A Refined Tableau Calculus with Controlled Blocking for the Description Logic $\mathcal{SHOI}$

Mohammad Khodadadi, Renate A. Schmidt, and Dmitry Tishkovsky*

School of Computer Science, The University of Manchester, UK

Abstract The paper presents a tableau calculus with several refinements for reasoning in the description logic $\mathcal{SHOI}$. The calculus uses non-standard rules for dealing with TBox statements. Whereas in existing tableau approaches a fixed rule is used for dealing with TBox statements, the tableau calculus uses a dynamically generated set of refined rules. This approach has become practical because reasoners with flexible sets of rules can be generated with the tableau prover generation prototype MetTeL. We also define and investigate variations of the unrestricted blocking mechanism in which equality reasoning is realised by ordered rewriting and the application of the blocking rule is controlled by excluding its application to a fixed, finite set of individual terms. Reasoning with the unique name assumption and excluding ABox individuals from the application of blocking can be seen as two separate instances of the latter. Experiments show the refinements lead to fewer rule applications and improved performance.

1 Introduction

There exist various tableau algorithms for reasoning in description logics [1]. In this paper we present a refinement of the tableau calculus introduced in [8] for the description logic $\mathcal{SHOI}$. Termination is ensured using a rewriting variant of the unrestricted blocking rule [15]. A sufficient condition for termination using unrestricted blocking is the finite model property [15]. The finite model property for $\mathcal{SHOI}$ is provided here by a standard filtration argument that does not involve tableau reasoning as in [2]. The core tableau rules are in line with a refined tableau calculus obtained in the tableau synthesis framework [14], but, exploiting the tree model property of $\mathcal{SHOI}$, transitive roles are accommodated via a propagation rule rather than a structural rule.

Different blocking mechanisms have been developed for description logic tableau algorithms. A common point of these mechanisms is that they essentially exploit kinds of the tree model property. They compare maximally expanded label sets of concept expressions through the construction of tree-like models. These blocking techniques provide strong termination results but some care is needed to ensure soundness. For more expressive logics, for example, logics with role inverse and nominals, back-and-forth traversal of a tree model is required with implicit backtracking together with forms of dynamic blocking [5,6].

* This research is supported by UK EPSRC research grant EP/H043748/1.

In [12,15], it was shown, the description logics $\mathcal{ALBO}$ and $\mathcal{ALBO}^{\mathsf{id}}$, which do not have the tree model property, can be decided using a labelled tableau approach enhanced by the unrestricted blocking mechanism, while many existing blocking mechanisms are not sufficient for description logics without a kind of tree-model property. The unrestricted blocking mechanism ensures weak termination. It is generic and reverts decisions only when needed, namely when a contradiction was obtained. While many techniques in the presented tableau calculus have similarities with existing tableau algorithms, there are also significant differences because our tableau calculus is designed to be proof-confluent and as general as possible. In this paper, we describe a rewriting variant of the unrestricted blocking rule, because equality reasoning is realised by ordered rewriting.

The paper is based on [8]. Its main contributions are twofold. First, we discuss a general technique of controlling the blocking rule by disabling its application to individual terms from an a priori given, finite set (Section 4). This approach can be utilised for reasoning in domains with the unique name assumption. Second, we use a novel approach for reasoning with respect to TBox statements (Section 5). Rather than using a fixed tableau rule for TBox statements, we dynamically generate rules for each statement. These dynamic rules are optimised by a rule refinement technique described in [16]. The MetTeL tool [17] allows us to automatically generate a prover for a specific knowledge base based on a calculus with dynamically generated rules. In order to evaluate the provers that use the tableau calculi with dynamically generated rules, an experimental comparison between them and provers that use the fixed tableau rule was undertaken (Section 6). Controlled variants of unrestricted blocking are also evaluated. Two repositories of existing ontologies are used as problem sets.

Long versions of the paper are [9] and with proofs [10].

