A Tableaux Decision Procedure for $\mathcal{SHOIQ}$

Ian Horrocks and Ulrike Sattler
School of Computer Science, University of Manchester, UK
horrocks|sattler@cs.man.ac.uk

Abstract

This paper presents a tableaux decision procedure for $\mathcal{SHOIQ}$, the DL underlying OWL DL. To the best of our knowledge, this is the first goal-directed decision procedure for $\mathcal{SHOIQ}$.

1 Introduction

Description Logics (DLs) are a family of logic based knowledge representation formalisms. Although they have a range of applications (e.g., configuration [McGuinness & Wright, 1998], and information integration [Calvanese *et al.*, 1998]), they are perhaps best known as the basis for widely used ontology languages such as OIL, DAML+OIL and OWL [Horrocks *et al.*, 2003], the last of which is now a W3C recommendation [Bechhofer *et al.*, 2004]. Two of the three language "species" described in the OWL specification are based on expressive DLs: OWL Lite is based on $\mathcal{SHIF}$, and OWL DL is based on $\mathcal{SHOIN}$. This decision was motivated by a requirement that key inference problems (such as ontology consistency) be decidable, and that it should be possible to provide reasoning services to support ontology design and deployment [Horrocks *et al.*, 2003].

Although the ontology consistency problem for $\mathcal{SHOIN}$ is known to be decidable,[1] to the best of our knowledge no "practical" decision procedure is known for it, i.e., no goal directed procedure that is likely to perform well with realistic ontology derived problems [Tobies, 2001; Horrocks & Sattler, 2001]. In this paper we present such a decision procedure for $\mathcal{SHOIQ}$, i.e., $\mathcal{SHOIN}$ extended with *qualified* number restrictions [Baader & Hollunder, 1991]. The algorithm extends the well-known tableaux algorithm for $\mathcal{SHIQ}$ [Horrocks *et al.*, 1999], which is the basis for several highly successful implementations [Horrocks & Patel-Schneider, 1998; Haarslev & Möller, 2001; Pellet, 2003].

The $\mathcal{O}$ in $\mathcal{SHOIQ}$ denotes *nominals*, i.e., classes with a singleton extension. Nominals are a prominent feature of hybrid logics [Blackburn & Seligman, 1995], and can also be viewed as a powerful generalisation of *ABox individuals* [Schaerf, 1994; Horrocks & Sattler, 2001]. They occur naturally in ontologies, e.g., when describing a class such as EUCountries by enumerating its members, i.e., {Austria, . . . , UnitedKingdom} (such an enumeration is equivalent to a disjunction of nominals). This allows applications to infer, e.g., that persons who only visit EUCountries can visit at most 15 countries.

One reason why DLs (and propositional modal and dynamic logics) enjoy good computational properties, such as being robustly decidable, is that they have some form of tree model property [Vardi, 1997], i.e., if an ontology is consistent, then it has a (form of) tree model. This feature is crucial in the design of tableaux algorithms, allowing them to search only for tree like models. More precisely, tableaux algorithms decide consistency of an ontology by trying to construct an abstraction of a model for it, a so-called "completion graph". For logics with the tree model property, we can restrict our search/construction to tree-shaped completion graphs.

Tableaux algorithms for expressive DLs employ a cycle detection technique called *blocking* to ensure termination. This is of special interest for $\mathcal{SHIQ}$, where the interaction between inverse roles and number restrictions results in the loss of the *finite model property*, i.e., there are consistent ontologies that only admit infinite models. On such an input, the $\mathcal{SHIQ}$ tableaux algorithm generates a finite tree-shaped completion graph that can be *unravelled* into an infinite tree model, and where a node in the completion graph may stand for infinitely many elements of the model. Even when the language includes nominals, but *excludes* one of number restrictions or inverse roles [Horrocks & Sattler, 2001; Hladik & Model, 2004], or if nominals are restricted to ABox individuals [Horrocks *et al.*, 2000], we can work on forest-shaped completion graphs, with each nominal (individual) being the root of a tree-like section; this causes no inherent difficulty as the size of the non-tree part of the graph is restricted by the number of individuals/nominals in the input.

