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, we use 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 [2]. In this paper we present a refinement of the tableau calculus introduced in [12] for the description logic $\mathcal{SHOI}$. Termination is ensured using a rewriting variant of the unrestricted blocking rule [19]. A sufficient condition for termination using unrestricted blocking is the finite model property [19], which $\mathcal{SHOI}$ is known to have [5]. The core tableau rules are in line with a refined tableau calculus obtained in the tableau synthesis framework [18], but, exploiting the tree model property of $\mathcal{SHOI}$, transitive roles are accommodated via propagation rules rather than structural rules.

Labelled tableau approaches allow for a flexible derivation procedure, and are not limited to logics with a form of tree model property. They are common for modal and description logics, hybrid logics and various other non-classical logics, cf. e.g., [6,2,3,4,1].

Different blocking mechanisms have been developed for description logic tableau algorithms. A common point of these mechanisms is essentially that they exploit kinds of the tree model property. They compare maximally expanded label sets of concept expressions through the construction of tree-like models.

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

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 using forms of dynamic blocking [9,10]. These blocking techniques provide strong termination results but some care is needed to ensure soundness. In [16,19], 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 only contradictions were obtained. While many techniques in the tableau calculus presented in this paper have similarities with techniques in existing tableau approaches, there are also significant differences because our tableau calculus is designed to be proof-confluent and as general as possible. We describe a rewriting variant of the unrestricted blocking rule, because equality reasoning is realised by ordered rewriting. In comparison to tableau calculi using essentially standard tableau rules for equality, as for example [3,16,19], rewriting performs fewer inferences. That is because essentially equivalent inference steps on equivalent individuals are avoided. The ordering ensures only the currently smallest individual term in an equivalence class is present in the current node of the tableau derivation.

While being generic the unrestricted blocking rule creates potentially a large number of branching points in the derivation. It is thus important to investigate ways of controlling the application of blocking while preserving soundness, completeness and termination. The number of applications of the blocking rule can be reduced by imposing additional side conditions or adding premises to the rule. In this paper we discuss a general technique of controlling the blocking rule by not applying it to members of an a priori given, finite set of individual terms. This variant of the blocking rule can be utilised for reasoning in domains with the unique name assumption, or where for example it is assumed that some of the given ABox individuals are distinct.

Non-standard in our approach is the use of dynamically generated rules for statements in the TBox rather than using a fixed rule. These dynamic rules are rule refinements obtained in accordance with [20]. Using the MetTeL tool [22] it is easy to generate tableau provers from tableau calculus specifications. This means there is no need to implement a prover for the new tableau calculi manually. This enables us to easily generate a prover for a specific knowledge base based on a calculus with dynamically generated rules. Following this approach we can build specialised provers for various computer applications using ontologies as the information backbone.

The paper is based on the workshop paper [12]. Its main contributions are threefold. First, it presents a labelled tableau calculus for the description logic $\mathcal{SHOI}$ (Section 3). Second, 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. Third, 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 atomic rule refinement described in [20].

The MetTeL tool [22] 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.

Additionally, we establish the finite model property for $\mathcal{SHOI}$ (Section 2). In [5], the finite model property for $\mathcal{SHOI}$ is obtained from a terminating tableau algorithm. By contrast, our proof of termination in Section 3 takes the reverse route. In the extended version [13] of this paper we provide an alternative proof of the finite model property for $\mathcal{SHOI}$ by a standard filtration argument that does not involve any form of tableau reasoning.

The paper is an extended version of the workshop paper [14]. Due to space limitations all proofs are omitted but can be found in [13].

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

The description logic $\mathcal{SHOI}$ [11,9] 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:

$$C \overset{\mathsf{def}}{=} A \mid \{a\} \mid \neg C \mid C \sqcup C \mid \exists R.C \qquad \text{and} \qquad R \overset{\mathsf{def}}{=} r \mid R^{-}.$$

The operators $\top, \bot, \sqcap$ and $\forall\cdot.\cdot$ are defined as usual. We assume that $(r^{-})^{-} \overset{\mathsf{def}}{=} r$ in order to simplify the syntax and avoid repetitive occurrences of the role inverse operator. Further, for every atomic role r, $\mathsf{Trans}(r)$ is used to specify that r is transitive. (The predicate Trans is defined on atomic roles only because, in order to specify that r^{-} is transitive, it is enough to state that r is transitive.)

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 $\mathcal{R}^+ \cup \{R \sqsubseteq R \mid R \text{ is a role}\}$.