2 Syntax and semantics of $\mathcal{SHOI}$

The description logic $\mathcal{SHOI}$ [7,5] extends the description logic $\mathcal{ALC}$ with singleton concepts, role inverse, transitive roles and role inclusion axioms. Its language is defined over disjoint sets of atomic concepts, atomic roles and individuals. The set of individuals is assumed to be finite. C and D denote concepts, A denotes an atomic concept, R and T denote roles, r denotes an atomic role, and a and b denote individuals. Concepts and roles are built from atomic concepts, individuals, and atomic roles using the connectives $\{\cdot\}$ (singleton operator), $\neg$, $\sqcup$, and $\exists \cdot . \cdot$ (existential restriction operator), $^-$ (role inverse operator) as defined by these BNFs: $C \stackrel{\text{def}}{=} A \mid \{a\} \mid \neg C \mid C \sqcup C \mid \exists R.C$ and $R \stackrel{\text{def}}{=} r \mid R^-$. The operators $\top, \bot, \sqcap$ and $\forall \cdot . \cdot$ are defined as usual. We assume that $(r^-)^- \stackrel{\text{def}}{=} r$.

A knowledge base consists of an ABox $\mathcal{A}$, a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$. A finite number of concept assertions of the form $a : C$ and role assertions of the form $(a, b) : R$ constitute the ABox. The hierarchy between concepts are expressed in the TBox using a finite set of inclusion statements of the form $C \sqsubseteq D$. The RBox is a finite set of transitivity statements $\mathsf{Trans}(r)$ for some atomic roles r

and inclusion statements of the form $R \sqsubseteq T$ which are used to express the hierarchy between roles. Normalisation of the RBox is not assumed.

We define the *closure* $\mathcal{R}^+$ of role inclusions in the RBox $\mathcal{R}$ as the smallest RBox that contains $\mathcal{R}$ and satisfies the following two properties: (i) if $Q \sqsubseteq R \in \mathcal{R}^+$ then $Q^- \sqsubseteq R^- \in \mathcal{R}^+$; (ii) if $Q \sqsubseteq R, R \sqsubseteq T \in \mathcal{R}^+$ then $Q \sqsubseteq T \in \mathcal{R}^+$. Given an RBox $\mathcal{R}$, let $\mathcal{R}^*$ denote the RBox $\mathcal{R}^+ \cup \{R \sqsubseteq R \mid R \text{ is a role}\}$.

An $\mathcal{SHOI}$*-model* $\mathcal{I}$ is a tuple $\mathcal{I} \stackrel{\text{def}}{=} (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty *domain* of interpretation and $\cdot^\mathcal{I}$ is an *interpretation function* which maps individuals to elements of $\Delta^\mathcal{I}$, concept names to subsets of $\Delta^\mathcal{I}$, and role names to binary relations over $\Delta^\mathcal{I}$. The interpretation function extends inductively to all concept and role expressions as follows.

$$\{a\}^\mathcal{I} \stackrel{\text{def}}{=} \{a^\mathcal{I}\} \quad (\neg C)^\mathcal{I} \stackrel{\text{def}}{=} \Delta^\mathcal{I} \setminus C^\mathcal{I} \quad (C \sqcup D)^\mathcal{I} \stackrel{\text{def}}{=} C^\mathcal{I} \cup D^\mathcal{I}$$
$$(\exists R.C)^\mathcal{I} \stackrel{\text{def}}{=} \{x \mid \exists y \in C^\mathcal{I}\ (x,y) \in R^\mathcal{I}\} \qquad (R^-)^\mathcal{I} \stackrel{\text{def}}{=} \{(x,y) \mid (y,x) \in R^\mathcal{I}\}$$

For any expression or statement E, E is *true (valid)* in the model $\mathcal{I}$ is denoted by $\mathcal{I} \models E$ and is defined as follows:

$$\mathcal{I} \models C \stackrel{\text{def}}{\Longleftrightarrow} C^\mathcal{I} = \Delta^\mathcal{I} \qquad \mathcal{I} \models a : C \stackrel{\text{def}}{\Longleftrightarrow} a^\mathcal{I} \in C^\mathcal{I}$$
$$\mathcal{I} \models R \sqsubseteq T \stackrel{\text{def}}{\Longleftrightarrow} R^\mathcal{I} \subseteq T^\mathcal{I} \qquad \mathcal{I} \models (a,b) : R \stackrel{\text{def}}{\Longleftrightarrow} (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$$
$$\mathcal{I} \models C \sqsubseteq D \stackrel{\text{def}}{\Longleftrightarrow} C^\mathcal{I} \subseteq D^\mathcal{I} \qquad \mathcal{I} \models \mathsf{Trans}(r) \stackrel{\text{def}}{\Longleftrightarrow} r^\mathcal{I} \text{ is transitive.}$$

A concept C is *satisfiable* in a model $\mathcal{I}$ iff $C^\mathcal{I} \neq \varnothing$. A concept is *satisfiable in* $\mathcal{I}$ *with respect to a knowledge base* if it is satisfiable in $\mathcal{I}$ whenever every statement of the knowledge base is true in $\mathcal{I}$. That is, C is satisfiable with respect to $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ in $\mathcal{I}$ iff $C^\mathcal{I} \neq \varnothing$ provided that $\mathcal{I} \models E$ for every $E \in \mathcal{A} \cup \mathcal{T} \cup \mathcal{R}$.

