Using Tableau to Decide Expressive Description Logics with Role Negation

Renate A. Schmidt[1] and Dmitry Tishkovsky[1]

School of Computer Science, The University of Manchester
{renate.schmidt,dmitry.tishkovsky}@manchester.ac.uk

Abstract. This paper presents a tableau approach for deciding description logics outside the scope of OWL DL/1.1 and current state-of-the-art tableau-based description logic systems. In particular, we define a sound and complete tableau calculus for the description logic $\mathcal{ALBO}$ and show that it provides a basis for decision procedures for this logic and numerous other description logics with full role negation. $\mathcal{ALBO}$ is the extension of $\mathcal{ALC}$ with the Boolean role operators, inverse of roles, domain and range restriction operators and it includes full support for nominals (individuals). $\mathcal{ALBO}$ is a very expressive description logic which subsumes Boolean modal logic and the two-variable fragment of first-order logic and reasoning in it is NExpTime-complete. An important novelty is the use of a generic, unrestricted blocking rule as a replacement for standard loop checking mechanisms implemented in description logic systems. An implementation of our approach exists in the MetTeL system.

1 Introduction

Mainstream description logics systems and ontology web languages provide a rich supply of concept operators, but there is currently little support for complex role operators. This places significant restrictions on the expressiveness of ontology languages and the utility of systems. For example, in the description logic $\mathcal{ALC}$ (and other popular extensions of $\mathcal{ALC}$) [1] it is possible to define a 'spam filter' as a mechanism for filtering out spam emails (i.e. $x \in \mathsf{spam\text{-}filter}$ iff $x \in \mathsf{mechanism} \wedge \exists y\,.\,(x,y) \in \mathsf{filter\text{-}out} \wedge y \in \mathsf{spam\text{-}email}$), and a 'sound spam filter' as a spam filter which filters out only spam emails (i.e. $x \in \mathsf{sound\text{-}spam\text{-}filter}$ iff $x \in \mathsf{spam\text{-}filter} \wedge \forall y\,.\,(x,y) \in \mathsf{filter\text{-}out} \rightarrow y \in \mathsf{spam\text{-}email}$), by specifying

$$\begin{aligned} \mathsf{spam\text{-}filter} &\stackrel{\text{def}}{=} \mathsf{mechanism} \sqcap \exists \mathsf{filter\text{-}out.spam\text{-}email} \\ (\dagger) \qquad \mathsf{sound\text{-}spam\text{-}filter} &\stackrel{\text{def}}{=} \mathsf{spam\text{-}filter} \sqcap \neg\exists \mathsf{filter\text{-}out.}\neg\mathsf{spam\text{-}email.} \end{aligned}$$

But it is not possible to define a 'complete spam filter' as a spam filter which filters out every spam email, i.e. $x \in \mathsf{complete\text{-}spam\text{-}filter}$ iff $x \in \mathsf{spam\text{-}filter} \wedge \forall y\,.\,y \in \mathsf{spam\text{-}email} \rightarrow (x,y) \in \mathsf{filter\text{-}out}$. This can be expressed by the following.

$$(\ddagger) \qquad \mathsf{complete\text{-}spam\text{-}filter} \stackrel{\text{def}}{=} \mathsf{spam\text{-}filter} \sqcap \neg\exists\neg\mathsf{filter\text{-}out.spam\text{-}email}$$

This uses the role negation operator which is not available in $\mathcal{ALC}$ or description logics that form the basis of OWL DL/1.1. Both (†) and (‡) involve universal

quantification but of a different kind. In (†) it is the image elements of a role that are universally quantified, while in (‡) it is the elements in a concept that are universally quantified. From an applications perspective there is little justification to give preference to one form of universal quantification over the other, since clearly both are useful. (†) expresses the *necessity* of a property and (‡) expresses the *sufficiency* of a property. Natural examples of both kinds of universal quantification can be found in many domains and every-day language.

In this paper we are interested in description logics that allow role negation (and can therefore accommodate examples such as the above), but also provide a range of other role operators not usually supported. In particular, we focus on a description logic, called $\mathcal{ALBO}$, which is an extension of the description logic $\mathcal{ALB}$ [6] with singleton concepts, called nominals in modal logic. $\mathcal{ALB}$ is the extension of $\mathcal{ALC}$, in which concepts and roles form a Boolean algebra, and additional operators include inverse of roles and a domain restriction operator. $\mathcal{ALBO}$ therefore extends $\mathcal{ALC}$ by union of roles, negation of roles, inverse of roles, and domain as well as range restriction. In addition, it provides full support for ABox individuals and singleton concepts.

None of the current state-of-the-art tableau-based description logic systems are able to handle $\mathcal{ALBO}$ (or $\mathcal{ALB}$). Because $\mathcal{ALBO}$ allows full negation of roles, it is out of the scope of OWL DL, OWL 1.1 and most description logic systems including FACT++, KAON2, PELLET, and RACERPRO. A tableau decision procedure for the description logic $\mathcal{ALCQI}b$ which allows for Boolean combinations of 'safe' occurrences of negated roles is discussed in [14]. Safeness essentially implies a 'guardedness' property which is violated by unsafe occurrences of role negation. Description logics with full, i.e. safe and unsafe, role negation can be decided however by translation to first-order logic and first-order resolution theorem provers such as MSPASS, SPASS, E and VAMPIRE. The paper [6] shows that the logic $\mathcal{ALB}$ can be decided by translation to first-order logic and ordered resolution. This result is extended in [4] to $\mathcal{ALB}$ with positive occurrences of composition of roles. $\mathcal{ALBO}$ can be embedded into the two-variable fragment of first-order logic with equality which can be decided with first-order resolution methods [3]. This means that $\mathcal{ALBO}$ is decidable and can be decided using first-order resolution methods.

$\mathcal{ALBO}$ is a very expressive description logic. It subsumes the Boolean modal logic and tense, hybrid versions of Boolean modal logic with the @ operator and nominals. It can also be shown that $\mathcal{ALBO}$ subsumes the two-variable fragment of first-order logic (without equality) [7]. The following constructs and statements can be handled in $\mathcal{ALBO}$.

- Role negation, the universal role, the sufficiency or window operator, the image operator, cross product, and (left and right) cylindrification.
- Role inclusion axioms and role equivalence axioms in the language of $\mathcal{ALBO}$.
- Role assertions in the language of $\mathcal{ALBO}$.
- Boolean combinations of both concept and role inclusion and equivalence axioms.
- Boolean combinations of concept and role assertions, including negated role assertions.