The difficulty in extending the $\mathcal{SHOQ}$ or $\mathcal{SHIQ}$ algorithms to $\mathcal{SHOIQ}$ is due to the interaction between nominals, number restrictions, and inverse roles, which leads to the *almost* complete loss of the tree model property, and causes the complexity of the ontology consistency problem to jump from ExpTime to NExpTime [Tobies, 2000]. To see this, consider an ontology containing the following two axioms that

[1] This is an immediate consequence of a reduction of DLs with transitive roles to DLs without such roles [Tobies, 2001] and the fact that applying this reduction to $\mathcal{SHOIN}$ yields a fragment of the two variable fragment of first order logic with counting quantifiers [Pacholski *et al.*, 1997].

use a nominal o to impose an upper bound, say 17, on the number of instances of the concept F:

$$\top \dot{\sqsubseteq} \exists R^-.o, \qquad o \dot{\sqsubseteq} (\leqslant 17\, R.F)$$

The first statement requires that, in a model of this ontology, every element has an incoming R-edge from o; the second statement restricts the number of R-edges going from o to instances of F to at most 17. In this case, we might need to consider arbitrarily complex relational structures amongst instances of F, and thus cannot restrict our attention to completion trees or forests. Let us assume that our ontology also forces the existence of an infinite number of instances of another concept, say N, which requires the above mentioned "block and unravel" technique. The consistency of the whole ontology then crucially depends on the relations enforced between instances of F and N, and whether the unravelling of the N-part violates atmost number restrictions that instances of F must satisfy. Summing up, a tableaux algorithm for $\mathcal{SHOIQ}$ needs to be able to handle both arbitrarily complex relational structures and finite tree structures representing infinite trees, and to make sure that all constraints are satisfied (especially number restrictions on relations between these two parts), while still guaranteeing termination.

Two key intuitions have allowed us to devise a tableaux algorithm that meets all of these requirements. The first intuition is that, when extending a $\mathcal{SHOIQ}$ completion graph, we can distinguish those nodes that may be arbitrarily interconnected (so-called *nominal nodes*) from those nodes that still form a tree structure (*so-called blockable nodes*). Fixing a (double exponential) upper bound on the number of nominal nodes is crucial to proving termination; it is not, however, enough to guarantee termination, as we may repeatedly create and merge nominal nodes (a so-called "yo-yo").

The second intuition is that the yo-yo problem can be overcome by "guessing" the *exact* number of new nominal nodes resulting from interactions between existing nominal nodes, inverse roles, and number restrictions. This guessing is implemented by a new expansion rule, $\mathsf{Ro?}$, which, when applied to a relevant $(\leqslant nR.C)$ concept, generates (nondeterministically) between 1 and n new nominal nodes, all of which are pairwise disjoint. This prevents the repeated yo-yo construction, and termination is now guaranteed by the upper bound on the number of nominal nodes and the use of standard blocking techniques for the blockable nodes. The nondeterminism introduced by this rule could clearly be problematical for large values of n, but large values in number restrictions are already known to be problematical for $\mathcal{SHIQ}$. Moreover, the rule has excellent "pay as you go" characteristics: in case number restrictions are functional (i.e., where n is 1),[2] the new rule becomes deterministic; in case there are no interactions between number restrictions, inverse roles and nominals, the rule will never be applied; in case there are no nominals, the new algorithm will behave like the algorithm for $\mathcal{SHIQ}$; and in case there are no inverse roles, the new algorithm will behave like the algorithm for $\mathcal{SHOQ}$.

For more examples and full proofs, we refer the reader to [Horrocks & Sattler, 2005].

[2] A feature of many realistic ontologies; see, e.g., the DAML ontology library at `http://www.daml.org/ontologies/`

2 Preliminaries

In this section, we introduce the syntax, semantics, and inference problems of the DL $\mathcal{SHOIQ}$.

Definition 1 Let $\mathbf{R}$ be a set of *role names* with both transitive and normal role names $\mathbf{R}_+ \cup \mathbf{R}_\mathsf{P} = \mathbf{R}$, where $\mathbf{R}_\mathsf{P} \cap \mathbf{R}_+ = \emptyset$. The set of *$\mathcal{SHOIQ}$-roles* (or *roles* for short) is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. A *role inclusion axiom* is of the form $R \sqsubseteq S$, for two roles R and S. A *role hierarchy* is a finite set of role inclusion axioms.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ which maps every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for $P \in \mathbf{R}$ and $R \in \mathbf{R}_+$,

$$\langle x, y\rangle \in P^\mathcal{I} \text{ iff } \langle y, x\rangle \in P^{-\mathcal{I}},$$
$$\text{and if } \langle x, y\rangle \in R^\mathcal{I} \text{ and } \langle y, z\rangle \in R^\mathcal{I}, \text{ then } \langle x, z\rangle \in R^\mathcal{I}.$$

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 $\mathcal{R}$.

We introduce some notation to make the following considerations easier.