A $\mathcal{SHOI}$*-model* $\mathcal{I}$ is a tuple $\mathcal{I} \overset{\mathsf{def}}{=} (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, 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 $\Delta^\mathcal{I}$, atomic concepts to subsets of $\Delta^\mathcal{I}$, and atomic roles to binary relations over $\Delta^\mathcal{I}$. The interpretation function extends inductively to all concept and role expressions as follows.

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

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.

$$\begin{array}{lll@{\qquad}lll}
\mathcal{I} \models C & \overset{\mathsf{def}}{\iff} & C^\mathcal{I} = \Delta^\mathcal{I} & \mathcal{I} \models a : C & \overset{\mathsf{def}}{\iff} & a^\mathcal{I} \in C^\mathcal{I} \\
\mathcal{I} \models R \sqsubseteq T & \overset{\mathsf{def}}{\iff} & R^\mathcal{I} \subseteq T^\mathcal{I} & \mathcal{I} \models (a,b) : R & \overset{\mathsf{def}}{\iff} & (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I} \\
\mathcal{I} \models C \sqsubseteq D & \overset{\mathsf{def}}{\iff} & C^\mathcal{I} \subseteq D^\mathcal{I} & \mathcal{I} \models \mathsf{Trans}(r) & \overset{\mathsf{def}}{\iff} & r^\mathcal{I} \text{ is transitive}
\end{array}$$

A concept C is *satisfiable* in a model $\mathcal{I}$ iff $C^\mathcal{I} \neq \emptyset$. 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 $C^\mathcal{I} \neq \emptyset$ 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 our tableau calculus for $\mathcal{SHOI}$ relies on the finite model property of the logic.

Theorem 1 (Finite model property of $\mathcal{SHOI}$ [5,13]). *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 $\boldsymbol{Tab_{\mathcal{SHOI}}}$

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 $s \overset{\mathsf{def}}{=} a \mid f(s, R, C)$, 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 grammar rule $E \overset{\mathsf{def}}{=} s : C \mid (s,t) : R \mid s \approx t$, where s and t are individual terms, C is a concept and R is a role.

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 that maps triples (x, ρ, χ) with $x \in \Delta^{\mathcal{I}}$, $\rho \subseteq (\Delta^{\mathcal{I}})^2$, $\chi \subseteq \Delta^{\mathcal{I}}$ to elements of $\Delta^{\mathcal{I}}$. The semantics of tableau formulae is specified by:

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

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*.

Having defined the tableau language, next we give a general description of how tableau derivations are constructed and define important notions of tableaux. 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 $X_0/\bot$.

If a rule of the calculus is applicable to a leaf node of the tableau with a matching substitution μ, and it is applied to the leaf node, then the tableau is extended by attaching to the leaf node n child nodes annotated with $N \cup X_i\mu$ for $i = 1, \ldots n$, respectively. In order to avoid redundancies we stipulate that a rule application to a leaf node annotated with N is *redundant* if there is a conclusion set X_i for some $i = 1, \ldots n$ of the rule such that $X_i\mu \subseteq N$, where μ is the matching substitution. This ensures rules are not applied more than once to the same sets of formulae.

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 $\mathit{Tab}(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$ a fully expanded tableau constructed using the calculus *Tab* for the input concept C (to be tested for satisfiability) and the knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$.

We use equality reasoning for individual terms to achieve termination for the calculus. Equality reasoning can be provided in various ways. One is to supply special tableau rules for reasoning modulo equalities within the branch in a similar way as it is done in [3,16,19]. Another is to use ordered term rewriting. Ordered rewriting is more efficient for handling equal individuals because it allows to reduce the number of tableau formulae in the current branch. Since all individual terms in any tableau derivation are ground, we are dealing with a special case of rewriting, namely, ground rewriting.