– Disjoint roles, symmetric roles and serial roles. (It is not difficult to extend our method and results to include full equality handling including reflexive roles, identity and diversity roles, and the test operator.)

$\mathcal{ALBO}$ is in fact very close to the brink of undecidability, because we know that adding (negative occurrences of) role composition to $\mathcal{ALB}$ takes us into undecidable territory [12].

Since $\mathcal{ALBO}$ subsumes Boolean modal logic it follows from [9] that the satisfiability problem in $\mathcal{ALBO}$ is NExpTime-hard. In [5] it is shown that satisfiability in the two-variable first-order fragment with equality is NExpTime-complete. It follows therefore that the computational complexity of $\mathcal{ALBO}$-satisfiability is NExpTime-complete.

Few tableau calculi or tableau procedures have been described for description logics with complex role operators, or equivalent dynamic modal logic versions. Ground semantic tableau calculi and tableau decision procedures are presented in [4] for the modal versions of $\mathcal{ALC}(\sqcup, \sqcap, ^{-1})$, i.e. $\mathcal{ALC}$ with role union, role intersection and role inverse. These are extended with the domain restriction operator, to $\mathcal{ALC}(\sqcup, \sqcap, ^{-1}, \restriction)$, in [11]. A semantic tableau decision procedure for $\mathcal{ALC}$ with role intersection, role inverse, role identity and role composition is described in [10]. None of these tableaux make provision for the role negation operator however. In [13] a sound and complete ground semantic tableau calculus is presented for Peirce logic, which is equivalent to the extension of $\mathcal{ALB}$ with role composition and role identity. However the tableau is not terminating because reasoning in Peirce logic is not decidable.

In this paper we develop a tableau approach which can decide description logics with the role negation operator. We present a ground semantic tableau approach which decides the description logic $\mathcal{ALBO}$. The style of tableau is similar to that of [4, 11, 13] but a notable difference is that our tableau calculi operate only on ground labelled *concept* expressions. This makes it easier in principle to implement the calculi as extensions of existing tableau-based description logic systems which can handle singleton concepts.

In order to limit the number of individuals in the tableau we need a mechanism for detecting periodicity in the underlying interpretations (models). *Standard loop checking* mechanisms are based on comparing sets of (labelled or unlabelled) concept expressions such as subset blocking or equality blocking. Instead of using the standard loop checking mechanisms, our approach uses a new inference rule, the *unrestricted blocking* rule, and equality reasoning. Our approach has the following advantages over standard loop checking.

– It is conceptually simple and easy to implement.
– It is universal and does not depend on the notion of a type.
– It is versatile and enables more controlled model construction in a tableau procedure. For instance, it can be used to construct small models for a satisfiable concept, including domain minimal models. Our tableau approach has the further advantage that it constructs real models, whereas the tableau procedures for many OWL DL/1.1 description logics construct only pseudo-

models (which are not always real models but can be completed to real models).
- Our blocking mechanism generalises to other logics, including full first-order logic.
- It can be simulated in first-order logic provers.

The unrestricted blocking rule corresponds to an unrestricted version of the first-order blocking rule invented by [2], simply called the *blocking rule*. The blocking rule is constrained to individuals ℓ and ℓ' such that the individual ℓ is an ancestor of the individual ℓ'. I.e. in the common branch of ℓ and ℓ', the individual ℓ' is obtained from ℓ as a result of a sequence of applications of the existential restriction rule. In this form, the rule can be used to simulate standard loop checking mechanisms such as subset blocking and equality blocking.

The structure of the paper is as follows. The syntax and semantics of $\mathcal{ALBO}$ is defined in Section 2. In Section 3 we define a tableau calculus for $\mathcal{ALBO}$ and prove that it is sound and complete without the unrestricted blocking rule. Section 4 introduces our blocking mechanism and proves soundness, completeness and termination of the extended tableau calculus. This allows us to define general criteria for decision procedures for $\mathcal{ALBO}$ and its sublogics which are discussed in Section 5. We conclude in Section 6.

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

The syntax of $\mathcal{ALBO}$ is defined over the signature $\sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$ of three disjoint alphabets: $\mathsf{O} = \{\ell_0, \ell_1, \ldots\}$ the alphabet of individuals (object names), $\mathsf{C} = \{p_0, p_1, \ldots\}$ the alphabet of concept symbols, and $\mathsf{R} = \{r_0, r_1, \ldots\}$ the alphabet of role symbols. The logical connectives are: $\neg$ (*negation*), $\sqcup$ (*union*), $\exists$ (*existential concept restriction*), $^{-1}$ (role *inverse*), $\restriction$ (*domain restriction*), $\downharpoonright$ (*range restriction*). *Concept expressions* (or *concepts*) and *role expressions* (or *roles*) are defined as follows:

$$C \stackrel{\text{def}}{=} p \mid \{\ell\} \mid \neg C \mid C \sqcup D \mid \exists R.C,$$
$$R \stackrel{\text{def}}{=} r \mid R^{-1} \mid \neg R \mid R \sqcup S \mid R\restriction C \mid R\downharpoonright C.$$

p ranges over the set C, ℓ ranges over O, and r ranges over R. The intersection operator $\sqcap$ on concepts and roles is defined as usual in terms of $\neg$ and $\sqcup$, and the top and bottom concepts are defined by $\top \stackrel{\text{def}}{=} p \sqcup \neg p$ and $\bot \stackrel{\text{def}}{=} p \sqcap \neg p$, respectively, for some concept name p. The universal restriction operator $\forall$ is a dual to the existential restriction operator $\exists$, specified by $\forall R.C \stackrel{\text{def}}{=} \neg\exists R.\neg C$.