1. The inverse relation on roles is symmetric, and to avoid considering roles such as R^{--}, we define a function Inv which returns the inverse of a role: for $R \in \mathbf{R}$, $\mathsf{Inv}(R) := R^-$ and $\mathsf{Inv}(R^-) := R$.
2. Since set inclusion is transitive and $R^\mathcal{I} \subseteq S^\mathcal{I}$ implies $\mathsf{Inv}(R)^\mathcal{I} \subseteq \mathsf{Inv}(S)^\mathcal{I}$, for a role hierarchy $\mathcal{R}$, we introduce $\sqsubseteq^*_\mathcal{R}$ as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. We use $R \equiv_\mathcal{R} S$ as an abbreviation for $R \sqsubseteq^*_\mathcal{R} S$ and $S \sqsubseteq^*_\mathcal{R} R$.
3. Obviously, a role R is transitive if and only if its inverse $\mathsf{Inv}(R)$ is transitive. However, in cyclic cases such as $R \equiv_\mathcal{R} S$, S is transitive if R or $\mathsf{Inv}(R)$ is a transitive role name. In order to avoid these case distinctions, the function Trans returns true if R is a transitive role—regardless whether it is a role name, the inverse of a role name, or equivalent to a transitive role name (or its inverse): $\mathsf{Trans}(S, \mathcal{R}) := \mathrm{true}$ if, for some P with $P \equiv_\mathcal{R} S$, $P \in \mathbf{R}_+$ or $\mathsf{Inv}(P) \in \mathbf{R}_+$; $\mathsf{Trans}(S, \mathcal{R}) := \mathrm{false}$ otherwise.
4. A role R is called *simple* w.r.t. $\mathcal{R}$ if $\mathsf{Trans}(S, \mathcal{R}) = \mathrm{false}$ for all $S \sqsubseteq^*_\mathcal{R} R$.
5. In the following, if $\mathcal{R}$ is clear from the context, we may use $\sqsubseteq^*$ and $\mathsf{Trans}(S)$ instead of $\sqsubseteq^*_\mathcal{R}$ and $\mathsf{Trans}(S, \mathcal{R})$.

Definition 2 Let N_C be a set of *concept names* with a subset $N_I \subseteq N_C$ of *nominals*. The set of *$\mathcal{SHOIQ}$-concepts* (or *concepts* for short) is the smallest set such that

1. every concept name $C \in N_C$ is a concept,
2. if C and D are concepts and R is a role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are also concepts (the last two are called universal and existential restrictions, resp.), and
3. if C is a concept, R is a simple role[3] and $n \in \mathbb{N}$, then $(\leqslant nR.C)$ and $(\geqslant nR.C)$ are also concepts (called atmost and atleast number restrictions).

[3] Restricting number restrictions to simple roles is required in order to yield a decidable logic [Horrocks *et al.*, 1999].

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$\begin{array}{rl} (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, & (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\ \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, & \sharp o^{\mathcal{I}} = 1 \text{ for all } o \in N_I, \\ (\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, C) \neq \emptyset\}, & \\ (\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, \neg C) = \emptyset\}, & \\ (\leqslant n R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \leqslant n\}, \text{ and} & \\ (\geqslant n R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp R^{\mathcal{I}}(x, C) \geqslant n\}, & \end{array}$$

where $\sharp M$ is the cardinality of a set M and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$.

For C and D (possibly complex) concepts, $C \dot{\sqsubseteq} D$ is called a *general concept inclusion* (GCI), and a finite set of GCIs is called a *TBox*.

An interpretation $\mathcal{I}$ *satisfies* a GCI $C \dot{\sqsubseteq} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies each GCI in $\mathcal{T}$; such an interpretation is called a *model of* $\mathcal{T}$.

A concept C is *satisfiable w.r.t. a role hierarchy $\mathcal{R}$ and a TBox $\mathcal{T}$* if there is a model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a *model of* C w.r.t. $\mathcal{R}$ and $\mathcal{T}$. A concept D *subsumes* a concept C w.r.t. $\mathcal{R}$ and $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{R},\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$. Two concepts C, D are *equivalent* w.r.t. $\mathcal{R}$ and $\mathcal{T}$ (written $C \equiv_{\mathcal{R},\mathcal{T}} D$) if they are mutually subsuming w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

As usual, subsumption and satisfiability can be reduced to each other. Like $\mathcal{SHIQ}$, in $\mathcal{SHOIQ}$, we can reduce reasoning w.r.t. TBoxes and role hierarchies to reasoning w.r.t. role hierarchies only: we can use an "approximation" of a universal role to *internalise* a TBox [Horrocks *et al.*, 1999; Horrocks & Sattler, 2005]. Hence, in the remainder of this paper, we restrict our attention without loss of generality to the satisfiability of $\mathcal{SHOIQ}$ concepts w.r.t. a role hierarchy.

Finally, we did not choose to make a *unique name assumption*, i.e., two nominals might refer to the same individual. However, the inference algorithm presented below can easily be adapted to the unique name case by a suitable initialisation of the inequality relation $\neq$ used in our algorithm.