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

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

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

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

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

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

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

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

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

$$(\sqcup)\colon \frac{s:C \sqcup D}{s:C \mid s:D}$$

$$(\neg\sqcup)\colon \frac{s:\neg(C \sqcup D)}{s:\neg C,\ s:\neg D}$$

$$(^-)\colon \frac{(s,t):R^-}{(t,s):R}$$

$$(\mathsf{id}_1)\colon \frac{s:C}{s:\{s\}}$$

$$(\mathsf{id}_2)\colon \frac{s:\neg\{t\}}{t:\{t\}}$$

$$(\mathsf{id}_3)\colon \frac{(s,t):R}{s:\{s\},\ t:\{t\}}$$

$$(\approx)\colon \frac{s:\{t\}}{s \approx t}\ (s \neq t)$$

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

In this paper, a *rewrite system* R is a binary relation on the set of all individual terms and consists of rewrite rules which are pairs of individual terms. In order to handle equalities, we orient each equality formula appearing in the current branch according to a special, strict partial ordering $\succ$ on individual terms. We denote by $s \to t$ a rewrite rule (s,t) in which $s \succ t$. Thus, if an equality formula $s \approx t$ appears in a node of a branch then either $s \to t$ or $t \to s$ is added as a rewrite rule to the rewrite system of the branch.

Our tableau calculus $Tab_{\mathcal{SHOI}}$ for the description logic $\mathcal{SHOI}$ is given in Figure 1. The $(\bot)$ rule is the closure rule. The $(\neg\neg)$ rule removes occurrences of double negation on concepts. The $(\sqcup)$ and $(\neg\sqcup)$ rules are standard rules for handling concept disjunctions. Given a tableau formula $s : \exists R.C$, the $(\exists)$ rule introduces Skolem term $f(s,R,C)$, as an R-successor of s (instead of introducing a fresh individual as might be done in other presentations). Using Skolem terms has many advantages that outweigh drawbacks and perceived inconveniences. In our setting, Skolem terms provide a convenient technical device to keep track of the order in which witnesses for existential quantification were introduced and record dependency on other witnesses. Such dependencies are then used to stop redundant rule applications when combined with term rewriting. In systems not using Skolem terms this information is typically captured by an ordering on individual constants. In addition, in combination with blocking there is no need to redo inference steps with existential extent that have already been performed or resurrect phantom concepts. That is because when rewriting happening on terms, we also rewrite their dependent Skolem terms and consequently some applications of the $(\exists)$ rule become redundant. For example, rewriting of the term $f(i,R,C)$ to i causes terms such as $f(f(i,R,C),R,C)$ and $f(f(f(i,R,C),R,C),R,C)$, which may appear in formulae in a branch, to be rewritten to i. There is also no need for status variables to keep track of whether individual constants are active or phantom in the deduction process.

The $(\neg\exists)$ rule is equivalent to the standard rule for universally restricted concepts. The $(\neg\exists^-)$ rule allows the backward propagation of concepts along inverted links. The $(^-)$ rule inverts a given link. The (tr) rule propagates negated existential concept restriction along a transitive link, while the (tr^-) rule does the same for inverse occurrences of transitive roles.

Equalities of the form $s : \{s\}$ are tautologies, which are used in our calculus as domain predicates for keeping track of the terms that have been introduced to a branch. This is achieved with the three rules (id_1), (id_2) and (id_3).

The $(\approx)$ rule is a special rule adding, what we call, a *rewrite trigger* $s \approx t$ to the branch. Let $\succ$ be any reduction ordering on the set of individuals in the branch. The addition of any tableau formula $s \approx t$ to a set N of formulae, which annotates a leaf tableau node, immediately triggers the following rewrite process. Suppose that $s \succ t$ (the case $t \succ s$ is symmetrical). Then, $s \to t$ is added to a rewrite system R associated with the current tableau branch. The tableau is extended by attaching one child node to the current leaf node. The child node is annotated by the set N' obtained by rewriting all the tableau formulae in N with respect to the rewrite system R. In particular, this means that, in N' every term s is replaced by a term u such that $s \overset{*}{\to} u$ with respect to R.

For each concept inclusion $C \sqsubseteq D$ of the TBox, the (TBox) rule propagates the concept $\neg C \sqcup D$ to every label occurring on the branch. The (RBox) rule propagates a link of a role into its super role according to the closure $\mathcal{R}^+$ of the given RBox $\mathcal{R}$. In Section 5 we replace the (TBox) rule by dynamically generated rules.

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* $\mathrm{Tab}_{\mathcal{SHOI}}$ *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* $\mathrm{Tab}_{\mathcal{SHOI}}$*-tableau for* $(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$ *has an open branch.*

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 $Tab(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$. In order to prove completeness of $Tab_{\mathcal{SHOI}}$, we prove its constructive completeness, which implies completeness. A tableau calculus *Tab* is *constructively complete* if for every open branch in any fully expanded tableau $Tab(\mathcal{A}, \mathcal{T}, \mathcal{R}, C)$ there is a model that validates the knowledge base $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ and satisfies C.

