Using Tableau to Decide Description Logics with Full Role Negation and Identity

RENATE A. SCHMIDT and DMITRY TISHKOVSKY, The University of Manchester, UK

This paper presents a tableau approach for deciding expressive description logics with full role negation and role identity. We consider the description logic $\mathcal{ALBO}^{\mathsf{id}}$, which is $\mathcal{ALC}$ extended with the Boolean role operators, inverse of roles, the identity role, and includes full support for individuals and singleton concepts. $\mathcal{ALBO}^{\mathsf{id}}$ is expressively equivalent to the two-variable fragment of first-order logic with equality and subsumes Boolean modal logic. In this paper we define a sound, complete and terminating tableau calculus for $\mathcal{ALBO}^{\mathsf{id}}$ that provides the basis for decision procedures for this logic and all its sublogics. An important novelty of our approach is the use of a generic unrestricted blocking mechanism. Unrestricted blocking is based on equality reasoning and a conceptually simple rule, which performs case distinctions over the identity of individuals. The blocking mechanism ties the proof of termination of tableau derivations to the finite model property of $\mathcal{ALBO}^{\mathsf{id}}$.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic**]: Computational logic; Mechanical Theorem Proving; Modal Logic; Proof Theory; I.2.3 [**Deduction and Theorem Proving**]: Deduction; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal Logic

General Terms: Algorithms; Theory; Verification

Additional Key Words and Phrases: Boolean modal logic; blocking; completeness; complexity; decidability; description logic; identity role; role negation; tableau-based reasoning

ACM Reference Format:
Renate A. Schmidt and Dmitry Tishkovsky. 2013. Using Tableau to Decide Description Logics with Full Role Negation and Identity , , Article (January), 30 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Mainstream description logic languages and ontology web languages are equipped with a rich supply of syntactic constructs and operators for supporting the needs of various applications. In these languages the support of concepts and roles, the main syntactic entities in description logic languages, is slightly uneven however. A notable absence is the negation operator for roles. In the description logic $\mathcal{ALC}$ and other popular extensions of $\mathcal{ALC}$ it is possible to define a *spam filter* as a mechanism for filtering out spam emails, and a *sound spam filter* as a spam filter that filters out only spam emails, by specifying

$$\mathsf{spam\text{-}filter} \stackrel{\mathsf{def}}{=} \mathsf{mechanism} \sqcap \exists \mathsf{filter\text{-}out}.\mathsf{spam\text{-}email} \quad \text{and}$$
$$\mathsf{sound\text{-}spam\text{-}filter} \stackrel{\mathsf{def}}{=} \mathsf{spam\text{-}filter} \sqcap \neg\exists \mathsf{filter\text{-}out}.\neg\mathsf{spam\text{-}email}.$$

Authors' address: R. A. Schmidt and D. Tishkovsky, School of Computer Science, The University of Manchester, Kilburn Building, Oxford Road, Manchester, M13 9PL, United Kingdom

DOI: http://dx.doi.org/10.1145/0000000.0000000

It is not possible however to define *a complete spam filter* as a spam filter that filters out every spam email. With role negation this can be expressed by the following.

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

The first occurrence of negation is a concept negation operator, which almost all description logics support, while the second occurrence is a role negation operator. The role negation operator is not available in $\mathcal{ALC}$ or other current description logics that form the basis of OWL DL/1.1/2.0 [Baader et al. 2003; OWL 2004; OWL 2 2009].

The three examples can be expressed in first-order logic as follows.

$$\forall x[\mathsf{spam\text{-}filter}(x) \leftrightarrow . \ \mathsf{mechanism}(x) \wedge \exists y[\mathsf{filter\text{-}out}(x,y) \wedge \mathsf{spam\text{-}email}(y)]]$$

$$(\dagger) \qquad \forall x[\mathsf{sound\text{-}spam\text{-}filter}(x) \leftrightarrow . \ \mathsf{spam\text{-}filter}(x) \wedge \forall y[\mathsf{filter\text{-}out}(x,y) \rightarrow \mathsf{spam\text{-}email}(y)]]$$

$$(\ddagger) \quad \forall x[\mathsf{complete\text{-}spam\text{-}filter}(x) \leftrightarrow . \ \mathsf{spam\text{-}filter}(x) \wedge \forall y[\mathsf{spam\text{-}email}(y) \rightarrow \mathsf{filter\text{-}out}(x,y)]].$$

The right-hand-sides of the equivalences (†) and (‡) involve universal quantification but of a different kind. In (†) it is the image elements of the role filter-out that are universally quantified, while in (‡) it is the elements in the concept spam-email that are universally quantified. (†) expresses the *necessity* of a property (what is filtered out is necessarily a spam message), while (‡) expresses the *sufficiency* of a property (being a spam message is sufficient to be filtered out). Natural examples of both kinds of universal quantification can be found in many domains and are common in every-day language.

Motivated by pushing the limits of description logics, 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 typically supported in mainstream description logics. In particular, we focus on a description logic called $\mathcal{ALBO}^{\mathsf{id}}$. $\mathcal{ALBO}^{\mathsf{id}}$ is the description logic $\mathcal{ALC}$ extended by union of roles, negation of roles, inverse of roles, and the identity role. In addition, it provides full support for ABox individuals and singleton concepts. $\mathcal{ALBO}^{\mathsf{id}}$ is an extension of the description logic $\mathcal{ALB}$, introduced in Hustadt and Schmidt [2000], with individuals and singleton concepts, called nominals in modal logic, and the identity role. $\mathcal{ALB}$ extends $\mathcal{ALC}$ in that concepts and roles form Boolean algebras, and additional operators are inverse of roles and a domain restriction operator. It is shown in this paper (Section 2) that the domain restriction operator can be linearly encoded using the other operators of $\mathcal{ALB}$.

$\mathcal{ALBO}^{\mathsf{id}}$ is a very expressive description logic. It subsumes Boolean modal logic [Gargov and Passy 1990; Gargov et al. 1987] and tense, hybrid versions of Boolean modal logic with the @ operator and nominals. It can also be shown that $\mathcal{ALBO}^{\mathsf{id}}$ is expressively equivalent to the two-variable fragment of first-order logic with equality. $\mathcal{ALBO}^{\mathsf{id}}$ is in fact very close to the brink of undecidability, because it is known that adding role composition to $\mathcal{ALB}$ takes us into undecidable territory.

Since $\mathcal{ALBO}^{\mathsf{id}}$ subsumes Boolean modal logic it follows from Lutz and Sattler [2002] that the satisfiability problem in $\mathcal{ALBO}^{\mathsf{id}}$ is NExpTime-hard. Grädel et al. [1997] showed that satisfiability in the two-variable first-order fragment with equality is NExpTime-complete. It follows therefore that the computational complexity of $\mathcal{ALBO}^{\mathsf{id}}$-satisfiability is NExpTime-complete.