3 A Tableau for $\mathcal{SHOIQ}$

For ease of presentation, as usual, we assume all concepts to be in *negation normal form* (NNF). Each concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of de Morgan's laws and the duality between existential and universal restrictions, and between atmost and atleast number restrictions, [Horrocks *et al.*, 2000]. For a concept C, we use $\doteq C$ to denote the NNF of $\neg C$, and we use $\mathsf{sub}(C)$ to denote the set of all subconcepts of C (including C). As usual, for a concept D and a role hierarchy $\mathcal{R}$, we define the set of "relevant sub-concepts" $\mathsf{cl}(D, R)$ as follows (we will often use $\mathsf{cl}(D)$ instead of $\mathsf{cl}(D, \mathcal{R})$):

$$\begin{array}{rl} \mathsf{cl}(D, \mathcal{R}) := & \mathsf{sub}(D) \cup \{\doteq C \mid C \in \mathsf{sub}(D)\} \cup \\ & \{\forall S.E \mid \{\forall R.E, \doteq \forall R.E\} \cap \mathsf{sub}(D) \neq \emptyset \\ & \qquad \text{and } S \text{ occurs in } \mathcal{R} \text{ or } D\}. \end{array}$$

Definition 3 If D is a $\mathcal{SHOIQ}$-concept in NNF, $\mathcal{R}$ a role hierarchy, and $\mathbf{R}_D$ is the set of roles occurring in D or $\mathcal{R}$, together with their inverses, a tableau T for D w.r.t. $\mathcal{R}$ is defined to be a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \to 2^{\mathsf{cl}(D)}$ maps each individual to a set of concepts which is a subset of $\mathsf{cl}(D)$, $\mathcal{E} : \mathbf{R}_D \to 2^{\mathbf{S} \times \mathbf{S}}$ maps each role in $\mathbf{R}_D$ to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s, t \in \mathbf{S}$, $C, C_1, C_2 \in \mathsf{cl}(D)$, $R, S \in \mathbf{R}_D$, and

$$S^T(s, C) := \{t \in \mathbf{S} \mid \langle s, t\rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\},$$

it holds that:

(P1) if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,

(P2) if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,

(P3) if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,

(P4) if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$,

(P5) if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s, t\rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$,

(P6) if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^* S$ with $\mathsf{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,

(P7) if $(\geqslant n S.C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \geqslant n$,

(P8) if $(\leqslant n S.C) \in \mathcal{L}(s)$, then $\sharp S^T(s, C) \leqslant n$,

(P9) if $(\leqslant n S.C) \in \mathcal{L}(s)$ and $\langle s, t\rangle \in \mathcal{E}(S)$, then $\{C, \doteq C\} \cap \mathcal{L}(t) \neq \emptyset$,

(P10) if $\langle s, t\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* S$, then $\langle s, t\rangle \in \mathcal{E}(S)$,

(P11) $\langle s, t\rangle \in \mathcal{E}(R)$ iff $\langle t, s\rangle \in \mathcal{E}(\mathsf{Inv}(R))$, and

(P12) if $o \in \mathcal{L}(s) \cap \mathcal{L}(t)$ for some $o \in N_I$, then $s = t$.

Lemma 4 A $\mathcal{SHOIQ}$-concept D in NNF is satisfiable w.r.t. a role hierarchy $\mathcal{R}$ iff D has a tableau w.r.t. $\mathcal{R}$.

The proof is similar to the one found in [Horrocks *et al.*, 1999]. Roughly speaking, we construct a model $\mathcal{I}$ from a tableau by taking $\mathbf{S}$ as its interpretation domain and adding the missing role-successorships for transitive roles. Then, by induction on the structure of formulae, we prove that, if $C \in \mathcal{L}(s)$, then $s \in C^{\mathcal{I}}$. Most cases are straightforward. As for $\mathcal{SHIQ}$, (P7) and (P8) together with the fact that only simple roles occur in number restrictions ensure that this implication holds for number restrictions C, and (P12) ensures that nominals are indeed interpreted as singletons. For the converse, we can easily transform any model into a tableau.

4 A tableau algorithm for $\mathcal{SHOIQ}$

From Lemma 4, an algorithm which constructs a tableau for a $\mathcal{SHOIQ}$-concept D can be used as a decision procedure for the satisfiability of D with respect to a role hierarchy $\mathcal{R}$. Such an algorithm will now be described in detail.

Definition 5 Let $\mathcal{R}$ be a role hierarchy and D a $\mathcal{SHOIQ}$-concept in NNF. A *completion graph* for D with respect to $\mathcal{R}$ is a directed graph $\mathbf{G} = (V, E, \mathcal{L}, \not\approx)$ where each node $x \in V$ is labelled with a set $\mathcal{L}(x) \subseteq \mathsf{cl}(D) \cup N_I \cup \{(\leqslant m R.C) \mid (\leqslant n R.C) \in \mathsf{cl}(D) \text{ and } m \leq n\}$ and each edge $\langle x, y\rangle \in E$ is labelled with a set of role names $\mathcal{L}(\langle x, y\rangle)$ containing (possibly inverse) roles occurring in D or $\mathcal{R}$. Additionally, we keep track of inequalities between nodes of the graph with a symmetric binary relation $\not\approx$ between the nodes of $\mathbf{G}$.