The termination result in the next section for a tableau calculus for $\mathcal{SHOI}$ relies on the finite model property of the logic.

Theorem 1 (Finite model property of $\mathcal{SHOI}$ [2,10]). *If a concept C is satisfiable with respect to a knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ in a $\mathcal{SHOI}$-model then it is satisfiable with respect to $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ in a finite $\mathcal{SHOI}$-model.*

3 Tableau calculus $\mathit{Tab}_{\mathcal{SHOI}}$

The language of the tableau calculus is an extension of the language of $\mathcal{SHOI}$ with equality formulae and individual terms used as labels. The set of (individual) terms s is defined inductively by the grammar rule $s \stackrel{\text{def}}{=} a \mid f(s, R, C)$, where a denotes any individual, C any concept, R any role, and f is a (fixed) function symbol. Terms which are not ABox individuals can be viewed as being Skolem terms. Formulae in the *tableau language* are ABox assertions over individual terms, and equalities of terms. More precisely, tableau formulae are defined by the following grammar rule, where s and t are individual terms, C is a concept and R is a role.

$$E \stackrel{\text{def}}{=} s : C \mid (s,t) : R \mid s \approx t$$

We extend the interpretation of $\mathcal{SHOI}$ to the tableau language as follows. For every $\mathcal{SHOI}$ interpretation $\mathcal{I}$, let the interpretation $f^{\mathcal{I}}$ in $\mathcal{I}$ of the function f be an arbitrary function mapping triples (s, R, C) with $s \in \Delta^{\mathcal{I}}$, $R \subseteq (\Delta^{\mathcal{I}})^2$, $C \subseteq \Delta^{\mathcal{I}}$ to elements of $\Delta^{\mathcal{I}}$. The semantics of tableau formulae is specified by:

$$(f(a, R, C))^{\mathcal{I}} \stackrel{\text{def}}{=} f^{\mathcal{I}}(a^{\mathcal{I}}, R^{\mathcal{I}}, C^{\mathcal{I}}), \qquad \mathcal{I} \models s : C \iff^{\text{def}} s^{\mathcal{I}} \in C^{\mathcal{I}},$$
$$\mathcal{I} \models s \approx t \iff^{\text{def}} s^{\mathcal{I}} = t^{\mathcal{I}}, \qquad \mathcal{I} \models (s, t) : R \iff^{\text{def}} (s^{\mathcal{I}}, t^{\mathcal{I}}) \in R^{\mathcal{I}}.$$

Since the interpretations of the formulae $s \approx t$, $s : \{t\}$ and $t : \{s\}$ coincide, we refer to them as *equalities*, and to formulae of the form $s : \neg\{t\}$ as *inequalities*.

Let *Tab* denote a tableau calculus comprising of a set of inference rules. A *derivation* or *tableau* for *Tab* is a finitely branching, ordered tree whose nodes are annotated by sets of tableau formulae. Assuming that C is the input concept to be tested for satisfiability with respect to a knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$, the root node of the tableau is the set $\{a : C\} \cup \mathcal{A}$, where a denotes a fresh individual and $\mathcal{A}$ is the ABox. Successor nodes are constructed in accordance with a set of inference rules in the calculus. The inference rules have the general form

$$\frac{X_0}{X_1 \mid \ldots \mid X_n} \text{ (side-condition)},$$

where X_0 is the set of premises and the X_i are the sets of conclusions. If $n = 0$, the rule is called *closure rule* and written $X_0/\bot$.

A rule is *applicable*, if the premises of the rule match formulae of one of the leaf nodes. If the rule is applied to a leaf node, then the tableau is extended, by attaching to the leaf node, n child nodes annotated with the formulae of the leaf node together with appropriate instantiations of the conclusions of the rule. In order to avoid redundancies we stipulate that a rule application is *redundant* if an annotation of one of the child nodes is contained in the leaf node annotation.