Theorem 3 (Completeness). $\mathrm{Tab}_{\mathcal{SHOI}}$ *is a (constructively) 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 [16] into the tableau calculus. In [16] equality reasoning is realised by tableau equality rules, whereas in this paper ordered rewriting is used. We therefore adapt the unrestricted blocking rule from [16] as follows:

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

In order to achieve termination the following condition must hold.

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 node, $s : \neg\{t\}$ indicates that s and t are not equal. The blocking is reversible, because, when no models can be found in the left branch, reversion is performed through standard backtracking.

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 (constructively) complete (Theorem 3), we get:

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

Based on [17,19] it can be shown that adding the rewriting version of unrestricted blocking to a sound and constructively 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 [18]. A procedure based on a tableau calculus is *fair* if any inference that is possible is performed eventually [19].

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 the more efficient depth-first iterative deepening search gives a decision procedure. Mainstream description logic tableau algorithms with less eager blocking conditions are strongly terminating. We expect to be able to show termination for algorithms based on $Tab_{\mathcal{SHOI}}$(ub) using depth-first left-to-right search as well.

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 ($\mathbf{ub_{noS}}$)

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. Also if the input concept is unsatisfiable with respect to the knowledge base then no blocking inference steps are needed. However not blocking is not an option, as it is not known in advance if a problem is unsatisfiable or not. Examples show without blocking, it is not possible to avoid infinite branches. 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 aid the discovery of 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 constructively 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.

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

In contrast to the unrestricted blocking rule, the rule is applicable to a pair of distinct terms s and t if *at least one* of them does not belong to S. In other words, the rule is not applied to two terms if *both* belong to S. This means the rule is not symmetric with respect to s and t, but this is not essential. One can however consider a symmetric variant of the rule where s and t are both required to be outside of S. Although the application of the symmetric rule is even more restricted, Theorem 8 below remains true.

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 $\mathit{Tab}_{\mathcal{SHOI}}(\mathrm{ub}_{\mathrm{noS}})$. 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 $\mathit{Tab}_{\mathcal{SHOI}}(\mathrm{ub})$. Guided by the model branch, a finite fully expanded branch for the same concept is constructed by $\mathit{Tab}_{\mathcal{SHOI}}(\mathrm{ub}_{\mathrm{noS}})$. 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 the blocking rule which are no longer applicable. The complete proof of a generic variant of this theorem for arbitrary description logics is presented in [13].

Different variations of the $(\mathrm{ub}_{\mathrm{noS}})$ 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 $(\mathrm{ub}_{\mathrm{noS}})$ rule can be used for reasoning modulo an implicit unique name assumption for a finite subset of the individuals. This is expected to be more efficient than adding explicit inequality assertions to the input set to ensure 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 $(\mathrm{ub}_{\mathrm{noS}})$ 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 [8], 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.

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

5 Refined tableau calculus

In this section we refine the calculus $\mathit{Tab}_{\mathcal{SHOI}}$ 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 $C \sqsubseteq D \in \mathcal{T}$. That is, rather than one rule schema for all statements, a set of rules, one for each statement, is present in the calculus. The following rule is generated for each statement $C \sqsubseteq D$ from the TBox $\mathcal{T}$.

$$\frac{s : \{s\}}{s : \neg C \mid s : D}$$

Subsequently, this rule is transformed to an equivalent rule where the disjunctive normal forms of the negation normal forms of $\neg C$ and D are split into branches of the rule. For example, for the TBox statement $\mathsf{Horse} \sqcap \mathsf{Baby} \sqsubseteq \mathsf{Foal}$, the following rule is obtained.

$$\frac{s : \{s\}}{s : \neg\mathsf{Horse} \mid s : \neg\mathsf{Baby} \mid s : \mathsf{Foal}}$$

Notice that all the atomic concepts in the generated rules are constants and they can only match with themselves. The benefit of such a 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 (constructive) completeness, by using *atomic rule refinement* introduced in [20]. Atomic rule refinement is a special case of general rule refinement which was introduced in [18]. Under the atomic rule 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 of the rule as $s : A$. For example, the rule for the TBox statement $\mathsf{Horse} \sqcap \mathsf{Baby} \sqsubseteq \mathsf{Foal}$ is refined to the following rule.

$$\frac{s : \mathsf{Horse},\ s : \mathsf{Baby}}{s : \mathsf{Foal}}$$

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

Let $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ 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 atomic rule refinement. Soundness and completeness of $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ is a direct consequence of the results in [20].