If $\langle x, y\rangle \in E$, then y is called a *successor* of x and x is called a *predecessor* of y. *Ancestor* is the transitive closure of predecessor, and *descendant* is the transitive closure of successor. A node y is called an R-*successor* of a node x if, for some R' with $R' \sqsubseteq^* R$, $R' \in \mathcal{L}(\langle x, y\rangle)$; x is called an R-*predecessor* of y if y is an R-successor of x. A node y is called a *neighbour* (R-*neighbour*) of a node x if y is a successor (R-successor) of x or if x is a successor ($\mathsf{Inv}(R)$-successor) of y.

For a role S and a node x in $\mathbf{G}$, we define the set of x's S-neighbours with C in their label, $S^{\mathbf{G}}(x, C)$, as follows:

$$S^{\mathbf{G}}(x, C) := \{y \mid y \text{ is an } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}.$$

$\mathbf{G}$ is said to contain a *clash* if

1. for some $A \in N_C$ and node x of $\mathbf{G}$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$,
2. for some node x of $\mathbf{G}$, $(\leqslant nS.C) \in \mathcal{L}(x)$ and there are $n+1$ S-neighbours $y_0, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ for each $0 \leq i \leq n$ and $y_i \neq y_j$ for each $0 \leq i < j \leq n$, *or*
3. for some $o \in N_I$, there are $x \neq y$ with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$.

If $o_1, \ldots, o_\ell$ are all the nominals occurring in D, then the tableau algorithm starts with the completion graph $\mathbf{G} = (\{r_0, r_1 \ldots, r_\ell\}, \emptyset, \mathcal{L}, \emptyset)$ with $\mathcal{L}(r_0) = \{D\}$ and $\mathcal{L}(r_i) = \{o_i\}$ for $1 \leq i \leq \ell$. $\mathbf{G}$ is then expanded by repeatedly applying the expansion rules given in Figures 1 and 2, stopping if a clash occurs.

Next, we define and explain some terms and operations used in the expansion rules:

Nominal Nodes and Blockable Nodes We distinguish two types of nodes in $\mathbf{G}$, *nominal* nodes and *blockable* nodes. A node x is a nominal node if $\mathcal{L}(x)$ contains a nominal. A node that is not a nominal node is a blockable node. A nominal $o \in N_I$ is said to be *new in* $\mathbf{G}$ if no node in $\mathbf{G}$ has o in its label.

Comment: like ABox individuals [Horrocks *et al.*, 2000], nominal nodes can be arbitrarily interconnected. In contrast, blockable nodes are only found in tree-like structures rooted in nominal nodes (or in r_0); a branch of such a tree may end with an edge leading to a nominal node.

In $\mathsf{R}O?$, we use *new* nominals to create new nominal nodes—intuitively, to fix the identity of certain, constrained neighbours of nominal nodes. An upper bound on the number of nominal nodes that can be generated in a given completion graph will be crucial for termination of the construction, given that blocking cannot be applied to nominal nodes.

Blocking A node x is *label blocked* if it has ancestors x', y and y' such that

1. x is a successor of x' and y is a successor of y',
2. y, x and all nodes on the path from y to x are blockable,
3. $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$, *and*
4. $\mathcal{L}(\langle x', x\rangle) = \mathcal{L}(\langle y', y\rangle)$.

In this case, we say that y *blocks* x. A node is *blocked* if either it is label blocked or it is blockable and its predecessor is blocked; if the predecessor of a blockable node x is blocked, then we say that x is *indirectly blocked*.

Comment: blocking is defined exactly as for $\mathcal{SHIQ}$, with the only difference that, in the presence of nominals, we must take care that none of the nodes between a blocking and a blocked one is a nominal node.

$\mathsf{R}\sqcap$: if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$

$\mathsf{R}\sqcup$: if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$\mathsf{R}\exists$: if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and
2. x has no safe S-neighbour y with $C \in \mathcal{L}(y)$,
then create a new node y with $\mathcal{L}(\langle x, y\rangle) = \{S\}$
and $\mathcal{L}(y) = \{C\}$

$\mathsf{R}\forall$: if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. there is an S-neighbour y of x with $C \notin \mathcal{L}(y)$
then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$

$\mathsf{R}\forall_+$: if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
2. there is some R with $\mathsf{Trans}(R)$ and $R \sqsubseteq^* S$,
3. there is an R-neighbour y of x with $\forall R.C \notin \mathcal{L}(y)$
then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$

$\mathsf{R}?$: if 1. $(\leqslant nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there
2. is an S-neighbour y of x with $\{C, \doteq\!C\} \cap \mathcal{L}(y) = \emptyset$
then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \doteq\!C\}$

$\mathsf{R}\geqslant$: if 1. $(\geqslant nS.C) \in \mathcal{L}(x)$, x is not blocked, and
2. there are not n safe S-neighbours $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$
then create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.