Description logics with full role negation can be decided by translation to first-order logic and first-order resolution theorem provers such as MSPASS [Hustadt et al. 1999], SPASS [Weidenbach et al. 2007], E [Schulz 2002] and VAMPIRE [Riazanov and Voronkov 1999]. The paper [Hustadt and Schmidt 2000] shows that the logic $\mathcal{ALB}$ can be decided by translation to first-order logic and ordered resolution. This result is extended by De Nivelle et al. [2000] to $\mathcal{ALB}$ with positive occurrences of composition

of roles. $\mathcal{ALBO}^{\mathsf{id}}$ can be embedded into the two-variable fragment of first-order logic with equality which can be decided with first-order resolution methods [De Nivelle and Pratt-Hartmann 2001]. This means that $\mathcal{ALBO}^{\mathsf{id}}$ can be decided using first-order resolution methods.

None of the current tableau-based description logic systems are however able to handle $\mathcal{ALBO}^{\mathsf{id}}$ or $\mathcal{ALB}$ because they do not support full role negation. In fact, 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 by De Nivelle et al. [2000] for the modal versions of $\mathcal{ALC}(\sqcup, \sqcap, ^{-1})$, that is, $\mathcal{ALC}$ with role union, role intersection and role inverse. These are extended with the domain restriction operator to $\mathcal{ALC}(\sqcup, \sqcap, ^{-1}, \restriction)$ by Schmidt [2006a]. A semantic tableau decision procedure for $\mathcal{ALC}$ with role intersection, role inverse, role identity and role composition is described by Massacci [2001]. None of these tableaux make provision for the role negation operator however.

A tableau decision procedure for the description logic $\mathcal{ALCQI}b$ which allows for Boolean combinations of 'safe' occurrences of negated roles is discussed by Tobies [2001]. From the clausal form of the first-order translation it can be seen that safeness ensures guardedness which is violated by unsafe occurrences of role negation. Guardedness is also important in the definition of a generalisation with relational properties of the dynamic modal logic corresponding to $\mathcal{ALC}(\sqcup, \sqcap, ^{-1})$ in De Nivelle et al. [2000], where it is shown that hyperresolution and splitting decides this logic. The hyperresolution decision procedure for this logic is essentially a hypertableau approach with similarities to Motik et al. [2009]. Decidability of propositional dynamic logic with negation of atomic relations using Büchi automata is shown by Lutz and Walther [2004]. Schmidt et al. [2004] presented a sound and complete ground semantic tableau calculus 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 present a ground semantic tableau approach that decides the description logic $\mathcal{ALBO}^{\mathsf{id}}$. In order to limit the number of individuals in the tableau derivation and guarantee termination we need a mechanism for finding finite models. *Standard loop checking* mechanisms such as subset blocking or equality blocking are based on comparing sets of (labelled or unlabelled) concept expressions in order to detect periodicity in the underlying models. Instead of using the standard loop checking mechanisms, our approach is based on a new inference rule, called 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 does not require specialised blocking tests and procedural descriptions in terms of status variables. All individuals are blockable and once blocked remain blocked.

— The blocking mechanism is generally sound and complete. This means it is applicable also to other deduction methods and other logics [Schmidt and Tishkovsky 2008] including full first-order logic, where it can be used to find finite models.

— It provides greater flexibility in constructing models. For instance, it can be used to construct small models for a satisfiable concept, including domain minimal models.

— The approach has the advantage that it constructs real models, whereas existing tableau procedures for many OWL description logics construct only pseudo-models that are not always real models but can be completed by post-processing to real models (which may be infinite).

— It can be implemented and simulated in first-order logic provers [Baumgartner and Schmidt 2008].

The style of presentation of our tableau calculus is similar to presentations in modal and hybrid logic, for example [Fitting 1972; Blackburn 2000; De Nivelle et al. 2000; Schmidt 2006a; Schmidt et al. 2004; del Cerro and Gasquet 2002]. Notable about our calculus compared, for example, with [Schmidt 2006a; Schmidt et al. 2004] is that it operates only on ground labelled *concept* expressions. This makes it easier in principle to implement the calculus as extensions of tableau-based description logic systems that can handle singleton concepts and include equality reasoning.

The structure of the paper is as follows. The syntax and semantics of $\mathcal{ALBO}^{\mathsf{id}}$ are defined in Section 2. In Section 3 we prove that $\mathcal{ALBO}^{\mathsf{id}}$ has the effective finite model property by reducing $\mathcal{ALBO}^{\mathsf{id}}$-satisfiability to the two-variable fragment of first-order logic. We define a tableau calculus for $\mathcal{ALBO}^{\mathsf{id}}$ in Section 4, and in Section 5, we prove that it is sound and complete without the unrestricted blocking rule. Sections 6 and 7 introduce the unrestricted blocking mechanism and prove soundness, completeness and termination of the tableau calculus extended with this mechanism. In Section 8 we prove an appropriate strategy for the application of the unrestricted blocking rule yields a tableau procedure with optimal complexity. We define general criteria for deterministic decision procedures for $\mathcal{ALBO}^{\mathsf{id}}$ (and any sublogics) in Section 9. Various issues concerning blocking and the results are discussed in Section 10.

The paper is an extended and improved version of the conference paper [Schmidt and Tishkovsky 2007] and builds on results in Schmidt and Tishkovsky [2008].

2. SYNTAX AND SEMANTICS OF $\mathcal{ALBO}^{\mathsf{id}}$

The syntax of $\mathcal{ALBO}^{\mathsf{id}}$ is defined over a signature, denoted by $\Sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$, consisting of three disjoint alphabets: O, the alphabet of individuals or object names, C, the alphabet of concept symbols, and R, the alphabet of role symbols. For individuals we use the notation a, a', a'', a_i, for concept symbols A, A', A_i, and for role symbols Q, Q', Q'', Q_i. The language includes a special role symbol id for the *identity* role which is regarded as a constant. The logical connectives are: $\neg$ (*negation*), $\sqcup$ (*union* for both concepts and roles), $\exists$ (*existential concept restriction*), and $^{-1}$ (role *inverse*). *Concept expressions* (or *concepts*) C, D and *role expressions* (or *roles*) R, S are defined by the following grammar rules:

$$C, D \stackrel{\text{def}}{=} A \mid \{a\} \mid \neg C \mid C \sqcup D \mid \exists R.C,$$
$$R, S \stackrel{\text{def}}{=} Q \mid R \sqcup S \mid R^{-1} \mid \neg R \mid \mathsf{id}.$$

We refer to $\{a\}$ as a *singleton* concept. By the *length* of an $\mathcal{ALBO}^{\mathsf{id}}$-expression E we understand the length in symbols of the word representing E in the above language.

A TBox is a finite set of concept inclusion statements of the form $C \sqsubseteq D$ and an RBox is a finite set of role inclusion statements of the form $R \sqsubseteq S$. An ABox is a finite set of statements of the form $a : C$ and $(a, a') : R$, called *concept assertions* and *role assertions*, respectively. A *knowledge base* is a tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$, and an ABox $\mathcal{A}$.