A *branch* in the tableau is a maximal path from the root of the tableau to a leaf node. If a closure rule has been applied in a branch then the branch is said to be *closed*. If a branch is not closed, it is called *open*. A tableau is *closed* if all its branches are closed. A branch is *fully expanded* if no more rules are applicable to its leaf node modulo redundancy. We call a tableau *fully expanded* iff all its branches are fully expanded. We denote by $Tab(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$ a fully expanded tableau constructed in the calculus *Tab* for the input concept C (to be tested for satisfiability) and the knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$.

In this paper, equality reasoning for individuals is done by means of ordered rewriting for efficiency reasons. If an equality formula $s \approx t$ is derived in a node, it triggers a rewriting of the current node with respect to the equalities of the node and a fixed reduction ordering.

Our tableau calculus $Tab_{\mathcal{SHOI}}$ for the description logic $\mathcal{SHOI}$ is given in Figure 1. It is not difficult to see that each rule of $Tab_{\mathcal{SHOI}}$ preserves satisfiability. Consequently we can state:

Theorem 2 (Soundness). *The tableau calculus* $\text{Tab}_{\mathcal{SHOI}}$ *is sound for* $\mathcal{SHOI}$. *That is, if a concept* C *is satisfiable with respect to the knowledge base* $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ *then any fully expanded* $\text{Tab}_{\mathcal{SHOI}}$*-tableau for* $(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$ *has an open branch.*

$$(\bot)\colon \frac{s : \neg C,\ s : C}{\bot} \qquad (\neg\neg)\colon \frac{s : \neg\neg C}{s : C}$$

$$(\exists)\colon \frac{s : \exists R.C}{f(s,R,C) : C,\ (s, f(s,R,C)) : R} \qquad (\sqcup)\colon \frac{s : C \sqcup D}{s : C \mid s : D}$$

$$(\neg\exists)\colon \frac{s : \neg\exists T.C,\ (s,t) : R}{t : \neg C}\ (R \sqsubseteq T \in \mathcal{R}^*) \qquad (\neg\sqcup)\colon \frac{s : \neg(C \sqcup D)}{s : \neg C,\ s : \neg D}$$

$$(\neg\exists^-)\colon \frac{s : \neg\exists T^-.C,\ (t,s) : R}{t : \neg C}\ (R \sqsubseteq T \in \mathcal{R}^*) \qquad (^-)\colon \frac{(s,t) : R^-}{(t,s) : R}$$

$$(\mathsf{tr})\colon \frac{s : \neg\exists T.C,\ (s,t) : R}{t : \neg\exists R.C}\ (R \sqsubseteq T \in \mathcal{R}^*,\ \mathsf{Trans}(R) \in \mathcal{R}) \qquad (\mathsf{id}_1)\colon \frac{s : C}{s : \{s\}}$$

$$(\mathsf{tr}^-)\colon \frac{s : \neg\exists T^-.C,\ (t,s) : R}{t : \neg\exists R^-.C}\ (R \sqsubseteq T \in \mathcal{R}^*,\ \mathsf{Trans}(R) \in \mathcal{R}) \qquad (\mathsf{id}_2)\colon \frac{s : \neg\{t\}}{t : \{t\}}$$

$$(\mathsf{RBox})\colon \frac{(s,t) : R}{(s,t) : T}\ (R \sqsubseteq T \in \mathcal{R}^+) \qquad (\mathsf{id}_3)\colon \frac{(s,t) : R}{s : \{s\},\ t : \{t\}}$$

$$(\mathsf{TBox})\colon \frac{s : \{s\}}{s : (\neg C \sqcup D)}\ (C \sqsubseteq D \in \mathcal{T}) \qquad (\approx)\colon \frac{s : \{t\}}{s \approx t}\ (s \neq t)$$

Figure 1. The tableau calculus $Tab_{\mathcal{SHOI}}$

A tableau calculus Tab is *complete* iff for every knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ and every concept C if C is unsatisfiable with respect to $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ then there is a closed tableau $Tab(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$.

Theorem 3 (Completeness). *$Tab_{\mathcal{SHOI}}$ is a complete tableau calculus for the description logic $\mathcal{SHOI}$.*

A form of blocking or loop-checking is necessary in order to ensure termination. We achieve termination by incorporating a variation of the *unrestricted blocking* mechanism described in [12] into the tableau calculus. In [12] equality reasoning is realised by tableau equality rules, whereas in this paper ordered rewriting is used. We therefore adapt the unrestricted blocking rule from [12] as follows:

$$(ub)\colon \frac{s : \{s\},\ t : \{t\}}{s \approx t \mid s : \neg\{t\}}\ (s \neq t).$$

Termination condition: In every open branch there is some node from which point onward before any application of the $(\exists)$ rule, all possible applications of the (ub) rule have been performed.