$\mathsf{R}\leqslant$: if 1. $(\leqslant nS.C) \in \mathcal{L}(z)$, z is not indirectly blocked, and
2. $\sharp S^{\mathbf{G}}(z, C) > n$ and there are two S-neighbours x, y of z with $C \in \mathcal{L}(x) \cap \mathcal{L}(y)$, and not $x \neq y$
then 1. if x is a nominal node, then $\mathsf{Merge}(y, x)$
2. else if y is a nominal node or an ancestor of x, then $\mathsf{Merge}(x, y)$
3. else $\mathsf{Merge}(y, x)$

Figure 1: The tableaux expansion rules for $\mathcal{SHIQ}$

Generating and Shrinking Rules and Safe Neighbours The rules $\mathsf{R}\geqslant$, $\mathsf{R}\exists$, and $\mathsf{R}O?$ are called *generating rules*, and the rules $\mathsf{R}\leqslant$ and $\mathsf{R}O$ are called *shrinking rules*. An R-neighbour y of a node x is *safe* if (i) x is blockable or if (ii) x is a nominal node and y is not blocked.

Pruning When a node y is *merged* into a node x, we "prune" the completion graph by removing y and, recursively, all blockable successors of y. More precisely, pruning a node y (written $\mathsf{Prune}(y)$) in $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all successors z of y, remove $\langle y, z\rangle$ from E and, if z is blockable, $\mathsf{Prune}(z)$;
2. remove y from V.

Merging merging a node y into a node x (written $\mathsf{Merge}(y, x)$) in $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all nodes z such that $\langle z, y\rangle \in E$
   (a) if $\{\langle x, z\rangle, \langle z, x\rangle\} \cap E = \emptyset$, then add $\langle z, x\rangle$ to E and set $\mathcal{L}(\langle z, x\rangle) = \mathcal{L}(\langle z, y\rangle)$,
   (b) if $\langle z, x\rangle \in E$, then set $\mathcal{L}(\langle z, x\rangle) = \mathcal{L}(\langle z, x\rangle) \cup \mathcal{L}(\langle z, y\rangle)$,

(c) if $\langle x, z\rangle \in E$, then set $\mathcal{L}(\langle x, z\rangle) = \mathcal{L}(\langle x, z\rangle) \cup \{\mathsf{Inv}(S) \mid S \in \mathcal{L}(\langle z, y\rangle)\}$, and

(d) remove $\langle z, y\rangle$ from E;

2. for all nominal nodes z such that $\langle y, z\rangle \in E$

(a) if $\{\langle x, z\rangle, \langle z, x\rangle\} \cap E = \emptyset$, then add $\langle x, z\rangle$ to E and set $\mathcal{L}(\langle x, z\rangle) = \mathcal{L}(\langle y, z\rangle)$,

(b) if $\langle x, z\rangle \in E$, then set $\mathcal{L}(\langle x, z\rangle) = \mathcal{L}(\langle x, z\rangle) \cup \mathcal{L}(\langle y, z\rangle)$,

(c) if $\langle z, x\rangle \in E$, then set $\mathcal{L}(\langle z, x\rangle) = \mathcal{L}(\langle z, x\rangle) \cup \{\mathsf{Inv}(S) \mid S \in \mathcal{L}(\langle y, z\rangle)\}$, and

(d) remove $\langle y, z\rangle$ from E;

3. set $\mathcal{L}(x) = \mathcal{L}(x) \cup \mathcal{L}(y)$;

4. add $x \not\approx z$ for all z such that $y \not\approx z$; and

5. $\mathsf{Prune}(y)$.

If y was merged into x, we call x a *direct heir* of y, and we use being an *heir* of another node for the transitive closure of being a "direct heir".

Comment: merging is a generalisation of what is often done to satisfy an atmost number restriction for a node x in case that x has too many neighbours. In $\mathcal{SHOIQ}$, we might need to merge nominal nodes that are related in some arbitrary, non-tree-like way, so we must take care of all incoming and outgoing edges. The usage of "heir" is quite intuitive since, after y has been merged into x, x has "inherited" all of y's properties, i.e., its label, its inequalities, and its incoming and outgoing edges (except for any outgoing edges to blockable nodes, which are removed by Prune).