Next, we define the semantics of $\mathcal{ALBO}^{\mathsf{id}}$. A *model* (or an *interpretation*) $\mathcal{I}$ of $\mathcal{ALBO}^{\mathsf{id}}$ is a tuple

$$\mathcal{I} = (\Delta^{\mathcal{I}}, A^{\mathcal{I}}, \ldots, a^{\mathcal{I}}, \ldots, Q^{\mathcal{I}}, \ldots),$$

where $\Delta^{\mathcal{I}}$ is a non-empty set, and for any concept symbol A, any role symbol Q and any individual a:

$$A^{\mathcal{I}} \text{ is a subset of } \Delta^{\mathcal{I}}, \quad a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \quad \text{and} \quad Q^{\mathcal{I}} \text{ is a binary relation over } \Delta^{\mathcal{I}}.$$

We expand the interpretation $\mathcal{I}$ to all concepts and roles with the following definition (for any concepts C, D, roles R, S, and individuals a):

$$\begin{aligned}
\{a\}^{\mathcal{I}} &\stackrel{\text{def}}{=} \{a^{\mathcal{I}}\}, & (R \sqcup S)^{\mathcal{I}} &\stackrel{\text{def}}{=} R^{\mathcal{I}} \cup S^{\mathcal{I}},\\
(\neg C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, & (R^{-1})^{\mathcal{I}} &\stackrel{\text{def}}{=} (R^{\mathcal{I}})^{-1} = \{(x,y) \mid (y,x) \in R^{\mathcal{I}}\},\\
(C \sqcup D)^{\mathcal{I}} &\stackrel{\text{def}}{=} C^{\mathcal{I}} \cup D^{\mathcal{I}}, & (\neg R)^{\mathcal{I}} &\stackrel{\text{def}}{=} (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}},\\
(\exists R.C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{x \mid \exists y \in C^{\mathcal{I}}\ (x,y) \in R^{\mathcal{I}}\}, & \mathsf{id}^{\mathcal{I}} &\stackrel{\text{def}}{=} \{(x,x) \mid x \in \Delta^{\mathcal{I}}\}.
\end{aligned}$$

A concept C is *satisfied* in a model $\mathcal{I}$ iff $C^{\mathcal{I}} \neq \emptyset$. A set $\mathcal{S}$ of concepts is satisfied in a model $\mathcal{I}$ iff every concept C in $\mathcal{S}$ is satisfied in $\mathcal{I}$. A concept (or a set of concepts) is *satisfiable* iff there is a model in which it is satisfied.

Slightly departing from description logic conventions, we say a concept C is *valid* in a model $\mathcal{I}$ iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. A set of concepts is valid in a model $\mathcal{I}$ iff all the concepts in the set are valid in $\mathcal{I}$. If E is a concept expression, a concept inclusion statement, a role inclusion statement, a concept assertion or a role assertion, we indicate by $\mathcal{I} \models E$ that E is valid in the model $\mathcal{I}$.[1] In particular, shortening the definition of concept validity, we can specify that, in any model $\mathcal{I}$ and for any concept C,

$$\mathcal{I} \models C \iff^{\text{def}} C^{\mathcal{I}} = \Delta^{\mathcal{I}}.$$

Concept and role inclusion statements are interpreted as subset relationships, and concept and role assertions are interpreted as element-of relationships. In particular, in any model $\mathcal{I}$ and for any concepts C, D, any roles R, S, and any individuals a, a':

$$\begin{aligned}
\mathcal{I} \models C \sqsubseteq D &\iff^{\text{def}} C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, & \mathcal{I} \models a : C &\iff^{\text{def}} a^{\mathcal{I}} \in C^{\mathcal{I}},\\
\mathcal{I} \models R \sqsubseteq S &\iff^{\text{def}} R^{\mathcal{I}} \subseteq S^{\mathcal{I}}, & \mathcal{I} \models (a,a') : R &\iff^{\text{def}} (a^{\mathcal{I}}, a'^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{aligned}$$

A concept C is *satisfiable with respect to a knowledge base* $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ iff C is satisfied in a model $\mathcal{I}$ validating all the statements from the knowledge base, that is, $\mathcal{I} \models E$ for every $E \in \mathcal{T} \cup \mathcal{R} \cup \mathcal{A}$.

$\mathcal{ALBO}^{\mathsf{id}}$ has considerable expressive power and many useful additional operators can be defined in it. These include the following standard operators:

Top concept:	$\top \stackrel{\text{def}}{=} A \sqcup \neg A$ (for some concept symbol A)
Bottom concept:	$\bot \stackrel{\text{def}}{=} \neg\top$
Concept intersection:	$C \sqcap D \stackrel{\text{def}}{=} \neg(\neg C \sqcup \neg D)$
Universal restriction:	$\forall R.C \stackrel{\text{def}}{=} \neg\exists R.\neg C$
Image operator:	$\exists^{-1}R.C \stackrel{\text{def}}{=} \exists R^{-1}.C$

[1] In the literature $\mathcal{I} \models E$ is normally only defined when E is an inclusion, equivalence or assertion, and then it is said that E is true in $\mathcal{I}$.

plus these operators, which can be defined using role negation:

Top/universal role:	$\nabla \stackrel{\text{def}}{=} Q \sqcup \neg Q$ (for some role symbol Q)
Bottom/empty role:	$\triangle \stackrel{\text{def}}{=} \neg\nabla$
Role intersection:	$R \sqcap S \stackrel{\text{def}}{=} \neg(\neg R \sqcup \neg S)$
Diversity role:	$\mathsf{div} \stackrel{\text{def}}{=} \neg\mathsf{id}$
Universal modality:	$\Box C \stackrel{\text{def}}{=} \forall\nabla.C$
Sufficiency operator:	$\overline{\forall} R.C \stackrel{\text{def}}{=} \neg\exists\neg R.C$

The bottom role can be expressed by using a concept inclusion stating that the domain of the role is empty. The sufficiency operator is also known as the window operator, see for example [Gargov et al. 1987].

Concept assertions can be internalised as concept expressions as follows:

Concept assertions:	$a : C \stackrel{\text{def}}{=} \exists\nabla.(\{a\} \sqcap C).$

A role assertion $(a, a') : R$ can be expressed as a concept assertion, namely

Role assertions:	$(a, a') : R \stackrel{\text{def}}{=} a : \exists R.\{a'\},$

or, using the above, it can be internalised as a concept expression. In addition, concept and role inclusion axioms can be internalised as concept expressions.