Next, we define the semantics of $\mathcal{ALBO}$. A *model* (or an *interpretation*) $\mathcal{I}$ of $\mathcal{ALBO}$ is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, p_0^{\mathcal{I}}, \ldots, \ell_0^{\mathcal{I}}, \ldots, r_0^{\mathcal{I}}, \ldots)$, where $\Delta^{\mathcal{I}}$ is a non-empty set, $p_i^{\mathcal{I}}$ is a subset of $\Delta^{\mathcal{I}}$, $\ell_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and $r_0^{\mathcal{I}}$ is a binary relation over $\Delta^{\mathcal{I}}$. The

semantics of concepts and roles in the model $\mathcal{I}$, i.e. $C^{\mathcal{I}}$ and $R^{\mathcal{I}}$, is specified by:

$$\begin{array}{rlrl}
\{\ell\}^{\mathcal{I}} & \stackrel{\text{def}}{=} \{\ell^{\mathcal{I}}\}, & (R^{-1})^{\mathcal{I}} & \stackrel{\text{def}}{=} (R^{\mathcal{I}})^{-1} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}, \\
(\neg C)^{\mathcal{I}} & \stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, & (\neg R)^{\mathcal{I}} & \stackrel{\text{def}}{=} (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} & \stackrel{\text{def}}{=} C^{\mathcal{I}} \cup D^{\mathcal{I}}, & (R \sqcup S)^{\mathcal{I}} & \stackrel{\text{def}}{=} R^{\mathcal{I}} \cup S^{\mathcal{I}}, \\
\multicolumn{4}{r}{(R\restriction C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \mid x \in C^{\mathcal{I}} \text{ and } (x,y) \in R^{\mathcal{I}}\},} \\
\multicolumn{4}{r}{(R\downharpoonright C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \mid y \in C^{\mathcal{I}} \text{ and } (x,y) \in R^{\mathcal{I}}\},} \\
\multicolumn{4}{r}{(\exists R.C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{x \mid \exists y \in C^{\mathcal{I}}\ (x,y) \in R^{\mathcal{I}}\}.}
\end{array}$$

A *TBox* (respectively *RBox*), is a (finite) set of inclusion statements $C \sqsubseteq D$ (respectively $R \sqsubseteq S$) which are interpreted in any model $\mathcal{I}$ as subset relationships, namely $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (respectively $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$). An *ABox* is a (finite) set of statements of the form $\ell : C$ or $(\ell, \ell') : R$, called concept assertions or role assertions. A *knowledge base* is a tuple (T, R, A) of a TBox T, an RBox R, and an ABox A.

In $\mathcal{ALBO}$ various additional operators can be defined, including:

Top role:	$\nabla \stackrel{\text{def}}{=} r \sqcup \neg r$ (for some role symbol r)
Bottom role:	$\triangle \stackrel{\text{def}}{=} \neg\nabla$
Universal modality:	$\Box C \stackrel{\text{def}}{=} \forall\nabla.C$
Image operator:	$\exists^{-1}R.C \stackrel{\text{def}}{=} \exists R^{-1}.C$
Sufficiency, or window, operator:	$\overline{\forall} R.C \stackrel{\text{def}}{=} \neg\exists\neg R.C$
Left cylindrification:	$C^c \stackrel{\text{def}}{=} \nabla\restriction C$
Right cylindrification:	${}^cC \stackrel{\text{def}}{=} \nabla\downharpoonright C$
Cross product:	$C \times D \stackrel{\text{def}}{=} C^c \sqcap {}^cD$

Concept assertions can be expressed as concept expressions as follows: $\ell : C \stackrel{\text{def}}{=} \exists\nabla.(\{\ell\} \sqcap C)$. (It is clear that $(\ell : C)^{\mathcal{I}} \neq \emptyset$ iff $\ell^{\mathcal{I}} \in C^{\mathcal{I}}$ in every model $\mathcal{I}$.) A role assertion $(\ell, \ell') : R$ can be expressed as a concept assertion, namely $(\ell, \ell') : R \stackrel{\text{def}}{=} \ell : \exists R.\{\ell'\}$, or by the above as a concept expression. We refer to $\ell : \exists R.\{\ell'\}$, or a role assertion, as a *link* (between the individuals ℓ and ℓ'). In addition, concept and role inclusion axioms are definable as concept expressions.

$$\begin{array}{rl}
C \sqsubseteq D & \stackrel{\text{def}}{=} \forall\nabla.(\neg C \sqcup D) \\
R \sqsubseteq S & \stackrel{\text{def}}{=} \forall\nabla.\forall\neg(\neg R \sqcup S).\bot
\end{array}$$

Thus, Boolean combinations of inclusion and assertion statements of concepts and roles are also expressible in $\mathcal{ALBO}$ as the corresponding Boolean combinations of the concepts which represent these statements. As usual, concept satisfiability in $\mathcal{ALBO}$ with respect to any knowledge base can be reduced to concept satisfiability with respect to a knowledge base where all TBox, RBox, and ABox are empty. Without loss of generality we therefore focus on the problem of concept satisfiability in $\mathcal{ALBO}$.

Often description logics are required to satisfy the unique name assumption. We do not assume it for $\mathcal{ALBO}$. This is inconsequential, because the unique name assumption can be enforced by disjointness statements of the form $\{\ell\} \sqcap \{\ell'\} \sqsubseteq \bot$ for every distinct pair of nominals that occur in the given knowledge base.

Above we defined the cylindrification operators and cross product in terms of the domain and range restriction operators. In fact, each of the operators in $\{\uparrow, \downarrow, \cdot^c, {}^c\cdot, \times\}$ are interdefinable. Thus, we could have defined $\mathcal{ALBO}$ as an extension of $\mathcal{ALCO}(\neg, \sqcup, {}^{-1})$ with one of these operators. It can be shown that regardless as to which of these is used to define $\mathcal{ALBO}$, problems in $\mathcal{ALBO}$ are linearly reducible to problems in $\mathcal{ALCO}(\neg, \sqcup, {}^{-1})$. For instance, suppose $\mathcal{ALBO}$ is defined as the extension of $\mathcal{ALCO}(\neg, \sqcup, {}^{-1})$ with left cylindrification. The satisfiability of a concept C, say, in $\mathcal{ALBO}$ can then be encoded in $\mathcal{ALCO}(\neg, \sqcup, {}^{-1})$ by replacing all occurrences of D^c in C by a new role symbol q_{D^c} uniquely associated with D^c and adding the definitions $\neg D \sqsubseteq \forall q_{D^c}.\bot$ and $D \sqsubseteq \neg\exists\neg q_{D^c}.\top$ to the knowledge base. In a similar way, problems involving the other operators from $\{\uparrow, \downarrow, \cdot^c, {}^c\cdot, \times\}$ can be linearly encoded in $\mathcal{ALCO}(\neg, \sqcup, {}^{-1})$. We leave it to the reader to make sharper observations concerning the definability of these operators in description logics.

Let us give our first technical result. The following theorem can be proved using a filtration argument.