The (ub) rule is applicable to any pair of distinct individual terms that are used as labels in the current leaf node. When it is applied, two tableau successor nodes are created. In the left node, $s \approx t$ acts as a trigger which induces rewriting modulo derived equalities. In the right branch, the leaf node is a copy of the current leaf node extended with the additional formula $s : \neg\{t\}$, which indicates that s and t are not equal.

Let $Tab_{\mathcal{SHOI}}(ub)$ be the calculus consisting of all the rules of $Tab_{\mathcal{SHOI}}$ and the (ub) rule. Since, the (ub) rule is sound, and $Tab_{\mathcal{SHOI}}$ is sound and complete (Theorem 3), we get:

Theorem 4. *$Tab_{\mathcal{SHOI}}(ub)$ is a sound and complete for $\mathcal{SHOI}$.*

Based on [13,15] it can be shown that adding the rewriting version of unrestricted blocking to a sound and complete, ground semantic tableau calculus ensures termination, if the logic has the finite model property. A tableau calculus *Tab* is *(weakly) terminating* iff for any finite set N, every closed tableau $Tab(N)$ is finite and every open tableau $Tab(N)$ has a finite open branch [14]. A procedure based on a tableau calculus is *fair* if any inference that is possible is performed eventually [15].

Theorem 5 (Termination). *Any fair procedure based on the tableau calculus* $Tab_{\mathcal{SHOI}}(ub)$ *is terminating for satisfiability in* $\mathcal{SHOI}$.

As branch selection fairness is particularly important, this provides a weak termination result and means that in an implementation breadth-first search or more efficient depth-first iterative deepening search guarantees termination. Mainstream description logic tableau algorithms with less eager blocking conditions are strongly terminating. We also expect to be able to show strong termination for algorithms based on $Tab_{\mathcal{SHOI}}(ub)$.

Theorem 6 (Decidability). *Any fair procedure based on the tableau calculus* $Tab_{\mathcal{SHOI}}(ub)$ *and satisfying the termination condition is a decision procedure for* $\mathcal{SHOI}$ *and its sublogics.*

4 Controlling the application of blocking using (ub_{noS})

The (ub) rule may potentially create a large number of branching points in the derivation, as it is applicable to all pairs of individual terms in the branch. The situation is worse if the knowledge base contains a large number of individuals and $\exists$-expressions. It is thus important to find ways of controlling the application of blocking without loosing termination. We may reduce the number of applications of the (ub) rule, and reduce the search space by imposing appropriate side conditions on the application of the blocking rule. Ideal are side-conditions, and additional premises, that maximise the chance of constructing a finite model without the need for backtracking. It is however not possible to know which identification of individual terms will be helpful for discovering a finite model quickly. It is clear that systematic approaches for selecting individual terms to identify are needed, and different approaches display different performances.

The following theorem holds for arbitrary restrictions of the (ub) rule.

Theorem 7 (Soundness and completeness). *The (ub) rule constrained by any additional premises or side-conditions is sound.* $Tab_{\mathcal{SHOI}}$ *extended with such a constrained rule is thus sound and complete for* $\mathcal{SHOI}$.

In this section we introduce a general technique for controlling the application of the (ub) rule. One possible way of controlling the (ub) rule is to find individual terms whose identification is known not to be essential for termination. It could also be that the domain of application dictates that certain individuals cannot

be equal. For example, a subset of the ABox individuals may be assumed to be uniquely named.

Let us assume it is possible to specify a *finite* set S of individual terms which we want to exclude from blocking or know their blocking is not essential. Consider the following variation of the (ub) rule.

$$(\mathsf{ub}_{\mathsf{noS}})\colon \frac{s : \{s\},\ t : \{t\}}{s \approx t \mid s : \neg\{t\}} (t \notin S, s \neq t)$$

In particular, it is not applied to pairs of terms appearing in the set S.

Let $Tab_{\mathcal{SHOI}}(\mathrm{ub_{noS}})$ be the calculus consisting of all the rules of $Tab_{\mathcal{SHOI}}$ and the ($\mathrm{ub_{noS}}$) rule.