Concept inclusion:	$C \sqsubseteq D \stackrel{\text{def}}{=} \Box(\neg C \sqcup D)$
Role inclusion:	$R \sqsubseteq S \stackrel{\text{def}}{=} \Box\forall\neg(\neg R \sqcup S).\bot$

Concept assertions, role assertions, concept inclusions and role inclusions in $\mathcal{ALBO}^{\mathsf{id}}$ are therefore all representable as concept expressions. This means that in $\mathcal{ALBO}^{\mathsf{id}}$ the distinction between TBoxes, ABoxes and RBoxes is not strictly necessary, and concept satisfiability in $\mathcal{ALBO}^{\mathsf{id}}$ with respect to any knowledge base can be reduced to concept satisfiability with respect to a knowledge base where the TBox, the RBox, and the ABox are all empty. More precisely, we have that a concept C is satisfiable with respect to a knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ iff the conjunction

$$C \sqcap \bigsqcap(\mathcal{T} \cup \mathcal{R} \cup \mathcal{A})$$

of the concept C and all the concept expressions from the set $\mathcal{T} \cup \mathcal{R} \cup \mathcal{A}$ is satisfiable. Similarly, a set $\mathcal{S}$ of concepts is satisfiable with respect to a knowledge base $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ iff the set $\mathcal{S} \cup \mathcal{T} \cup \mathcal{R} \cup \mathcal{A}$ is satisfiable. Without loss of generality, in this paper, we therefore focus on the problem of concept satisfiability in $\mathcal{ALBO}^{\mathsf{id}}$.

Boolean combinations of inclusion and assertion statements of concepts and roles are expressible in $\mathcal{ALBO}^{\mathsf{id}}$, as the corresponding Boolean combinations of the concept expressions representing these statements. In fact, we are allowed to use such statements in concept and role expressions within the language of $\mathcal{ALBO}^{\mathsf{id}}$ provided that a well-formed $\mathcal{ALBO}^{\mathsf{id}}$ expression is obtained after replacing all these statements in the original expression for well-formed $\mathcal{ALBO}^{\mathsf{id}}$ expressions which define them. For example, the following expression is a well-formed concept expression in $\mathcal{ALBO}^{\mathsf{id}}$:

$$((\mathsf{Alice} : \neg\exists\mathsf{likes}.\mathsf{football\_fan}) \sqcap (\mathsf{Bob} : \mathsf{football\_fan}) \sqcap (\mathsf{hasFriend} \sqsubseteq \mathsf{likes})) \sqsubseteq ((\mathsf{Alice}, \mathsf{Bob}) : \neg\mathsf{hasFriend}).$$

It says that Alice and Bob cannot be friends because Bob is fanatical about football and Alice does not like football fans. The sentence uses an assumption expressed as a role inclusion that somebody's friend is necessarily a person who he or she likes. Notice that the role inclusion statement denotes an $\mathcal{ALBO}^{\mathsf{id}}$ expression which essentially uses the role negation operator.

The encodings of the operators mentioned above preserve logical equivalence. Additionally, $\mathcal{ALBO}^{\mathsf{id}}$ also has enough expressive power to represent concept satisfiability problems which involve the operators defined below:

Test operator:	$(C?)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid x \in C^{\mathcal{I}}\},$
Domain restriction:	$(R\restriction C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \in R^{\mathcal{I}} \mid x \in C^{\mathcal{I}}\},$
Range restriction:	$(R\downharpoonright C)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \in R^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\},$
Left cylindrification:	$(D^c)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid x \in D^{\mathcal{I}}\},$
Right cylindrification:	$({}^cD)^{\mathcal{I}} \stackrel{\text{def}}{=} \{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid y \in D^{\mathcal{I}}\},$
Cross product:	$(C \times D)^{\mathcal{I}} \stackrel{\text{def}}{=} C^{\mathcal{I}} \times D^{\mathcal{I}} = \{(x,y) \mid x \in C^{\mathcal{I}}, y \in D^{\mathcal{I}}\}.$

The test operator can be represented using the range restriction operator and the identity role: $(C?)^{\mathcal{I}} = (\mathsf{id}\downharpoonright C)^{\mathcal{I}}$. In the presence of role inverse, the universal role and role intersection any single operator from $\{\restriction, \downharpoonright, \cdot^c, {}^c\cdot, \times\}$ is enough to define the remaining operators from this set. In any model $\mathcal{I}$, the following equalities are true.

$$
\begin{array}{rclcrcl}
(R\restriction C)^{\mathcal{I}} &=& \left((R^{-1}\downharpoonright C)^{-1}\right)^{\mathcal{I}} & & (R\downharpoonright C)^{\mathcal{I}} &=& \left((R^{-1}\restriction C)^{-1}\right)^{\mathcal{I}} \\
(D^c)^{\mathcal{I}} &=& \left(({}^cD)^{-1}\right)^{\mathcal{I}} & & ({}^cD)^{\mathcal{I}} &=& \left(D^{c-1}\right)^{\mathcal{I}} \\
(D^c)^{\mathcal{I}} &=& (\nabla\restriction D)^{\mathcal{I}} & & ({}^cD)^{\mathcal{I}} &=& (\nabla\downharpoonright D)^{\mathcal{I}} \\
(R\restriction D)^{\mathcal{I}} &=& (R \sqcap D^c)^{\mathcal{I}} & & (R\downharpoonright D)^{\mathcal{I}} &=& (R \sqcap {}^cD)^{\mathcal{I}} \\
(D^c)^{\mathcal{I}} &=& (D \times \top)^{\mathcal{I}} & & ({}^cD)^{\mathcal{I}} &=& (\top \times D)^{\mathcal{I}} \\
(C \times D)^{\mathcal{I}} &=& (C^c \sqcap {}^cD)^{\mathcal{I}} & & & &
\end{array}
$$

Now it is enough to show that one of these operators can be encoded in $\mathcal{ALBO}^{\mathsf{id}}$. For example, expressions involving left cylindrification can be *linearly* encoded in $\mathcal{ALBO}^{\mathsf{id}}$ by replacing all occurrences of D^c in C by a new role symbol Q_D uniquely associated with D and adding the definitions $\neg D \sqsubseteq \forall Q_D.\bot$ and $D \sqsubseteq \neg\exists\neg Q_D.\top$ to the knowledge base. This encoding preserves satisfiability equivalence. In a similar way, the other operators from $\{\restriction, \downharpoonright, \cdot^c, {}^c\cdot, \times\}$ can be linearly encoded in $\mathcal{ALBO}^{\mathsf{id}}$.

Often description logics are required to satisfy the unique name assumption. We do not assume it for $\mathcal{ALBO}^{\mathsf{id}}$. However, the unique name assumption can be enforced by adding disjointness statements of the form $\{a\} \sqcap \{a'\} \sqsubseteq \bot$, for every distinct pair of individuals that occur in the given knowledge base, to the TBox.

In this paper we do not use role assertions in the form $(a, a') : R$ but use them in the form $a : \exists R.\{a'\}$. We refer to $a : \exists R.\{a'\}$ as a *link* (between the individuals a and a').

