) = \widehat{\mathcal{N}}'(\sigma, \langle r_i, l_i \rangle) \cup \{\bar{\omega}_0\}$;
  foreach $\sigma' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)$ **do**
    Build a star-type ω with $\lambda_\omega = l_i$, $\bar{\nu}_\omega = \{\langle r_i^-, l_0 \rangle\}$ and $\bar{\mu}_\omega = \bar{\mu}_{\sigma'}$;
    $\mathcal{N}_{k+1} := \mathcal{N}_{k+1} \cup \{\omega\}$;
    if $\sigma' = \bar{\omega}_0$ **then**
      $\delta(\omega) := \delta(\sigma') - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)} \delta(\omega')$;
      $\delta(\sigma') := \delta(\sigma') - (\delta(\sigma) - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)} \delta(\omega'))$;
    else
      $\delta(\omega) := \delta(\sigma'), \delta(\sigma') := 0$;
    if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
      $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma')) := \widehat{\delta}(\Phi(\sigma')) - 1$;
    else
      if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
        $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;

Build a star-type ω with $\lambda_\omega = l_0$, $\bar{\nu}_\omega = \bar{\nu}_\sigma$ and $\bar{\mu}_\omega = \bar{\mu}_\sigma$;
$\delta(\omega) := \delta(\sigma), \delta(\sigma) := 0, \mathcal{N}_k := \mathcal{N}_k \cup \{\omega\}$;
if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
  $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma')) := \widehat{\delta}(\Phi(\sigma')) - 1$;
else
  if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
    $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;

Algorithm 2: $\mathsf{updateLabel}(\sigma, l_0)$ updates frame when modifying λ_σ by assigning l to λ_σ.

Input : $\sigma = \langle \lambda_\sigma, \bar{\mu}_\sigma, \bar{\nu}_\sigma \rangle \in \mathcal{N}_k$; $\bar{\nu}_\sigma = \{\langle r, l \rangle\}$ $r_0 \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$; $l_0 \subseteq \mathsf{cl}(\mathcal{T},\mathcal{R})$ and $\mathcal{F} = \langle (\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$

Output: the frame obtained by updating $\mathcal{F}$

Let $\mathcal{N}(\sigma) = \mathcal{N}'(\sigma) \cup \{\omega_0\}$;
foreach $\sigma' \in \mathcal{N}(\sigma)$ **do**
  Let $\bar{\nu}(\sigma') = \{\langle s, h \rangle\}$;
  Let $\bar{\mu}(\sigma') = (\langle s_1, h_1 \rangle, \cdots, \langle s_i, h_i \rangle, \cdots, \langle s_n, h_n \rangle)$ with $s_i = r^-$ and $h_i = \lambda_\sigma$;
  Let $\mathcal{N}(\sigma') = \mathcal{N}'(\sigma') \cup \{\omega'_0\}$;
  foreach $\sigma'' \in \mathcal{N}(\sigma')$ **do**
    Build a star-type ω with $\lambda_\omega = \lambda_{\sigma''}$, $\bar{\nu}_\omega = \bar{\nu}_{\sigma''}$ and $\bar{\mu}_\omega = (\bar{\mu}_{\sigma''}, \langle s^-, \lambda_{\sigma'} \rangle)$;
    $\mathcal{N}_{k-2} := \mathcal{N}_{k-2} \cup \{\omega\}$;
    if $\sigma'' = \omega'_0$ **then**
      $\delta(\omega) := \delta(\sigma') - \sum_{\omega' \in \mathcal{N}'(\sigma')} \delta(\omega')$;
      $\delta(\sigma'') := \delta(\sigma'') - (\delta(\sigma') - \sum_{\omega' \in \mathcal{N}'(\sigma')} \delta(\omega'))$;
    else
      $\delta(\omega) := \delta(\sigma''), \delta(\sigma'') := 0$;
    if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$ **then**
      $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma'')) := \widehat{\delta}(\Phi(\sigma'')) - 1$;
    else
      if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
        $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;
  Build a star-type ω with $\lambda_\omega = l_0$, $\bar{\nu}_\omega = \bar{\nu}_{\sigma'}$ and $\bar{\mu}_\omega = (\bar{\mu}_{\sigma'}, \langle r_0^-, l_0 \rangle)$;
  $\mathcal{N}_{k-1} := \mathcal{N}_{k-1} \cup \{\omega\}$;
  if $\sigma' = \omega_0$ **then**
    $\delta(\omega) := \delta(\sigma) - \sum_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega')$;
    $\delta(\sigma') := \delta(\sigma') - (\delta(\sigma) - \sum_{\omega' \in \mathcal{N}'(\sigma)} \delta(\omega'))$;
  else
    $\delta(\omega) := \delta(\sigma'), \delta(\sigma') := 0$;
  if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$ **then**
    $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma')) := \widehat{\delta}(\Phi(\sigma')) - 1$;
  else
    if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
      $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;
Build a star-type ω with $\lambda_\omega = \lambda_\sigma$, $\bar{\nu}_{\sigma_0} = \{\langle r_0, l_0 \rangle\}$ and $\bar{\mu}_\omega = \bar{\mu}_\sigma$;
$\delta(\omega) := \delta(\sigma), \delta(\sigma) := 0, \mathcal{N}_k := \mathcal{N}_k \cup \{\omega\}$;
if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1,\cdots,H\}} \mathcal{N}_i$ **then**
  $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma)) := \widehat{\delta}(\Phi(\sigma)) - 1$;
else
  if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
    $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;

Algorithm 3: $\mathsf{updatePredRay}(\sigma, \langle r, l \rangle, r_0, l_0)$ updates frame when modifying a ray $\langle r, l \rangle$ of a star-type $\sigma \in \mathcal{N}_k$ by assigning r_0, l_0 to respective r and l.