Level (of Nominal Nodes) Let $o_1, \ldots, o_\ell$ be all the nominals occurring in the input concept D. We define the *level* of a nominal node y inductively as follows: (i) each (nominal) node x with an $o_i \in \mathcal{L}(x)$, $1 \leq i \leq \ell$, is of level 0, and (ii) a nominal node x is of level i if x is not of some level $j < i$ and x has a neighbour that is of level $i - 1$.

Comment: if a node with a lower level is merged into another node, the level of the latter node may be reduced, but it can never be increased because Merge preserves all edges connecting nominal nodes. The completion graph initially contains only level 0 nodes.

Strategy (of Rule Application) the expansion rules are applied according to the following strategy:

1. Ro is applied with highest priority.
2. Next, R$\leqslant$ and Ro? are applied, and they are applied first to nominal nodes with lower levels. In case they are both applicable to the same node, Ro? is applied first.
3. All other rules are applied with a lower priority.

Comment: this strategy is necessary for termination, and in particular to fix an upper bound on the number of applications of Ro?.

We can now finish the description of the tableau algorithm: A completion graph is *complete* if it contains a clash, or when none of the rules is applicable. If the expansion rules can be applied to D and $\mathcal{R}$ in such a way that they yield a complete and clash-free completion graph, then the algorithm returns "D is *satisfiable* w.r.t. $\mathcal{R}$", and "D is *unsatisfiable* w.r.t. $\mathcal{R}$" otherwise.

Ro:	if for some $o \in N_I$ there are 2 nodes x, y with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and not $x \not\approx y$ then $\mathsf{Merge}(x, y)$
Ro?:	if 1. $(\leqslant nS.C) \in \mathcal{L}(x)$, x is a nominal node, and there is a blockable S-neighbour y of x such that $C \in \mathcal{L}(y)$ and x is a successor of y 2. there is no m with $1 \leqslant m \leqslant n$, $(\leqslant mS.C) \in \mathcal{L}(x)$, and there are m nominal S-neighbours $z_1, \ldots, z_m$ of x with $C \in \mathcal{L}(z_i)$ and $y_i \not\approx y_j$ for all $1 \leq i < j \leq m$. then 1. guess $m \leqslant n$ and set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{(\leqslant mS.C)\}$ 2. create m new nodes $y_1, \ldots, y_m$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C, o_i\}$ for $o_i \in N_I$ new in $\mathbf{G}$, and $y_i \not\approx y_j$ for $1 \leq i < j \leq m$,

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

Lemma 6 Let D be a $\mathcal{SHOIQ}$ concept in NNF and $\mathcal{R}$ a role hierarchy.

1. When started with D and $\mathcal{R}$, the completion algorithm terminates.
2. D has a tableau w.r.t. $\mathcal{R}$ if and only if the expansion rules can be applied to D and $\mathcal{R}$ such that they yield a complete and clash-free completion graph.

Proof (sketch): Let $m = |\mathsf{cl}(D)|$, k the number of roles and their inverses in D and $\mathcal{R}$, n the maximal number in atmost number restrictions, and $o_1, \ldots, o_\ell$ be all nominals occurring in D, and let $\lambda := 2^{2m+k}$.

Termination is a consequence of the usual $\mathcal{SHIQ}$ conditions (1 – 3 below) with respect to the blockable tree parts of the graph [Horrocks *et al.*, 1999], plus the fact that there is a bound on the number of new nominal nodes that can be added to $\mathbf{G}$ by Ro? (4 below): (1) all but the shrinking rules strictly extend the completion graph by adding new nodes and edges or extending node labels, while neither removing nodes, edges, or elements from node labels; (2) new nodes are only added by the generating rules, and each of these rules is applied at most once for a given concept in the label of a given node x or its heirs; (3) the length of a path consisting entirely of blockable nodes is bounded by λ: this is due to the blocking condition and the fact that, if x is blockable, then $\mathcal{L}(x) \subseteq \mathsf{cl}(D, \mathcal{R})$ and thus does not contain nominals; and (4) at most $\mathcal{O}(\ell(mn)^\lambda)$ nominal nodes are generated.

To see (4), observe that Ro? can only be applied after a nominal o_i, for $i \leq \ell$, has been added to the label of a blockable node x in a branch of one of the blockable "trees"; otherwise, a blockable node cannot have a nominal node as a successor. In this case, Ro will immediately merge x with an existing level 0 node, say r_i. As a consequence of this merging, it is possible that the predecessor of x is merged into a nominal node n_1 by R$\leqslant$ (due to the pruning part of merging, this cannot happen to a successor of x). By definition, n_1 is of level 0 or 1. Repeating this argument, it is possible that all ancestors of x are merged into nominal nodes. However, as the maximum length of a sequence of blockable nodes is λ, blockable ancestors of x can only be merged into nominal nodes of level below λ. Given the preconditions of Ro?, this implies that we can only apply Ro? to nominal nodes of level below λ. This, together with (2) above and some counting, yields the bound of $\mathcal{O}(\ell(mn)^\lambda)$.