Theorem 8. *Let S be a finite set of individual terms. Then $Tab_{\mathcal{SHOI}}(\mathrm{ub_{noS}})$ is sound, complete and terminating for $\mathcal{SHOI}$.*

Replacing the (ub) rule with the ($\mathrm{ub_{noS}}$) rule, the calculus remains sound and complete, since the ($\mathrm{ub_{noS}}$) rule is a sound rule. However, preservation of termination needs to be formally proved. This can be done by showing that there exists a *finite open* branch for any satisfiable concept C when constructing the complete tableau using $Tab_{\mathcal{SHOI}}(\mathrm{ub_{noS}})$. Since the existence of an open branch is ensured by soundness, we just need to show there is a *finite* open branch. This can be shown by constructing a finite, fully expanded and open branch, with the use of a *model branch* built for the given concept using $Tab_{\mathcal{SHOI}}(ub)$. Guided by the model branch, a finite fully expanded branch for the same concept is constructed by $Tab_{\mathcal{SHOI}}(\mathrm{ub_{noS}})$. During the construction, an association function is used to limit the possible selection of branches to the ones that mimic the model branch. The association function is formed using the instances of blocking rule which are no longer applicable. The complete proof of a generic variant of this theorem for arbitrary description logic is presented in [10].

Different variations of the ($\mathrm{ub_{noS}}$) rule can be introduced based on how S is chosen. Possible criteria for choosing members of S are syntactic criteria, for example, individual terms that are not used as labels for any $\exists$-expressions.

Also, the ($\mathrm{ub_{noS}}$) rule can be used for reasoning modulo an implicit unique name assumption for a finite subset of the individual terms. This is expected to be more efficient than adding explicit inequality assertions to the input set to ensure the unique name assumption, which may cause a drop in performance by increasing the overhead for premise selection. Let S_i be a finite set of individual terms which are assumed to be uniquely named. For each set S_i, an instance of the ($\mathrm{ub_{noS}}$) rule should be introduced. An ontology which contains national identification numbers of people as well as student identification numbers, is a good example for this case. None of the national identification numbers (represented by individuals) should be identifiable, equally no student identification numbers should refer to the same person. But a national identification number and a student identification number can refer to the same person.

In our setting, ABox individuals are not excluded from being blocked as in many description logic tableau systems and the blocking rule is applicable to

the pairs of ABox individuals. So, we may form a set S using all the ABox individuals. Then, similar to [4], no terms from S are identified which were not created during the derivation. For this case individuals in S need to be specified to be smallest with respect to the reduction ordering $\prec$ and this instance of the ($\mathrm{ub}_{\mathrm{noS}}$) rule needs to be used.

$$(\mathsf{ub}_{\mathsf{noABox}}): \frac{s : \{s\},\, t : \{t\}}{s \approx t \mid s : \neg\{t\}} \ (t \text{ is not an ABox individual }, \ s \neq t)$$

5 Refined tableau calculus

In this section we refine the calculus $Tab_{\mathcal{SHOI}}$ presented in Section 3. The idea of the refinement is that the (TBox) rule is replaced by dynamically generated and refined tableau rules. In the first step, all the atomic concepts in the TBox $\mathcal{T}$ are equi-satisfiably replaced by constant concepts and the parametric (TBox) rule is represented as a set of tableau rules for each $C \sqsubseteq D \in \mathcal{T}$. The benefit of this replacement of the (TBox) rule by a set of rules is the possibility of refining the rules. This allows to reduce the branching factor of the rules, while preserving soundness and completeness.

In the second step, we apply the *atomic rule refinement* introduced in [16]. Atomic rule refinement is a special case of general rule refinement which is introduced in [14]. Under this refinement, all conclusions of a rule that are of the form $s : \neg A$, where A is an atomic concept or a singleton, are moved to the premise position of the rule as $s : A$. For example, for the TBox statement $A \sqcap B \sqsubseteq C$ the rule

$$\frac{s : \{s\}}{s : \neg A \mid s : \neg B \mid s : C} \quad \text{is refined to} \quad \frac{s : A,\, s : B}{s : C}.$$

We apply the atomic rule refinement to all rules obtained from TBox statements. Consequently there are fewer branches in the conclusion and additional premises are added that limit application of the rules. (Similar refinements on instances of the (RBox) rule are possible for more expressive logics with negated role assertions.)

Let $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ denote the calculus which consists of the refined generated tableau rules from the TBox $\mathcal{T}$ and all rules of $Tab_{\mathcal{SHOI}}$ except the (TBox) rule. That is, for each statement $C \sqsubseteq D \in \mathcal{T}$ a corresponding tableau rule is generated and refined according to the atomic rule refinement. Soundness and completeness of $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ is a direct consequence of the results in [16].