Input : $\sigma = \langle \lambda_\sigma, \bar{\mu}_\sigma, \bar{\nu}_\sigma \rangle \in \mathcal{N}_k$; $\langle r, l \rangle \notin \bar{\nu}_\sigma$ is ray of σ; $r_0 \subseteq \mathbf{R}_{(\mathcal{T},\mathcal{R})}$; $l_0 \subseteq \mathsf{cl}(\mathcal{T}, \mathcal{R})$ and $\mathcal{F} = \langle (\mathcal{N}_0, \cdots, \mathcal{N}_H), \delta, \Phi, \widehat{\delta} \rangle$

Output: the frame obtained by updating $\mathcal{F}$

Let $\bar{\mu}_\sigma = (\langle r_1, l_1 \rangle, \cdots, \langle r_i, l_i \rangle, \cdots, \langle r_k, l_k \rangle)$;
foreach $\langle r_i, l_i \rangle$ *with* $1 \leq i \leq k$ **do**
  Let $\widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle) = \widehat{\mathcal{N}}'(\sigma, \langle r_i, l_i \rangle) \cup \{\omega_0\}$;
  foreach $\sigma' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)$ **do**
    Let $\bar{\mu}_{\sigma'} = (\langle s_1, h_1 \rangle, \cdots, \langle s_j, h_j \rangle, \cdots, \langle s_{k'}, h_{k'} \rangle)$;
    foreach $\langle s_j, h_j \rangle$ *with* $1 \leq j \leq k'$ **do**
      Let $\widehat{\mathcal{N}}(\sigma', \langle s_j, h_j \rangle) = \widehat{\mathcal{N}}'(\sigma', \langle s_j, h_j \rangle) \cup \{\omega_1\}$;
      foreach $\sigma'' \in \widehat{\mathcal{N}}(\sigma', \langle s_j, h_j \rangle)$ **do**
        Build a star-type ω with $\lambda_\omega = h_j$, $\bar{\nu}_\omega = \{\langle s_j^-, \lambda_{\sigma'} \rangle\}$ and $\bar{\mu}_\omega = \bar{\mu}_{\sigma''}$;
        $\mathcal{N}_{k+2} := \mathcal{N}_{k+2} \cup \{\omega\}$;
        if $\sigma'' = \omega_1$ **then**
          $\delta(\omega) := \delta(\sigma') - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma', \langle s_j, h_j \rangle)} \delta(\omega')$;
          $\delta(\sigma'') := \delta(\sigma'') - (\delta(\sigma') - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma', \langle s_j, h_j \rangle)} \delta(\omega'))$;
        else
          $\delta(\omega) := \delta(\sigma'')$, $\delta(\sigma'') := 0$;
        if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
          $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma'')) := \widehat{\delta}(\Phi(\sigma'')) - 1$;
        else
          if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
            $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;
    Build a star-type ω with $\lambda_\omega = l_i$, $\bar{\nu}_\omega = \{\langle r_i^-, \lambda_\sigma \rangle\}$ and $\bar{\mu}_\omega = \bar{\mu}_{\sigma'}$;
    $\mathcal{N}_{k+1} := \mathcal{N}_{k+1} \cup \{\omega\}$;
    if $\sigma' = \omega_0$ **then**
      $\delta(\omega) := \delta(\sigma) - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)} \delta(\omega')$;
      $\delta(\sigma') := \delta(\sigma') - (\delta(\sigma) - \sum_{\omega' \in \widehat{\mathcal{N}}(\sigma, \langle r_i, l_i \rangle)} \delta(\omega'))$;
    else
      $\delta(\omega) := \delta(\sigma')$, $\delta(\sigma') := 0$;
    if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
      $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma')) := \widehat{\delta}(\Phi(\sigma')) - 1$;
    else
      if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
        $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;
Build a star-type ω with $\lambda_\omega = l_0$, $\bar{\nu}_\omega = \bar{\nu}_\sigma$ and $\bar{\mu}_\omega = (\bar{\mu}_\sigma, \langle r_0, l_0 \rangle)$;
$\delta(\omega) := \delta(\sigma)$, $\delta(\sigma) := 0$, $\mathcal{N}_k := \mathcal{N}_k \cup \{\omega\}$;
if $\omega \not\approx \omega'$ *for all* $\omega' \in \bigcup_{i \in \{1, \cdots, H\}} \mathcal{N}_i$ **then**
  $\Phi(\omega) := \{\omega\}$, $\widehat{\delta}(\Phi(\omega)) := 1$ and $\widehat{\delta}(\Phi(\sigma)) := \widehat{\delta}(\Phi(\sigma)) - 1$;
else
  if $o \notin \lambda_\omega$ *for all* $o \in \mathbf{C}_o$ **then**
    $\widehat{\delta}(\Phi(\omega)) := \widehat{\delta}(\Phi(\omega)) + 1$;

Algorithm 4: $\mathsf{updateSuccRay}(\sigma, \langle r, l \rangle, r_0, l_0)$ updates frame when modifying a ray $\langle r, l \rangle \notin \bar{\nu}_\sigma$ of a star-type $\sigma \in \mathcal{N}_k$ by assigning r_0, l_0 to respective r and l.