Later we refer to the description logics $\mathcal{ALCO}$ and $\mathcal{ALCO}^{\Box}$. $\mathcal{ALCO}$ is a sublogic of $\mathcal{ALBO}^{\mathsf{id}}$ without all the role operators and the identity role. $\mathcal{ALCO}^{\Box}$ extends $\mathcal{ALCO}$ with the universal modality operator $\Box$.

3. EFFECTIVE FINITE MODEL PROPERTY

In this section we show that $\mathcal{ALBO}^{\mathsf{id}}$ has the effective finite model property. In order to make use of the well-known result of Grädel et al. [1997] that the two-variable fragment of first-order logic with equality has the finite model property, we encode

$\mathcal{ALBO}^{\mathsf{id}}$-satisfiability of concepts in the two-variable fragment of first-order logic with equality. Additionally, we define a mapping of models for the two-variable fragment back to $\mathcal{ALBO}^{\mathsf{id}}$-models and prove that it preserves satisfiability modulo the mentioned encoding.

For every concept symbol A, role symbol Q, and individual a, let $\underline{A}$ be a unary predicate symbol, $\underline{Q}$ a binary predicate symbol, and $\underline{a}$ a constant which are uniquely associated with A, Q, and a, respectively. The following defines a mapping ST as the standard translation of $\mathcal{ALBO}^{\mathsf{id}}$ parametrised by the variables x and y. The standard translation of concepts is:

$$\begin{aligned}
\mathsf{ST}_x(A) &\stackrel{\text{def}}{=} \underline{A}(x) & \mathsf{ST}_x(\{a\}) &\stackrel{\text{def}}{=} x \approx \underline{a}\\
\mathsf{ST}_x(\neg C) &\stackrel{\text{def}}{=} \neg \mathsf{ST}_x(C) & \mathsf{ST}_x(C \sqcup D) &\stackrel{\text{def}}{=} \mathsf{ST}_x(C) \vee \mathsf{ST}_x(D)\\
\mathsf{ST}_x(\exists R.C) &\stackrel{\text{def}}{=} \exists y\, (\mathsf{ST}_{xy}(R) \wedge \mathsf{ST}_y(C)).
\end{aligned}$$

The standard translation of roles is:

$$\begin{aligned}
\mathsf{ST}_{xy}(Q) &\stackrel{\text{def}}{=} \underline{Q}(x,y) & \mathsf{ST}_{xy}(\mathsf{id}) &\stackrel{\text{def}}{=} x \approx y\\
\mathsf{ST}_{xy}(R^{-1}) &\stackrel{\text{def}}{=} \mathsf{ST}_{yx}(R) & \mathsf{ST}_{xy}(R \sqcup S) &\stackrel{\text{def}}{=} \mathsf{ST}_{xy}(R) \vee \mathsf{ST}_{xy}(S)\\
\mathsf{ST}_{xy}(\neg R) &\stackrel{\text{def}}{=} \neg \mathsf{ST}_{xy}(R).
\end{aligned}$$

Clearly, only two free variables occur in $\mathsf{ST}_x(C)$ for every concept C. Given a concept C, the translation $\mathsf{ST}_x(C)$ of C can be computed in linear time with respect to the length of C. Furthermore it can be shown that the length of $\mathsf{ST}_x(C)$ (that is, the length in symbols of the word representing $\mathsf{ST}_x(C)$ excluding commas and parentheses) increases only by a constant factor compared to the length of C. More precisely, the following lemma can be shown by induction on the structure of C.

LEMMA 3.1. *If n is the length of C then the length of $\mathsf{ST}_x(C)$ does not exceed $3n$.*

As the standard translation just follows the definition of the semantics of $\mathcal{ALBO}^{\mathsf{id}}$, every first-order model $\mathcal{I}$ over the signature $\underline{\Sigma} = (\{\underline{a} \mid a \in \mathsf{O}\}, \{\underline{A} \mid A \in \mathsf{C}\}, \{\underline{Q} \mid Q \in \mathsf{R}\})$ can be viewed as an $\mathcal{ALBO}^{\mathsf{id}}$-model, where the interpretations $A^{\mathcal{I}}$, $Q^{\mathcal{I}}$, and $a^{\mathcal{I}}$ coincide with the interpretations of the first-order symbols $\underline{A}$, $\underline{Q}$, and $\underline{a}$, respectively. Similarly, every $\mathcal{ALBO}^{\mathsf{id}}$-model can be seen as a first-order model over the signature $\underline{\Sigma}$.

We write $\mathcal{I} \models \phi[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]$ to indicate that all free variables of a first-order formula ϕ are contained in the set $\{x_1, \ldots, x_n\}$ and ϕ is true in $\mathcal{I}$, where the variables $x_1, \ldots, x_n$ are respectively assigned the domain elements $a_1, \ldots, a_n$.

The following theorem can be proved by simultaneous induction on the lengths of concepts and roles.

THEOREM 3.2. *Let $\mathcal{I}$ be any $\mathcal{ALBO}^{\mathsf{id}}$-model. Then, for any concept C, any role R and any $a, b \in \Delta^{\mathcal{I}}$,*

— *$a \in C^{\mathcal{I}}$ iff $\mathcal{I} \models \mathsf{ST}_x(C)[x \mapsto a]$, and*
— *$(a,b) \in R^{\mathcal{I}}$ iff $\mathcal{I} \models \mathsf{ST}_{xy}(R)[x \mapsto a, y \mapsto b]$.*

The effective finite model property for description logics can be stated as follows. A description logic L has the *effective finite model property* iff there is a computable function $\mu : \mathbb{N} \to \mathbb{N}$, such that the following holds.

> For every concept C, if C is satisfiable in an L-model then there is a finite L-model for C with the number of elements in the domain not exceeding $\mu(n)$, where n is the length of C.

We call μ the *(model) bounding function* for L.

The notion of the effective finite model property for fragments of first-order logic can be obtained from the above definition replacing the phrases 'logic L' and 'concept C' with 'fragment $\mathcal{F}$' and 'formula ϕ', respectively.

The effective finite model property is sometimes called the bounded model property or the small model property and is stronger than the finite model property. A logic has the finite model property if any satisfiable concept (formula) has a finite model. With the effective finite model property there is a bound on the size of the model that is given a priori based on the given concept (or formula). This means there is a naive procedure for deciding satisfiability, which is based on successively computing larger and larger models within the bound and model checking the given concept with respect to the obtained models.

The effective finite model property for the two-variable fragment of first-order logic with equality is proved by Mortimer [1975]. An exponential model bounding function μ is given by Grädel et al. [1997]. Because we use this function in Sections 7 and 8, we define it explicitly:

$$\mu(n) \stackrel{\text{def}}{=} 3(n\lfloor\log(n+1)\rfloor)\cdot 2^n.$$

THEOREM 3.3 (GRÄDEL ET AL. [1997]). *The two variable fragment of first-order logic with equality has the effective finite model property with model bounding function* $\mu(n)$.