Theorem 9. *$Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ is a sound, complete and terminating tableau calculus for reasoning in $\mathcal{SHOI}$ with respect to a knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ with a fixed TBox $\mathcal{T}$.*

6 Implementation and experimental results

In order to compare the performance of provers based on the calculi $Tab_{\mathcal{SHOI}}(ub)$ and $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$, an experiment has been designed. MetTeL version 2.0-487

was used to generate provers based on $Tab_{\mathcal{SHOI}}(ub)$ and $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ for various ontologies. MetTeL generates Java code for a tableau prover from the specification of the syntax of a logic and the specification of tableau calculus. By default, the tableau provers generated with MetTeL use a depth-first left to right search strategy. While specifying the specification of tableau calculus, appropriate rule priorities were assigned to ensure fairness of the expansion strategy and hence guarantee termination. The generated provers were used with no modification in this experiment.

In order to embrace an extensive range of problems with varying input sizes and expressivity, the experiment used the TONES ontology repository [18] and the corpus of OWL DL ontologies from [11]. The complete repositories of 874 ontologies were downloaded. A translator using the OWL API [3] was developed to prepare appropriate input for MetTeL. This translator converts each ontology into three forms. The first form provided input to Fact++ [19] to validate the translation and outputs of the provers. The second form, translated the ontology so that we could check its consistency with a prover generated by MetTeL using the specification of $Tab_{\mathcal{SHOI}}(ub)$ as tableau specification. The third form was used in two ways. First, it was used to produce a tableau specification for $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ containing the dynamic rules generated from the ontology. Second, the remaining ontology axioms were translated so that the prover generated using the specification of $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ could check its consistency. Inputs prepared for both provers were then used with a log file from Fact++ to produce additional problem sets. Fact++ produces a log file which contains the class hierarchy of the ontology. For a randomly picked subsumption relation $C \sqsubseteq D$ in the hierarchy and a fresh individual s, $s : C$ and $s : D$ were added to the input file to form an additional satisfiable input and respectively $s : C$ and $s : \neg D$ to form an additional unsatisfiable input. This experiment was aimed at evaluating the effect on reasoning performance when using $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ in comparison to $Tab_{\mathcal{SHOI}}(ub)$. This means we checked the consistency of the input and omitted checking satisfiability of all concepts and calculating concept hierarchies.

The developed translator successfully translated 628 ontologies and each prover was executed on 2480 inputs with a timeout of 100 seconds. The comparison was done by measuring the execution time of the prover. The results of this comparison are presented in Table 1. For the set of results *with timeout*, when a prover did not return any answer within 100 seconds, 100 seconds were used in the calculation of the average time. While for the set of results *without timeout*, if one of the provers under comparison required more than 100 seconds, that input is not included in the results. The results show that the generated provers based on the refined tableau calculus were faster for unsatisfiable inputs. Inspection showed this was mainly a consequence of having additional closure rules. These closure rules were refinements of dynamically generated rules from TBox statements where all the conclusions are turned into premises in a rule. A significant drop in memory use was exhibited when using $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$ compared to $Tab_{\mathcal{SHOI}}(ub)$ specially for unsatisfiable inputs. As expected, the performance of the system was not comparable with Fact++.

	With timeout			Without timeout		
Input	count	$Tab_{\mathcal{SHOI}}(ub)$	$Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$	count	$Tab_{\mathcal{SHOI}}(ub)$	$Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$
Ontology consistency	628	**27.627**	43.094	346	**0.951**	**1.049**
Satisfiable inputs	924	**60.847**	65.999	180	13.447	**0.869**
Unsatisfiable inputs	928	21.521	**3.643**	760	5.053	**1.841**

Table 1. Average run times in seconds for $Tab_{\mathcal{SHOI}}(ub)$ and $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(ub)$

	With timeout			Without timeout		
Input	count	(ub)	$(\mathrm{ub}_{\mathrm{noABox}})$	count	(ub)	$(\mathrm{ub}_{\mathrm{noABox}})$
Ontology consistency	628	**27.627**	37.162	346	**0.951**	1.650
Satisfiable inputs	924	**60.847**	74.466	180	**13.447**	**14.043**
Unsatisfiable inputs	928	**21.521**	**21.842**	760	**5.053**	**5.223**