Theorem 9. *$Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ is sound, constructively 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 analyse the practical benefit of atomic rule refinement and the $(\mathrm{ub}_{\mathrm{noS}})$ rule, two experiments were designed. MetTeL version 2.0-487 was used to generate provers based on variants of $Tab_{\mathcal{SHOI}}(\mathrm{ub})$ and $Tab^{\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ for various ontologies. MetTeL generates Java code for a tableau prover from the specification of the syntax of a logic and the specification of a tableau calculus.[1] 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 the fairness of the expansion strategy and hence guarantee termination. The generated provers were used with no modification in this experiment. For simplicity of implementation, instead of the propagation rules (tr) and (tr^-) standard transitivity rules were used in the calculi. We indicate the variations by the superscript $^+$.

In order to embrace an extensive range of problems with varying input sizes and expressivity, the experiment used the TONES ontology repository [23] and the corpus of OWL DL ontologies from [15]. The complete repositories of 874 ontologies were downloaded. A translator using the OWL API [7] was developed to prepare appropriate input for MetTeL. Each ontology was converted into three forms with the translator. The first form provided input to Fact++ [24] which was then used to validate the translation and outputs of the provers. The second form were translations of the ontology so that we could check its consistency with a prover generated by MetTeL using $Tab^{+}_{\mathcal{SHOI}}(\mathrm{ub})$ as the tableau specification. The third form was used in two ways. First, it was used to produce a tableau specification for $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ containing the dynamic rules generated from the ontology. Second, the remaining ontology axioms were translated so that the prover generated using the specification of $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ could check its consistency. Inputs prepared for both provers were then used with a results file from Fact++ to produce additional problem sets. One of the results files produced by Fact++ 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 $s : \neg D$ were added to form an additional unsatisfiable input. This experiment was aimed at evaluating the effect on reasoning performance when using $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ in comparison to $Tab^{+}_{\mathcal{SHOI}}(\mathrm{ub})$. This means we checked the consistency of the input but 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 the comparison of $Tab^{+}_{\mathcal{SHOI}}(\mathrm{ub})$ and $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}(\mathrm{ub})$ 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 in 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

[1] More information about how to generate a tableau prover using MetTeL is available in [21].

	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)	(ub$_{\text{noABox}}$)	count	(ub)	(ub$_{\text{noABox}}$)
Ontology consistency	628	43.094	**35.893**	346	**1.049**	**1.025**
Satisfiable inputs	924	65.999	**56.300**	180	0.869	**0.661**
Unsatisfiable inputs	928	**3.643**	**3.635**	760	**1.841**	**1.832**

Table 2. Average run times in seconds for $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub) and $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub$_{\text{noABox}}$)

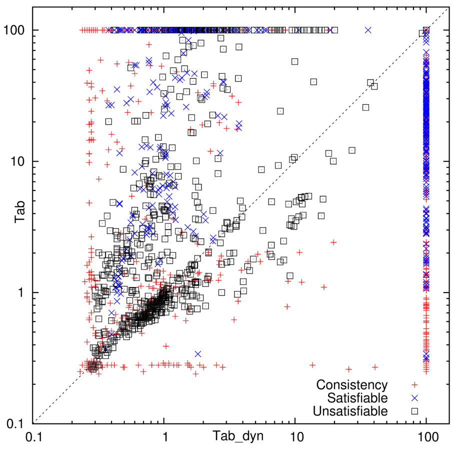
a consequence of having additional closure rules. These closure rules were refinements of dynamically generated rules from TBox statements where all the conclusions have been turned into premises in a rule. The scatter plot on the left of Figure 2 gives a more differentiated picture of the performance. On average we observed a 22% drop in memory use for satisfiable inputs and a 74% drop for unsatisfiable inputs when using $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub) in comparison to $Tab^{+}_{\mathcal{SHOI}}$(ub). As expected, the performance of the systems were not comparable with FACT++.

Moreover, an experiment to compare the performance of $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub) and $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub$_{\text{noABox}}$), using the same inputs as before, was designed. Since it is not yet possible to express rules such as the (ub$_{\text{noS}}$) rule in the METTEL rule specification language, we generated a prover for the tableau calculus $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$ without any blocking mechanism. Then, code implementing the (ub$_{\text{noABox}}$) 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 and on the right in Figure 2.