Taking into account Lemma 3.1, let

$$\mu_0(n) \stackrel{\text{def}}{=} 3(3n\lfloor\log(3n+1)\rfloor)\cdot 2^{3n} = 9(n\lfloor\log(3n+1)\rfloor)\cdot 8^n.$$

Theorems 3.2 and 3.3 imply:

THEOREM 3.4 (EFFECTIVE FINITE MODEL PROPERTY OF $\mathcal{ALBO}^{\mathsf{id}}$). $\mathcal{ALBO}^{\mathsf{id}}$ *has the effective finite model property with model bounding function* $\mu_0(n)$.

This provides an upper bound for the worst-case complexity of testing concept satisfiability for $\mathcal{ALBO}^{\mathsf{id}}$. The complexity result for Boolean modal logic of Lutz and Sattler [2002] gives the needed lower bound for concluding:

THEOREM 3.5. *The concept satisfiability problem for* $\mathcal{ALBO}^{\mathsf{id}}$ *is* NExpTime*-complete.*

4. TABLEAU CALCULUS

Let T denote a tableau calculus comprising of a set of inference rules. A *derivation* or *tableau* for T is a finitely branching, ordered tree whose nodes are sets of labelled concept expressions. Assuming that $\mathcal{S}$ is the input set of concept expressions to be tested for satisfiability, the root node of the tableau is the set $\{a : C \mid C \in \mathcal{S}\}$, where a denotes a fresh individual. Successor nodes are constructed in accordance with the inference rules in the calculus. The inference rules have the general form

$$\frac{X_0}{X_1 \mid \ldots \mid X_n},$$

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 $X_0/\bot$. An inference rule is applicable to a selected labelled concept expression E in a node of the tableau, if E together with possibly other labelled concept expressions in the node, are simultaneous instantiations of all the premises of the rule. Then n successor nodes are created which contain the formulae of the current node and the appropriate instances of $X_1, \ldots, X_n$.

In a tableau, a maximal path from the root node is called a *branch*. For a branch $\mathcal{B}$ of a tableau we write $D \in \mathcal{B}$ to indicate that the concept D has been derived in $\mathcal{B}$, that is, D belongs to a node of the branch $\mathcal{B}$. The notion of a tableau branch we use in this paper can be viewed in two ways. On the one hand it has a procedural flavour as a path of nodes in the tableau. On the other hand a branch can be identified with the set-theoretical union of the nodes in it. A more careful distinction between these perspectives leads to the classical notion of a Hintikka set but this is not essential for the paper.

We use the notation $T(\mathcal{S})$ for a fully expanded tableau built by applying the rules of the calculus T starting with the set $\mathcal{S}$ as input. That is, we assume that all branches in the tableau are fully expanded and all applicable rules have been applied in $T(\mathcal{S})$. For any concepts $C_1, \ldots, C_n$, the notation $T(C_1, \ldots, C_n)$ is used instead of $T(\{C_1, \ldots, C_n\})$. We assume that, *in each tableau branch, any rule is applied at most once to the same set of premises*, which is a standard assumption for tableau derivations in order to prevent unnecessary reapplication of rules.

A branch of a tableau is *closed* if a closure rule has been applied in this branch, otherwise the branch is called *open*. Clearly, expansion of any closed branch can be stopped immediately after the first application of a closure rule in the branch. The tableau $T(\mathcal{S})$ is *closed* if all its branches are closed and $T(\mathcal{S})$ is *open* otherwise. The calculus T is *sound* iff for any (possibly infinite) set of concepts $\mathcal{S}$, each $T(\mathcal{S})$ is open whenever $\mathcal{S}$ is satisfiable. T is *complete* iff for any (possibly infinite) unsatisfiable set of concepts $\mathcal{S}$ there is a tableau $T(\mathcal{S})$ which is closed. T is said to be *(weakly) terminating* (for satisfiability) iff for every *finite* set of concepts $\mathcal{S}$ every open tableau $T(\mathcal{S})$ has a finite open branch and every closed tableau $T(\mathcal{S})$ is finite.[2]

Let $T_{\mathcal{ALBO}^{\mathsf{id}}}$ be the tableau calculus consisting of the rules listed in Figure 1.

The first group of rules are the rules for decomposing concept expressions. They are in fact the rules for testing satisfiability of concepts for the description logic $\mathcal{ALCO}$, that is, $\mathcal{ALC}$ with individuals. The ($\bot$) rule is the closure rule. The ($\neg\neg$) rule removes occurrences of double negation on concepts. (The rule is superfluous if double negations are eliminated using on-the-fly rewrite rules, but is included to simplify the completeness proof somewhat.) The ($\sqcup$) and ($\neg\sqcup$) rules are standard rules for handling concept disjunctions. As usual, and in accordance with the semantics of the existential restriction operator, for any 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 in the calculus that generates new individuals. An additional side-condition is that the new individual a' is uniquely associated with the premise $a : \exists R.C$.[3] As usual we assume the ($\exists$) rule is applied only to expressions of the form $a : \exists R.C$, when C is not a singleton, that is, $C \neq \{a''\}$ for any individual a''. This condition prevents the application of the ($\exists$) rule to expressions of the form $a : \exists R.\{a''\}$ because this would just cause another witness of an R-successor of a to be created when a'' is already such a witness. The ($\neg\exists$) rule is equivalent to the standard rule for universally restricted concept expressions. The (sym), (mon), and (refl) rules are the equality rules for individuals, familiar from hybrid logic tableau systems. They can be viewed as versions of standard rules for first-order equality. The (refl) rule is formulated a bit unusually, because it adds a validity $a : \{a\}$ to the branch. In our presentation expressions of the form $a : \{a\}$ are used to represent the individuals occurring on a branch, which is

[2] The latter is optional for the case of $\mathcal{ALBO}^{\mathsf{id}}$.

[3] The side-condition is not strictly needed because the calculus is non-destructive. The condition is however important to prevent too many individuals to be introduced for variations of the calculus based on non-retaining rules, or as soon as, for example, deletion or simplification rules are included, or equality reasoning is based on rewriting or some other form of substitution of individuals is performed.

Rules for $\mathcal{ALCO}$:

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

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

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

$$(\mathsf{sym})\colon \frac{a : \{a'\}}{a' : \{a\}} \qquad (\neg\mathsf{sym})\colon \frac{a : \neg\{a'\}}{a' : \neg\{a\}}$$

$$(\mathsf{mon})\colon \frac{a : \{a'\},\ \ a' : C}{a : C} \qquad (\mathsf{refl})\colon \frac{a : C}{a : \{a\}}$$

Rules for complex roles:

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

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

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

$$(\exists\mathsf{id})\colon \frac{a : \exists\mathsf{id}.\{a'\}}{a : \{a'\}} \qquad (\neg\exists\mathsf{id})\colon \frac{a : \neg\exists\mathsf{id}.C}{a : \neg C}$$

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

exploited by the $(\neg\exists\neg)$ rule. In order to keep track of all individual appearances in a branch, we assume that the (refl) rule has a higher priority than the $(\exists)$ rule. See also remarks in Section 10. The $(\neg\mathsf{sym})$ rule is needed to ensure that any negated singleton concept eventually appears as a label in a concept assertion.