Table 2. Average run times in seconds for $Tab_{\mathcal{SHOI}}(ub)$ and $Tab_{\mathcal{SHOI}}(\mathrm{ub}_{\mathrm{noABox}})$

Moreover, an experiment to compare the performance of $Tab_{\mathcal{SHOI}}(ub)$ and $Tab_{\mathcal{SHOI}}(\mathrm{ub}_{\mathrm{noABox}})$, using the same inputs as before, was designed. Since it is not yet possible to express rules such as the $(\mathrm{ub}_{\mathrm{noS}})$ rule in the MetTeL tableau rule language, we generated a prover for the tableau calculus $Tab_{\mathcal{SHOI}}$ without any blocking mechanism. Then, code implementing the $(\mathrm{ub}_{\mathrm{noABox}})$ rule was manually added to the generated Java code. In order to have a fair comparison, the prover for the (ub) rule was also created by manually adding code implementing the (ub) rule. The results of the comparison are presented in Table 2.

The experimental results show there is not a big difference between the performance of the provers based on $Tab_{\mathcal{SHOI}}(ub)$ and $Tab_{\mathcal{SHOI}}(\mathrm{ub}_{\mathrm{noABox}})$. This is mainly caused by the small number of ABox individuals in a large number of ontologies in the test set.

7 Concluding remarks

A refined version of the tableau calculus in [8] was presented which uses dynamically generated tableau rules when reasoning with respect to a knowledge base. Following the presented procedure in [16] where one can refine tableau rules to reduce the branching factor, the generated tableau rules are refined. This paper investigated a controlled variant of the unrestricted blocking rule not applied to members of an a priori defined, finite set. This variant can be utilised for scenarios such as reasoning under unique name assumption.

A comparison was done between the provers generated using the tableau calculus with dynamically generated tableau rules, and a prover with fixed rules for dealing with TBox and RBox statements. The results show the former is more optimised for unsatisfiable inputs. The analysis of the reduction in the branching points and complexity is left as future work.

Other future plans include studying the relationship between properties of a logic and its required minimal blocking criteria. That is, expressing side conditions that can be used to control the unrestricted blocking rule to be applied as little as possible. This should be done without endangering termination.

References

1. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.
2. C. L. Duc and M. Lamolle. Decidability of description logics with transitive closure of roles in concept and role inclusion axioms. In V. Haarslev, D. Toman, and G. E. Weddell, eds, *Proc. DL'10*, vol. 573 of *CEUR Workshop Proceedings*, 2010.
3. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
4. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proc. KR'06*, pp. 57–67. AAAI Press, 2006.
5. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Logic Comput.*, 9(3):385–410, 1999.
6. I. Horrocks and U. Sattler. A tableau decision procedure for SHOIQ. *J. Automat. Reason.*, 39(3):249–276, 2007.
7. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. LPAR'99*, vol. 1705 of *LNCS*, pp. 161–180. Springer, 1999.
8. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. An abstract tableau calculus for the description logic SHOI using unrestricted blocking and rewriting. In *Proc. DL'12*, vol. 846 of *CEUR Workshop Proceedings*, pp. 224–234, 2012.
9. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. Refined tableau with controlled blocking for the description logic SHOI. To appear in *Proc. TABLEAUX'13*, 2013.
10. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. Refined tableau with controlled blocking for the description logic SHOI. Manuscript, `http://www.mettel-prover.org/papers/controlled.pdf`, 2013.
11. N. Matentzoglu, S. Bail, and B. Parsia. A corpus of OWL DL ontologies. To appear in *Proc. DL'13*, 2013.
12. R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *Proc. ISWC+ASWC'07*, pp. 438–451. Springer, 2007.
13. R. A. Schmidt and D. Tishkovsky. A general tableau method for deciding description logics, modal logics and related first-order fragments. In *Proc. IJCAR'08*, vol. 5195 of *LNCS*, pp. 194–209. Springer, 2008.
14. R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Logical Methods in Comput. Sci.*, 7(2):1–32, 2011.
15. R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *arXiv e-Print*, abs/1208.1476, 2012.
16. D. Tishkovsky and R. A. Schmidt. Refinement in the tableau synthesis framework. *arXiv e-Print*, abs/1305.3131, 2013.
17. D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeI^2. In *Proc. JELIA'12*, vol. 7519 of *LNAI*, pp. 492–495. Springer, 2012.
18. TONES. The tones ontology repository, 5 Mar. 2013.
19. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *Proc. IJCAR'06*, LNCS, pp. 292–297. Springer, 2006.