Theorem 1 (Finite Model Property). *$\mathcal{ALBO}$ has the finite model property, i.e., if a concept C is satisfiable, then it has a finite model.*

3 Tableau calculus

Let T denote a tableau calculus and C a concept. We denote by $T(C)$ a finished tableau built using the rules of the calculus T starting with the concept C as input. I.e. we assume that all branches in the tableau are expanded and all applicable rules of T have been applied in $T(C)$. As usual we assume that all the rules of the calculus are applied *non-deterministically, to a tableau.* A branch of a tableau is *closed* if a contradiction has been derived in this branch, otherwise the branch is called *open.* The tableau $T(C)$ is *closed* if all its branches are closed and $T(C)$ is *open* otherwise. We say that T is *terminating* (for satisfiability) iff for every concept C either $T(C)$ is finite whenever $T(C)$ is closed or $T(C)$ has a finite open branch if $T(C)$ is open. T is *sound* iff C is unsatisfiable whenever $T(C)$ is closed for all concepts C. T is *complete* iff for any concept C, C is satisfiable (has a model) whenever $T(C)$ is open.

Let $T_{\mathcal{ALBO}}$ be the tableau calculus consisting of the rules listed in Figure 1. Inference steps are performed in the usual way. A rule is applied to a set of expressions (often just one expression) in a branch of a tableau, if the expressions are instances of the premises of the rule. Then, in the case of a non-branching rule, the corresponding instances of the conclusions of the rule are added to the branch. A branching rule splits the branch into several branches (here two) and adds the corresponding instances of the conclusions to each branch.

Rules for $\mathcal{ALCO}$:

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

$$(\neg\sqcup)\colon \frac{\ell : \neg(C \sqcup D)}{\ell : \neg C, \quad \ell : \neg D} \qquad (\sqcup)\colon \frac{\ell : (C \sqcup D)}{\ell : C \mid \ell : D}$$

$$(\mathsf{sym})\colon \frac{\ell : \{\ell'\}}{\ell' : \{\ell\}} \qquad (\neg\mathsf{sym})\colon \frac{\ell : \neg\{\ell'\}}{\ell' : \neg\{\ell\}} \qquad (\mathsf{mon})\colon \frac{\ell : \{\ell'\}, \quad \ell' : C}{\ell : C} \qquad (\mathsf{refl})\colon \frac{\ell : C}{\ell : \{\ell\}}$$

$$(\exists)\colon \frac{\ell : \exists R.C}{\ell : \exists R.\{\ell'\}, \quad \ell' : C}(\ell' \text{ is new}) \qquad (\neg\exists)\colon \frac{\ell : \neg\exists R.C, \quad \ell : \exists R.\{\ell'\}}{\ell' : \neg C}$$

$$(\mathsf{bridge})\colon \frac{\ell : \exists R.\{\ell'\}, \quad \ell' : \{\ell''\}}{\ell : \exists R.\{\ell''\}}$$

Rules for complex roles:

$$(\exists\sqcup)\colon \frac{\ell : \exists(R \sqcup S).\{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell : \exists S.\{\ell'\}} \qquad (\neg\exists\sqcup)\colon \frac{\ell : \neg\exists(R \sqcup S).C}{\ell : \neg\exists R.C, \quad \ell : \neg\exists S.C}$$

$$(\exists^{-1})\colon \frac{\ell : \exists R^{-1}.\{\ell'\}}{\ell' : \exists R.\{\ell\}} \qquad (\neg\exists^{-1})\colon \frac{\ell : \neg\exists R^{-1}.C, \quad \ell' : \exists R.\{\ell\}}{\ell' : \neg C}$$

$$(\exists\restriction)\colon \frac{\ell : \exists(R\restriction C).\{\ell'\}}{\ell : C, \quad \ell : \exists R.\{\ell'\}} \qquad (\neg\exists\restriction)\colon \frac{\ell : \neg\exists(R\restriction C).D}{\ell : \neg C \mid \ell : \neg\exists R.D}$$

$$(\exists\downarrow)\colon \frac{\ell : \exists(R\downarrow C).\{\ell'\}}{\ell' : C, \quad \ell : \exists R.\{\ell'\}} \qquad (\neg\exists\downarrow)\colon \frac{\ell : \neg\exists(R\downarrow C).D}{\ell : \neg\exists R.\neg(\neg C \sqcup \neg D)}$$

$$(\exists\neg)\colon \frac{\ell : \exists\neg R.\{\ell'\}}{\ell : \neg\exists R.\{\ell'\}} \qquad (\neg\exists\neg)\colon \frac{\ell : \neg\exists\neg R.C, \quad \ell' : \{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell' : \neg C}$$

Fig. 1. Tableau calculus $T_{\mathcal{ALBO}}$ for $\mathcal{ALBO}$.

The first group of rules are standard for $\mathcal{ALC}$ and reasoning with individuals. The $(\bot)$ rule is the closure rule. The $(\neg\neg)$ rule removes occurrences of double concept negation. (The rule is superfluous if double negations are eliminated using on-the-fly rewrite rules.) The $(\sqcup)$ and $(\neg\sqcup)$ rules are standard rules for handling concept disjunctions. The (sym), (mon), (refl), and (bridge) rules are the equality rules for individuals, which are familiar from hybrid logic tableau systems, and can be viewed as versions of standard rules for first-order equality. The (refl) rule is formulated perhaps a bit unusually, but the purpose of the premise is to ensure that the rule is realised only for individuals actually occurring in the branch. The $(\neg\mathsf{sym})$ rule is needed to ensure that any negated singleton concept will eventually appear as a label in a concept assertion. As usual, and in accordance with the semantics of the existential restriction operator, for every existentially restricted concept the $(\exists)$ rule creates a new individual with this concept and adds a link to the new individual. It is the only rule which generates new individuals in the calculus. The $(\neg\exists)$ rule is equivalent to the standard propagation rule for universally restricted concept expressions.

The rules in the second group are rules for decomposing complex role expressions. They can be divided into two subgroups: rules for positive existential role occurrences and rules for negated existential role occurrences (in the left and right columns, respectively). Due to the presence of the ($\exists$) rule, the rules for positive existential roles can be restricted to role assertions. (On the side we note that the rule ($\exists\neg$) can be replaced by this closure rule: $\ell : \exists\neg R.\{\ell'\}, \ell : \exists R.\{\ell'\}/\bot$.) Among the rules for negated existential roles, the ($\neg\exists^{-1}$) rule and and the ($\neg\exists\neg$) rule are special. The ($\neg\exists^{-1}$) rule allows the backward propagation of concept expressions along inverted links (ancestor links). The ($\neg\exists\neg$) rule is the rule for the sufficiency operator. It expands a universally restricted concept in which the role is negated according to the semantics: $x \in (\neg\exists\neg R.C)^{\mathcal{I}}$ iff $\forall y((x,y) \in R^{\mathcal{I}} \vee y \in (\neg C)^{\mathcal{I}})$. That is, ℓ' is implicitly quantified by a universal quantifier. The effect of the second premise, $\ell' : \{\ell'\}$, is to instantiate ℓ' with individuals that occur in the branch. The remaining rules in this subgroup are based on obvious logical equivalences in $\mathcal{ALBO}$.