The rules in the second group are the 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. These are listed in the left and right columns, respectively. Due to the presence of the $(\exists)$ rule, the rules for positive existential roles are restricted to role assertions, that is, the concept expressions immediately below the $\exists$ operator are singleton concepts.

Among the rules for negated existential roles, the $(\neg\exists^{-1})$ rule 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:

$$x \in (\neg\exists\neg R.C)^{\mathcal{I}} \iff \forall y \left((x, y) \in R^{\mathcal{I}} \vee y \in (\neg C)^{\mathcal{I}}\right).$$

That is, a' in the rule is implicitly quantified by a universal quantifier. The effect of the second premise, $a' : \{a'\}$, is to instantiate a' with individuals that occur in the branch. The remaining rules in this subgroup are based on obvious logical equivalences in $\mathcal{ALBO}^{\mathsf{id}}$.

1.	$a_0 : \neg(\neg\exists(Q \sqcup \neg Q).A \sqcup \exists Q.A)$	given
2.	$a_0 : \neg\neg\exists(Q \sqcup \neg Q).A)$	$(\neg\sqcup)$,1
3.	$a_0 : \neg\exists Q.A$	$(\neg\sqcup)$,1
4.	$a_0 : \exists(Q \sqcup \neg Q).A$	$(\neg\neg)$,2
5.	$a_0 : \exists(Q \sqcup \neg Q).\{a_1\}$	$(\exists)$,4
6.	$a_1 : A$	$(\exists)$,4
7.	$a_0 : \{a_0\}$	(refl),1
8.	$a_1 : \{a_1\}$	(refl),6
9.	▶$a_0 : \exists Q.\{a_1\}$	$(\exists\sqcup)$,5
10.	$a_1 : \neg A$	$(\neg\exists)$,3,9
11.	Unsatisfiable	$(\bot)$,6,10
12.	▶$a_0 : \exists\neg Q.\{a_1\}$	$(\exists\sqcup)$,5
13.	$a_0 : \neg\exists Q.\{a_1\}$	$(\exists\neg)$,12
14.	Satisfiable	(no more rules are applicable)

Fig. 2. A derivation in $T_{\mathcal{ALBO}^{\mathsf{id}}}$

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

Now, given an input set of concepts $\mathcal{S}$, a tableau derivation is constructed as follows. First, preprocessing is performed. This pushes occurrences of the role inverse operator in every concept in $\mathcal{S}$ inward toward atomic concepts by exhaustively applying the following role equivalences from left to right:

$$(R \sqcup S)^{-1} = R^{-1} \sqcup S^{-1}, \qquad (\neg R)^{-1} = \neg(R^{-1}),$$
$$(R^{-1})^{-1} = R, \qquad \mathsf{id}^{-1} = \mathsf{id}.$$

This preprocessing is required to allow the tableau algorithm to handle the role inverse operator correctly. Pushing inverse inwards is the only preprocessing required and can be done in linear time. Transformation to negation normal form is optional but for practical purposes not necessarily a good choice because obvious properties of expressions and their negation require unnecessary inference steps to reveal.

Suppose $\mathcal{S}'$ is the set of concepts resulting from preprocessing. Then we build a complete tableau $T_{\mathcal{ALBO}^{\mathsf{id}}}(\mathcal{S}')$ as described above.

In the rest of the paper when we refer to the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$ we assume the described preprocessing with respect to role inverse has been applied to the input concept (set) before the rules of the calculus are applied. It is also important to note that $a : C$ and all labelled expressions and assertions really denote concept expressions.

An example of a finite derivation in $T_{\mathcal{ALBO}^{\mathsf{id}}}$ is shown in Figure 2 for the concept

$$\neg\left(\neg\exists(Q \sqcup \neg Q).A \sqcup \exists Q.A\right).$$

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. The black triangles denote branching points in the derivation. A branch expansion after a branching point is indicated by appropriate indentation. The derivation presents a finished tableau with two branches; the left branch is closed and the right branch is open. Hence, because the tableau calculus is complete, which is shown in the next section, the input concept $\neg\left(\neg\exists(Q \sqcup \neg Q).A \sqcup \exists Q.A\right)$ is satisfiable.

5. SOUNDNESS AND COMPLETENESS

We turn to proving soundness and completeness of the calculus. It is easy to see that every rule preserves the satisfiability of concept assertions. More precisely, given an $\mathcal{ALBO}^{\mathsf{id}}$-model, for any rule $X_0/X_1 \mid \cdots \mid X_m$ of the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$ if a set $X_0\sigma$ of instantiations of premises of the rule under a substitution σ is satisfiable in the model then one of the sets $X_1\sigma, \ldots, X_m\sigma$ is also satisfiable in the model. Any rule with this property is said to be *sound*. Since all the rules of the calculus are sound this implies the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$ is sound.

THEOREM 5.1 (SOUNDNESS). *$T_{\mathcal{ALBO}^{\mathsf{id}}}$ is a sound tableau calculus for satisfiability in $\mathcal{ALBO}^{\mathsf{id}}$.*

For proving completeness, suppose that a tableau $T_{\mathcal{ALBO}^{\mathsf{id}}}(\mathcal{S})$ for a given set of concepts $\mathcal{S}$ is open, that is, it contains an open branch $\mathcal{B}$. From $\mathcal{B}$ we construct a model $\mathcal{I}(\mathcal{B})$ for the satisfiability of $\mathcal{S}$ as follows. By definition, let

$$a \sim_{\mathcal{B}} a' \iff^{\mathsf{def}} a : \{a'\} \in \mathcal{B}.$$

The rules (sym), (mon), and (refl) ensure that $\sim_{\mathcal{B}}$ is an equivalence relation on individuals. Define the equivalence class $\|a\|$ of an individual a by:

$$\|a\| \stackrel{\mathsf{def}}{=} \{a' \mid a \sim_{\mathcal{B}} a'\}.$$

We set $\Delta^{\mathcal{I}(\mathcal{B})} \stackrel{\mathsf{def}}{=} \{\|a\| \mid a : \{a\} \in \mathcal{B}\}$, and for every $Q \in \mathsf{R}$, $A \in \mathsf{C}$ and $a \in \mathsf{O}$ we define

$$\begin{aligned}
(\|a\|, \|a'\|) \in Q^{\mathcal{I}(\mathcal{B})} &\iff^{\mathsf{def}} a : \exists Q.\{a''\} \in \mathcal{B} \text{ for some } a'' \sim_{\mathcal{B}} a', \\
\|a\| \in A^{\mathcal{I}(\mathcal{B})} &\iff^{\mathsf{def}} a : A \in \mathcal{B}, \\
a^{\mathcal{I}(\mathcal{B})} &\stackrel{\mathsf{def}}{=} \begin{cases} \|a\|, & \text{if } a : \{a\} \in \mathcal{B}, \\ \|a'\| & \text{for some } \|a'\| \in \Delta^{\mathcal{I}(\mathcal{B})}, \text{ otherwise.} \end{cases}
\end{aligned}$$

The rule (mon) ensures that the definitions of $Q^{\mathcal{I}(\mathcal{B})}$ and $A^{\mathcal{I}(\mathcal{B})}$ are correct and do not depend on the choice of the representative a of the equivalence class $\|a\|$. Finally, we expand the interpretation mapping $\cdot^{\mathcal{I}(\mathcal{B})}$ to all concepts and roles in the expected way using induction on their lengths. This completes the definition of the model $\mathcal{I}(\mathcal{B})$ extracted from a branch $\mathcal{B}$.