The remainder of the proof is very similar to the $\mathcal{SHIQ}$ case. For the second claim in Lemma 6, for the "if" direction, we can obtain a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ from a complete and clash-free completion graph $\mathbf{G}$ by *unravelling* blockable "tree" parts of the graph as usual. For the "only if" direction, using a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ for D w.r.t. $\mathcal{R}$ to steer the non-deterministic rules R$\sqcup$, R?, R$\leqslant$, and Ro?, we can indeed construct a complete and clash-free graph. □

Since subsumption can be reduced to (un)satisfiability and $\mathcal{SHOIQ}$ can internalise general TBoxes, we have the following result.

Theorem 7 The $\mathcal{SHOIQ}$ tableau algorithm is a decision procedure for satisfiability and subsumption of $\mathcal{SHOIQ}$ concepts w.r.t. TBoxes and role hierarchies.

5 Discussion

We have presented what is, to the best of our knowledge, the first goal-directed decision procedure for $\mathcal{SHOIQ}$ (and so $\mathcal{SHOIN}$). Given that $\mathcal{SHOIQ}$ is NExpTime-complete [Tobies, 2000], it is clear that, in the worst case, any decision procedure will behave very badly, i.e., not terminate in practice. However, the algorithm given here is designed to behave well in many typically encountered cases, and to exhibit a "pay as you go" behaviour: if an input TBox, role hierarchy, and concept do not involve any one of inverse roles, number restrictions, or nominals, then Ro? will not be applied, and the corresponding non-deterministic guessing is avoided. This is even true for inputs that do involve all of these three constructors, but only in a "harmless" way. Hence, our $\mathcal{SHOIQ}$ algorithm can be implemented to perform just as well on $\mathcal{SHIQ}$ knowledge bases as state-of-the-art DL reasoners for $\mathcal{SHIQ}$ [Horrocks & Patel-Schneider, 1998; Haarslev & Möller, 2001; Pellet, 2003]. To find out whether our algorithm can handle some non-trivial, "true" $\mathcal{SHOIQ}$ inputs, we are currently extending a highly optimised DL reasoner, FaCT++ [Tsarkov & Horrocks, 2004], to implement the algorithm described here.

References

[Baader & Hollunder, 1991] F. Baader and B. Hollunder. Qualifying number restrictions in concept languages. In *Proc. of KR'91*. Morgan Kaufmann, 1991.

[Bechhofer *et al.*, 2004] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-ref/`.

[Blackburn & Seligman, 1995] P. Blackburn and J. Seligman. Hybrid languages. *J. of Logic, Language and Information*, 4, 1995.

[Calvanese *et al.*, 1998] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*. Morgan Kaufmann, 1998.

[Haarslev & Möller, 2001] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, vol. 2083 of *LNAI*. Springer, 2001.

[Hladik & Model, 2004] J. Hladik and J. Model. Tableau systems for SHIO and SHIQ. In *Proc. of DL 2004*. CEUR, 2004. Available from `ceur-ws.org`.

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, vol. 1705 of *LNAI*. Springer, 1999.

[Horrocks *et al.*, 2000] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. CADE 2000*, vol. 1831 of *LNCS*. Springer, 2000.

[Horrocks *et al.*, 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[Horrocks & Patel-Schneider, 1998] I. Horrocks and P. F. Patel-Schneider. FaCT and DLP: Automated reasoning with analytic tableaux and related methods. In *Proc. of TABLEAUX'98*, vol. 1397 of *LNAI*. Springer, 1998.

[Horrocks & Sattler, 2001] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of IJCAI 2001*. Morgan Kaufmann, 2001.

[Horrocks & Sattler, 2005] I. Horrocks and U. Sattler. A tableaux decision procedure for SHOIQ. available at `www.cs.man.ac.uk/~sattler/ulis-ps.html`, 2005.

[McGuinness & Wright, 1998] D. L. McGuinness and J. R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, 13(4), 1998.

[Pacholski *et al.*, 1997] L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In *Proc. of LICS'97*. IEEE Computer Society Press, 1997.

[Pellet, 2003] Pellet OWL reasoner. Maryland Information and Network Dynamics Lab, 2003. `http://www.mindswap.org/2003/pellet/index.shtml`.

[Schaerf, 1994] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2), 1994.

[Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12, 2000.

[Tobies, 2001] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[Tsarkov & Horrocks, 2004] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of DL 2004*. CEUR, 2004. Available from `ceur-ws.org`.

[Vardi, 1997] M. Y. Vardi. Why is modal logic so robustly decidable. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 31. American Mathematical Society, 1997.