Tableau rules which do not produce new individuals are called *type-completing* rules. In the case of $T_{\mathcal{ALBO}}$, with the exception of the ($\exists$) rule, all rules are type-completing.

Now, given an input concept C, a tableau derivation is constructed as follows. First, preprocessing is performed which pushes the role inverse operators toward atomic concepts by exhaustively applying the following role equivalences from left to right.

$$(\neg R)^{-1} = \neg(R^{-1}), \qquad (R \sqcup S)^{-1} = R^{-1} \sqcup S^{-1},$$
$$(R\restriction C)^{-1} = R^{-1}\downarrow C, \qquad (R\downarrow C)^{-1} = R^{-1}\restriction C, \qquad (R^{-1})^{-1} = R.$$

Next, the preprocessed input concept C is tagged with a fresh individual name ℓ that does not occur in C. Then we build a complete tableau $T_{\mathcal{ALBO}}(C)$ by applying the rules of $T_{\mathcal{ALBO}}$ to the concept assertion $\ell : C$ as described above. It is however important to note that $\ell : C$ and all labelled expressions and assertions really denote concept expressions.

We turn to proving soundness and completeness of the calculus. Because every rule preserves the satisfiability of concept assertions, it is easy to see that the calculus $T_{\mathcal{ALBO}}$ is sound for $\mathcal{ALBO}$.

For proving completeness, suppose that a tableau $T_{\mathcal{ALBO}}(C)$ for the given concept C is open, i.e. it contains an open branch $\mathcal{B}$. We construct a model $\mathcal{I}$ for the satisfiability of C as follows. By definition, let $\ell \sim \ell' \stackrel{\text{def}}{\Longleftrightarrow} \ell : \{\ell'\} \in \mathcal{B}$. It is clear that the rules (sym), (mon), and (refl) ensure that $\sim$ is an equivalence relation on individuals. The equivalence class $\|\ell\|$ of a representative ℓ is defined by: $\|\ell\| \stackrel{\text{def}}{=} \{\ell' \mid \ell \sim \ell'\}$. We set

$$\Delta^{\mathcal{I}} \stackrel{\text{def}}{=} \{\|\ell\| \mid \ell : \{\ell\} \in \mathcal{B}\}, \qquad r^{\mathcal{I}} \stackrel{\text{def}}{=} \{(\|\ell\|, \|\ell'\|) \mid \ell : \exists r.\{\ell'\} \in \mathcal{B}\},$$
$$p^{\mathcal{I}} \stackrel{\text{def}}{=} \{\|\ell\| \mid \ell : p \in \mathcal{B}\}, \qquad \ell^{\mathcal{I}} \stackrel{\text{def}}{=} \begin{cases} \|\ell\|, & \text{if } \ell : \{\ell\} \in \mathcal{B}, \\ \|\ell'\| & \text{for some } \|\ell'\| \in \Delta^{\mathcal{I}}, \text{ otherwise.} \end{cases}$$

1. TBox $= \{\exists r.p\}$ ..... given
2. $\ell_0 : \exists r.p$ TBox,1
3. $\ell_1 : p$ ..... $(\exists)$,2
4. $\ell_0 : \exists r.\{\ell_1\}$ ..... $(\exists)$,2
5. $\ell_1 : \exists r.p$ TBox,1
6. $\ell_0 \not\sim \ell_1$ ..... Loop checking: p is a difference
7. $\ell_2 : p$ ..... $(\exists)$,5
8. $\ell_1 : \exists r.\{\ell_2\}$ ..... $(\exists)$,5
9. $\ell_2 : \exists r.p$ TBox,1
10. $\ell_1 \sim \ell_2$ ..... Loop checking

Fig. 2. Standard loop checking mechanism in $\mathcal{ALCO}$.

The rules (sym), (mon), (refl), and (bridge) ensure that the definition of $\mathcal{I}$ does not depend on representatives of the equivalence classes.

The following lemma is proved by induction over the structure C.

Lemma 1. *(1) If $\ell : D \in \mathcal{B}$ then $\|\ell\| \in D^{\mathcal{I}}$ for any concept D.*
(2) For every role R and every concept D
(2a) $\ell : \exists R.\{\ell'\} \in \mathcal{B}$ implies $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$,
(2b) if $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$ and $\ell : \neg\exists R.D \in \mathcal{B}$ then $\ell' : \neg D \in \mathcal{B}$.

A consequence of this lemma is completeness of the tableau calculus.

Theorem 2. *$T_{\mathcal{ALBO}}$ is a sound and complete tableau calculus for satisfiability in $\mathcal{ALBO}$.*

4 Blocking

There are satisfiable concepts which result in an infinite $T_{\mathcal{ALBO}}$-tableau, where all open branches are infinite. The concept $\neg\exists(s \sqcup \neg s).\neg\exists r.p$ is such an example. Indeed, since the prefix $\neg\exists(s \sqcup \neg s).\neg$ is equivalent to the universal modality, the concept $\ell : \exists r.p$ is propagated to every individual ℓ in every branch of the tableau. The concept $\ell : \exists r.p$ itself, each time triggers the creation of a new individual with the $(\exists)$ rule. Thus, any branch of the tableau contains infinitely many individuals. The branches have however a regular structure that can be detected with loop detection or blocking mechanisms.