The experimental results show using the (ub$_{\text{noABox}}$) rule had a small benefit in most cases, but there was a group of satisfiable problems not solved using the (ub) rule within the timeout that could be solved under 10 seconds when using the (ub$_{\text{noABox}}$) rule. A closer analysis of some of the problems suggested this was because they implicitly force the unique name assumption for a large number of ABox individuals.

7 Concluding remarks

A tableau decision procedure for the description logic $\mathcal{SHOI}$ was presented in this paper. A refined version of the tableau calculus in [12] was presented



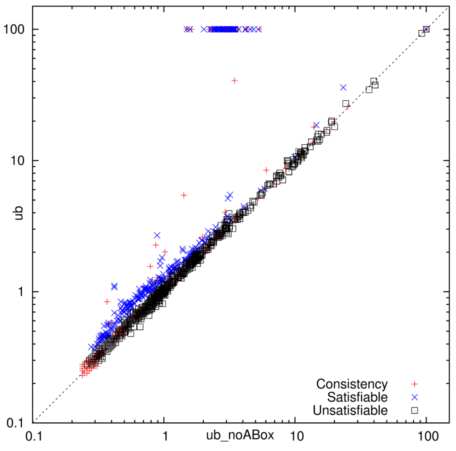


Figure 2. Scatter plots of $Tab^{+}_{\mathcal{SHOI}}$(ub) vs. $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub) and $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$(ub) vs. $Tab^{+,\mathsf{dyn},\mathcal{T}}_{\mathcal{SHOI}}$($\mathrm{ub}_{\mathrm{noABox}}$)

which uses dynamically generated tableau rules when reasoning with respect to a knowledge base. Following a rule refinement technique in [20] the generated tableau rules were refined leading to a smaller search space. We 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 the fixed rule for dealing with TBox statements. The results showed the former is more optimised especially 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 minimally required 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. Expressing existing blocking mechanisms as variants of unrestricted blocking mechanism, is also one of our future plans. Using these results, we hope to be able to provide uniform explanations and implementations of blocking mechanisms in tableau provers.

References

1. R. Alenda, N. Olivetti, C. Schwind, and D. Tishkovsky. Tableau calculi for CSL over min-spaces. In *Proc. CSL'10*, vol. 6247 of *LNCS*, pp. 52–66. Springer, 2010.
2. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

3. T. Bolander and P. Blackburn. Termination for hybrid tableaus. *J. Logic Comput.*, 17(3):517–554, 2007.
4. M. Cialdea Mayer and S. Cerrito. Nominal substitution at work with the global and converse modalities. In *Proc. AiML-8*, pp. 57–74. College Publ., 2010.
5. C. L. Duc and M. Lamolle. Decidability of description logics with transitive closure of roles in concept and role inclusion axioms. In *Proc. DL'10*, vol. 573 of *CEUR Workshop Proceedings*, 2010.
6. M. Fitting. *Proof methods for modal and intuitionistic logics*. Kluwer, 1983.
7. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
8. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proc. KR'06*, pp. 57–67. AAAI Press, 2006.
9. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Logic Comput.*, 9(3):385–410, 1999.
10. I. Horrocks and U. Sattler. A tableau decision procedure for SHOIQ. *J. Automat. Reasoning*, 39(3):249–276, 2007.
11. 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.
12. 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.
13. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. A refined tableau calculus with controlled blocking for the description logic SHOI. `http://www.mettel-prover.org/papers/controlled.pdf`, 2013.
14. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. A refined tableau calculus with controlled blocking for the description logic SHOI. To appear in *Proc. DL'13*, *CEUR Workshop Proceedings*, 2013.
15. N. Matentzoglu, S. Bail, and B. Parsia. A corpus of OWL DL ontologies. To appear in *Proc. DL'13*, *CEUR Workshop Proceedings*, 2013.
16. 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.
17. 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.
18. R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Logical Methods in Comput. Sci.*, 7(2):1–32, 2011.
19. 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.
20. D. Tishkovsky and R. A. Schmidt. Refinement in the tableau synthesis framework. *arXiv e-Print*, abs/1305.3131, 2013.
21. D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. Mettel2: Towards a tableau prover generation platform. In *Proc. PAAR'12*. EasyChair Proceedings, 2012.
22. D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL^2. In *Proc. JELIA'12*, vol. 7519 of *LNAI*, pp. 492–495. Springer, 2012.
23. TONES. The tones ontology repository, 5 Mar. 2013.
24. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *Proc. IJCAR'06*, LNCS, pp. 292–297. Springer, 2006.