Observe that satisfiability of the concept $\neg\exists(s \sqcup \neg s).\neg\exists r.p$ corresponds to satisfiability of the TBox $\{\exists r.p\}$ in the description logic $\mathcal{ALCO}$. Figure 2 demonstrates how *standard loop checking* (subset blocking) for the description logic $\mathcal{ALCO}$ with general TBoxes detects a loop in this example. (In the figure each line in the derivation is numbered on the left. The rule applied and the number of the premise(s) to which it was applied to produce the labelled concept expression (assertion) in each line is specified on the right.) Loop checking tests are performed after all the type-completing rules have been applied to all concept expressions labelled with a specific individual relative to an ancestor individual. In the tableau in Figure 2, two loop checking tests are performed, namely in step 6 and step 10. Take step 10. All the type-completing rules have been applied to all concept expressions of the form $\ell_2 : C$ and $\ell_1 : C$. This means the $\exists r.p$-types of the individuals ℓ_1 and ℓ_2, of the partial model constructed so

1. $\ell_0 : \exists r.p$ ..... given
2. $\ell_0 : \exists s.p$ ..... given
3. $\ell_0 : \exists t.\neg\exists t.\exists s^{-1}.(p \sqcup \neg p)$ ..... given
4. $\ell_0 : \neg\exists t.\exists\neg t.\neg\exists s^{-1}.(p \sqcup \neg p)$ ... given
5. $\ell_1 : p$ ..... $(\exists)$,1
6. $\ell_0 : \exists r.\{\ell_1\}$ ..... $(\exists)$,1
7. $\ell_1 : \{\ell_1\}$ ..... (refl),5
8. $\ell_2 : p$ ..... $(\exists)$,2
9. $\ell_0 : \exists s.\{\ell_2\}$ ..... $(\exists)$,2
10. $\ell_2 : \{\ell_2\}$ ..... (refl),8
11. $\ell_3 : \neg\exists t.\exists s^{-1}.(p \sqcup \neg p)$ ..... $(\exists)$,3
12. $\ell_0 : \exists t.\{\ell_3\}$ ..... $(\exists)$,3
13. $\ell_3 : \neg\exists\neg t.\neg\exists s^{-1}.(p \sqcup \neg p)$ .. $(\neg\exists)$,4,12
14. ▶$\ell_3 : \exists t.\{\ell_2\}$ ..... $(\neg\exists\neg)$,13,10
15. $\ell_2 : \neg\exists s^{-1}.(p \sqcup \neg p)$ ..... $(\neg\exists)$,11,14
16. $\ell_0 : \neg(p \sqcup \neg p)$ ..... $(\neg\exists^{-1})$,15,9
17. Unsatisfiable ..... after a few steps
18. ▶$\ell_2 : \neg\neg\exists s^{-1}.(p \sqcup \neg p)$ .. $(\neg\exists\neg)$,13,10
19. $\ell_2 : \exists s^{-1}.(p \sqcup \neg p)$ ..... $(\neg\neg)$,18
20. ▶$\ell_3 : \exists t.\{\ell_1\}$ ..... $(\neg\exists\neg)$,13,7
21. $\ell_1 : \neg\exists s^{-1}.(p \sqcup \neg p)$ ... $(\neg\exists)$,11,20
... .
22. ▶$\ell_1 : \neg\neg\exists s^{-1}.(p \sqcup \neg p)$. $(\neg\exists\neg)$,13,7
... .

Fig. 3. Global effect of the introduction of a new individual

far, are complete. Comparison of the types shows that they coincide and, consequently, the individuals ℓ_1 and ℓ_2 can be identified. At step 6 the $\exists r.p$-types of ℓ_0 and ℓ_1 are different because $\ell_0 : p$ is not present in the branch. The derivation therefore cannot yet stop, but does in step 10.

This example illustrates the simplest form of standard loop checking used in description and modal logic tableau procedures. This form of loop checking is too simple to handle role negation though. The problem is that the introduction of a new individual in a tableau has, in general, a global effect in the provisional model constructed so far. This global effect is illustrated by the example in Figure 3. (The black triangles in the figure denote branching points in the derivation. A branch expansion after a branching point is indicated by appropriate indentation.) At step 10 none of the type-completing rules need to be applied to concepts labelled with ℓ_1 and ℓ_2. Although at this point ℓ_1 and ℓ_2 are labels of the same subconcepts of the given concepts, we cannot make them equal. The reason is that at step 11 a new individual is introduced which causes a few applications of the $(\neg\exists)$ rule, and as a result, at step 21, the types of ℓ_1 and ℓ_2 are now distinguished by the concept $\exists s^{-1}.(p \sqcup \neg p)$.

The examples illustrate that the reason for non-termination of $T_{\mathcal{ALBO}}$ is the possible infinite generation of labels. The following lemma holds, where $\#^{\exists}(\mathcal{B})$ denotes the number of applications of the $(\exists)$ rule in a branch $\mathcal{B}$.

Lemma 2. *If $\#^{\exists}(\mathcal{B})$ is finite then $\mathcal{B}$ is finite.*

In order to turn the calculus $T_{\mathcal{ALBO}}$ into a terminating calculus for $\mathcal{ALBO}$, we introduce a new, different approach to blocking.

Let $<$ be an ordering on individuals in the branch which is a linear extension of the order of introduction of the individuals during the derivation. I.e. let $\ell < \ell'$, whenever the first appearance of individual ℓ' in the branch is strictly later than the first appearance of individual ℓ. We add the following rule, called the *unrestricted blocking* rule, to the calculus.

$$(\mathsf{ub}):\ \frac{\ell : \{\ell\},\quad \ell' : \{\ell'\}}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Moreover, we require that the following conditions hold.

(c1) Any rule is applied at most once to the same set of premises.
(c2) The ($\exists$) rule is applied only to expressions of the form $\ell : \exists R.C$, when C is not a singleton, i.e. $C \neq \{\ell''\}$ for some individual ℓ''.
(c3) If $\ell : \{\ell'\}$ appears in a branch and $\ell < \ell'$ then all further applications of the ($\exists$) rule to expressions of the form $\ell' : \exists R.C$ are not performed within the branch.
(c4) In every open branch there is some node from which point onwards before any application of the ($\exists$) rule all possible applications of the (ub) rule have been performed.

We use the notation $T_{\mathcal{ALBO}}(\text{ub})$ for the extension of $T_{\mathcal{ALBO}}$ with this rule and this blocking mechanism.

The blocking requirements (c1)–(c4) are sound in the sense that they cannot cause an open branch to become closed. The (ub) rule is sound in the usual sense. Thus, the following theorem holds.

Theorem 3. *$T_{\mathcal{ALBO}}(\text{ub})$ is a sound and complete tableau calculus for $\mathcal{ALBO}$.*

Let $\mathcal{B}$ be the *leftmost open branch with respect to the rule* (ub) in the $T_{\mathcal{ALBO}}(\text{ub})$ tableau for a given concept C. Assume that $\mathcal{I}$ is a model constructed from $\mathcal{B}$ as in Section 3.

It can be shown that the tableau procedure mimics the construction of a finite model used in the proof of Theorem 1. Thus the following lemma holds.

Lemma 3. *$\Delta^{\mathcal{I}}$ is finite.*

For every $\|\ell\| \in \Delta^{\mathcal{I}}$, let $\#^{\exists}(\|\ell\|)$ denote the number of applications of the ($\exists$) rule to concepts of the form $\ell' : \exists R.D$ with $\ell' \in \|\ell\|$.

Lemma 4. *(1) $\#^{\exists}(\|\ell\|)$ is finite for every $\|\ell\| \in \Delta^{\mathcal{I}}$.*
(2) $\#^{\exists}(\mathcal{B})$ is finite.

Corollary 1. *If the leftmost branch with respect to the rule* (ub) *in a $T_{\mathcal{ALBO}}(\text{ub})$ tableau is open then the branch is finite.*

The termination theorem is an immediate consequence.

Theorem 4 (Termination). *$T_{\mathcal{ALBO}}(\text{ub})$ is a terminating tableau calculus for satisfiability in $\mathcal{ALBO}$.*

Notice that condition (c4) is essential for ensuring termination of a $T_{\mathcal{ALBO}}(\text{ub})$ derivation. Indeed, it easy to see that without (c4) the $T_{\mathcal{ALBO}}(\text{ub})$ tableau for the concept $\neg(\exists(s \sqcup \neg s).\neg\exists r.p \sqcup \exists(s \sqcup \neg s).\neg\exists r.\neg p)$ would not terminate because new individuals are generated more often than their equality check is performed via the rule (ub).

1. $\ell_0 : C$ ..... given
2. $\ell_0 : \{\ell_0\}$ ..... (refl),1
3. $\ell_0 : \neg\exists(s \sqcup \neg s).\neg\exists r.p$ ..... $(\neg\sqcup)$,1
4. $\ell_0 : \neg\neg\exists t.\neg\exists r.p$ ..... $(\neg\sqcup)$,1
5. $\ell_0 : \exists t.\neg\exists r.p$ ..... $(\neg\neg)$,4
6. $\ell_0 : \neg\exists s.\neg\exists r.p$ ..... $(\neg\exists\sqcup)$,3
7. $\ell_0 : \neg\exists\neg s.\neg\exists r.p$ ..... $(\neg\exists\sqcup)$,3
8. $\blacktriangleright\ell_0 : \exists s.\{\ell_0\}$ ..... $(\neg\exists\neg)$,7,2
9. $\ell_0 : \neg\neg\exists r.p$ ..... $(\neg\exists)$,8,6
10. $\ell_0 : \exists r.p$ ..... $(\neg\neg)$,9
11. $\ell_1 : p$ ..... $(\exists)$,10
12. $\ell_0 : \exists r.\{\ell_1\}$ ..... $(\exists)$,10
13. $\ell_1 : \{\ell_1\}$ ..... (refl),11
14. $\blacktriangleright\ell_0 : \exists s.\{\ell_1\}$ ..... $(\neg\exists\neg)$,7,13
15. $\ell_1 : \neg\neg\exists r.p$ ..... $(\neg\exists)$,14,6
16. $\ell_1 : \exists r.p$ ..... $(\neg\neg)$,15
17. $\blacktriangleright\ell_0 : \{\ell_1\}$ ..... (ub)
18. $\ell_2 : \neg\exists r.p$ ..... $(\exists)$,5
19. $\ell_0 : \exists t.\{\ell_2\}$ ..... $(\exists)$,5
20. $\ell_2 : \{\ell_2\}$ ..... (refl),18
21. $\blacktriangleright\ell_0 : \exists s.\{\ell_2\}$ ..... $(\neg\exists\neg)$,7,20
22. $\ell_2 : \neg\neg\exists r.p$ ..... $(\neg\exists)$,6,21
23. Unsatisfiable ..... $(\bot)$,18,22
24. $\blacktriangleright\ell_2 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,7,20
25. Unsatisfiable ..... $(\bot)$,18,24
26. $\blacktriangleright\ell_0 : \neg\{\ell_1\}$ ..... (ub)
27. $\ell_2 : p$ ..... $(\exists)$,16
28. $\ell_1 : \exists r.\{\ell_2\}$ ..... $(\exists)$,16
29. $\ell_2 : \{\ell_2\}$ ..... (refl),27
30. $\blacktriangleright\ell_0 : \exists s.\{\ell_2\}$ ..... $(\neg\exists\neg)$,7,29
31. $\ell_2 : \neg\neg\exists r.p$ ..... $(\neg\exists)$,30,6
32. $\ell_2 : \exists r.p$ ..... $(\neg\neg)$,31
33. **Non-terminating** .....
..... Repetition of 16–32
34. $\blacktriangleright\ell_2 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,7,29
35. ..... Similarly to 30–33
36. $\blacktriangleright\ell_1 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,7,13
37. ..... Similarly to 14–35
38. $\blacktriangleright\ell_0 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,7,2
39. ..... Similarly to 8–37

Fig. 4. An infinite derivation, due to unfair selection of concepts

5 Decision procedures

When turning the presented calculus $T_{\mathcal{ALBO}}$(ub) into a deterministic decision procedure it is crucial that this is done in a *fair* way. A procedure is fair if, when an inference is possible forever, then it is performed eventually. This means a deterministic tableau procedure based on $T_{\mathcal{ALBO}}$(ub) may not defer the use of an applicable rule indefinitely. Note that we understand fairness in a 'global' sense. That is, a tableau procedure has to be fair not only to expressions in a particular branch but to expressions in *all* branches of a tableau. In other words, a procedure is fair if it is makes both the branch selection, and the selection of expressions to which to apply a rule to, in a fair way.

Theorem 5. *Any fair tableau procedure based on $T_{\mathcal{ALBO}}$(ub) is a decision procedure for $\mathcal{ALBO}$ and all its sublogics.*

Note that we do not assume that the branches are expanded in a depth-first left-to-right order. Nevertheless, it also follows from our results that:

Theorem 6. *Any fair tableau procedure based on $T_{\mathcal{ALBO}}$(ub) which uses a depth-first and left-to-right strategy, with respect to branch selection of the* (ub) *rule, is a decision procedure for $\mathcal{ALBO}$ and all its sublogics.*

To illustrate the importance of fairness we give two examples. The concept

$$C \stackrel{\text{def}}{=} \neg\left(\exists(s \sqcup \neg s).\neg\exists r.p \sqcup \neg\exists t.\neg\exists r.p\right)$$

1. $\ell_0 : \neg\exists(s \sqcup \neg s).\neg\exists r.p$ ..... given
2. $\ell_0 : \{\ell_0\}$ ..... (refl),1
3. $\ell_0 : \neg\exists s.\neg\exists r.p$ ..... $(\neg\exists\sqcup)$,1
4. $\ell_0 : \neg\exists\neg s.\neg\exists r.p$ ..... $(\neg\exists\sqcup)$,1
5. ▶$\ell_0 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,4,2
6. $\ell_0 : \exists r.p$ ..... $(\neg\neg)$,5
7. $\ell_1 : p$ ..... $(\exists)$,6
8. $\ell_0 : \exists r.\{\ell_1\}$ ..... $(\exists)$,6
9. $\ell_1 : \{\ell_1\}$ ..... (refl),7
10. ▶$\ell_1 : \neg\neg\exists r.p$ ..... $(\neg\exists\neg)$,4,9
11. $\ell_1 : \exists r.p$ ..... $(\neg\neg)$,5
12. ▶$\ell_0 : \neg\{\ell_1\}$ ..... (ub),2,9
13. $\ell_2 : p$ ..... $(\exists)$,11
14. $\ell_1 : \exists r.\{\ell_2\}$ ..... $(\exists)$,11
15. **Non-terminating** .....
..... Repetition of 7–14
16. ▶$\ell_0 : \{\ell_1\}$ ..... (ub),2,9
17. ..... Never expanded
18. ▶$\ell_0 : \exists s.\{\ell_1\}$ ..... $(\neg\exists\neg)$,4,9
19. ..... Never expanded
20. ▶$\ell_0 : \exists s.\{\ell_0\}$ ..... $(\neg\exists\neg)$,4,2
21. ..... Never expanded

Fig. 5. An infinite derivation, due to unfair selection of branches

is not satisfiable. Figure 4 gives a depth-first left-to-right derivation which is unfair and does not terminate. It can be seen that the derivation is infinite because the application of the $(\exists)$ rule to $\ell_0 : \exists t.\neg\exists r.p$ is deferred forever and, consequently, a contradiction is not found. This illustrates the importance of fairness for (refutational) completeness.

The next example illustrates the importance of fairness for decidability. The concept $\neg\exists(s\sqcup\neg s).\neg\exists r.p$ is satisfiable. The derivation in Figure 5 is obtained with a depth-first *right-to-left* strategy. However, the repeated selection of the right branch at (ub) choice points causes all the individuals in the branch to be pairwise non-equal. The concept $\ell : \exists r.p$ re-appears repeatedly, for every individual ℓ in the branch. This triggers the repeated generation of a new individual by the $(\exists)$ rule, resulting in an infinite derivation. This strategy is unfair because all branches except for the rightmost branch get ignored.

In an implemented prover, optimisations, good heuristics and clever backtracking techniques are important. The standard optimisations such as simplification, backjumping, dynamic backtracking, different heuristics for branch selection and rule selection, lemma generation, et cetera, are all compatible with the presented calculi and procedures. An obvious simplification, for example, is the on-the-fly removal of double negations from concepts, and especially from roles, as this reduces a number of applications of the $(\neg\exists\neg)$ rule.

Since the presented tableaux operate only on ground labelled *concept* expressions, they can in principle be implemented as extensions of existing tableau-based description logic systems which can handle singleton concepts. We have implemented the unrestricted blocking rule as a plug-in to the MetTeL tableau prover [8], and tests with various description logics are encouraging.

6 Conclusion

We have presented a new, general tableau approach for deciding expressive description logics with complex role operators, including 'non-safe' occurrences of role negation. The tableau decision procedures found in the description logic

literature, and implemented in existing tableau-based description logic systems, can handle a large class of description logics, but cannot currently handle description logics with full role negation such as $\mathcal{ALB}$ or $\mathcal{ALBO}$. The present paper closes this gap. An important novelty of our approach is the use of a blocking mechanism based on small inference steps rather than 'big' tests performed on sets of expressions or assertions which are often tailored toward specific logics. Our techniques are versatile and are not limited to $\mathcal{ALBO}$ or its sublogics, but carry over to all description logics and also other logics including first-order logic.

Acknowledgements. We thank the referees for valuable comments. The work is supported by EPSRC research grant EP/D056152/1.

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *Description Logic Handbook*. Cambridge Univ. Press, 2003.
2. P. Baumgartner and R. A. Schmidt. Blocking and other enhancements for bottom-up model generation methods. In *Proc. IJCAR'06*, vol. 4130 of *LNAI*, pp. 125–139. Springer, 2006.
3. H. De Nivelle and I. Pratt-Hartmann. A resolution-based decision procedure for the two-variable fragment with equality. In *Proc. IJCAR'01*, vol. 2083 of *LNAI*, pp. 211–225. Springer, 2001.
4. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Log. J. IGPL*, 8(3):265–292, 2000.
5. E. Grädel, P. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. *Bull. Sect. Log.*, 3:53–69, 1997.
6. U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, vol. 1761 of *LNAI*, pp. 191–205. Springer, 2000.
7. U. Hustadt, R. A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *J. Relat. Methods Comput. Sci.*, 1:251–276, 2004.
8. U. Hustadt, D. Tishkovsky, F. Wolter, and M. Zakharyaschev. Automated reasoning about metric and topology (System description). In *Proc. JELIA'06*, vol. 4160 of *LNAI*, pp. 490–493. Springer, 2006.
9. C. Lutz and U. Sattler. The complexity of reasoning with Boolean modal logics. In *Advances in Modal Logics, Vol. 3*. CSLI Publ., 2002.
10. F. Massacci. Decision procedures for expressive description logics with intersection, composition, converse of roles and role identity. In *Proc. IJCAI'01*, pp. 193–198. Morgan Kaufmann, 2001.
11. R. A. Schmidt. Developing modal tableaux and resolution methods via first-order resolution. In *Advances in Modal Logic, Vol. 6*, pp. 1–26. College Publ., 2006.
12. R. A. Schmidt and U. Hustadt. Mechanised reasoning and model generation for extended modal logics. In *Theory and Applications of Relational Structures as Knowledge Instruments*, vol. 2929 of *LNCS*, pp. 38–67. Springer, 2003.
13. R. A. Schmidt, E. Orlowska, and U. Hustadt. Two proof systems for Peirce algebras. In *Proc. RelMiCS'03*, vol. 3051 of *LNCS*, pp. 238–251. Springer, 2004.
14. S. Tobies. *Decidability Results and Practical Algorithms for Logics in Knowledge Representation*. PhD dissertation, RWTH Aachen, 2001.