Let $\prec$ be the ordering on expressions (concepts and roles) of $\mathcal{ALBO}^{\mathsf{id}}$ induced by the rules of $T_{\mathcal{ALBO}^{\mathsf{id}}}$. That is, $\prec$ is the smallest transitive ordering on the set of all $\mathcal{ALBO}^{\mathsf{id}}$ expressions satisfying:

$$\begin{array}{lll}
C \prec \neg C & C \prec C \sqcup D & R \prec \neg R \\
C \prec \exists R.C & D \prec C \sqcup D & R \prec R^{-1} \\
R \prec \exists R.C & \neg C \prec \neg(C \sqcup D) & R \prec R \sqcup S \\
\neg C \prec \neg\exists R.C & \neg D \prec \neg(C \sqcup D) & S \prec R \sqcup S
\end{array}$$

Note that $\prec$ does not coincide with the direct subexpression ordering. $\prec$ is a well-founded ordering, since the length of the expression to the left in a clause is always strictly less than the length of the expression to the right.

LEMMA 5.2. *For every role R and every concept D*

(1) $a : \exists R.\{a'\} \in \mathcal{B}$ implies $(\|a\|, \|a'\|) \in R^{\mathcal{I}(\mathcal{B})}$,
(2) if $(\|a\|, \|a'\|) \in R^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists R.D \in \mathcal{B}$ then $a' : \neg D \in \mathcal{B}$.

PROOF. We prove the statement by induction on the ordering $\prec$ for the relation R. We consider all cases corresponding to the possible forms that a role R can have.

Case $R = Q$. Since $a : \exists Q.\{a'\} \in \mathcal{B}$ implies $(\|a\|, \|a'\|) \in Q^{\mathcal{I}(\mathcal{B})}$ using the definition of $Q^{\mathcal{I}(\mathcal{B})}$, property (1) holds trivially. For (2) let $(\|a\|, \|a'\|) \in Q^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists Q.D \in \mathcal{B}$. Using the definition of $Q^{\mathcal{I}(\mathcal{B})}$, there is an $a'' \sim_{\mathcal{B}} a'$ such that $a : \exists Q.\{a''\} \in \mathcal{B}$. Therefore, by the $(\neg\exists)$ rule, $a'' : \neg D$ is in the branch $\mathcal{B}$, too. Finally, $a' : \neg D \in \mathcal{B}$ by the (mon) rule.

Case $R = \mathsf{id}$. For property (1) let $a : \exists\mathsf{id}.\{a'\} \in \mathcal{B}$. Then $a : \{a'\} \in \mathcal{B}$ by the $(\exists\mathsf{id})$ rule and, hence, $\|a\| = \|a'\|$. Consequently, $(\|a\|, \|a'\|) \in \mathsf{id}^{\mathcal{I}(\mathcal{B})}$. For (2) suppose that $(\|a\|, \|a'\|) \in \mathsf{id}^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists\mathsf{id}.D \in \mathcal{B}$. Hence, $\|a\| = \|a'\|$ and, therefore, $a : \{a'\}$ is in $\mathcal{B}$. Further, by the $(\neg\exists\mathsf{id})$ rule, $a : \neg D \in \mathcal{B}$. Finally, by the (mon) rule, $a' : \neg D$ is in the branch $\mathcal{B}$.

Case $R = S^{-1}$. For property (1) let $a : \exists S^{-1}.\{a'\} \in \mathcal{B}$. Then $a' : \exists S.\{a\} \in \mathcal{B}$ by the rule $(\exists^{-1})$. By the induction hypothesis for $S \prec R$ we have $(\|a'\|, \|a\|) \in S^{\mathcal{I}(\mathcal{B})}$. Consequently, $(\|a\|, \|a'\|) \in (S^{-1})^{\mathcal{I}(\mathcal{B})}$. For (2) suppose that $(\|a\|, \|a'\|) \in (S^{-1})^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists S^{-1}.D \in \mathcal{B}$. As all the occurrences of the inverse operator have been pushed through other role connectives and double occurrences of $^{-1}$ have been removed[4] we can assume that $S = Q$ for some role name Q. Hence, $(\|a'\|, \|a\|) \in Q^{\mathcal{I}(\mathcal{B})}$ and, by the definition of $Q^{\mathcal{I}(\mathcal{B})}$, there is $a'' \sim_{\mathcal{B}} a$ such that $a' : \exists Q.\{a''\} \in \mathcal{B}$. By the (mon) rule, $a'' : \neg\exists Q^{-1}.D \in \mathcal{B}$. Finally, by the $(\neg\exists^{-1})$ rule, $a' : \neg D$ is in the branch $\mathcal{B}$.

Case $R = S_0 \sqcup S_1$. For (1) suppose $a : \exists(S_0 \sqcup S_1).\{a'\} \in \mathcal{B}$. Hence, $a : \exists S_0.\{a'\} \in \mathcal{B}$ or $a : \exists S_1.\{a'\} \in \mathcal{B}$ by the $(\exists\sqcup)$ rule. Thus, by the induction hypothesis for $S_0 \prec R$ and $S_1 \prec R$ we have either $(\|a\|, \|a'\|) \in S_0^{\mathcal{I}(\mathcal{B})}$ or $(\|a\|, \|a'\|) \in S_1^{\mathcal{I}(\mathcal{B})}$. Finally, by the semantics of the relational $\sqcup$ connective we obtain $(\|a\|, \|a'\|) \in (S_0 \sqcup S_1)^{\mathcal{I}(\mathcal{B})}$. For (2) let $(\|a\|, \|a'\|) \in (S_0 \sqcup S_1)^{\mathcal{I}(\mathcal{B})} = S_0^{\mathcal{I}(\mathcal{B})} \cup S_1^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists(S_0 \sqcup S_1).D \in \mathcal{B}$. By the $(\neg\exists\sqcup)$ rule we obtain that both $a : \neg\exists S_0.D$ and $a : \neg\exists S_1.D$ are in $\mathcal{B}$. Therefore, by the induction hypothesis for $S_0 \prec R$ and $S_1 \prec R$ the property (2) holds and, thus, we have $a' : \neg D \in \mathcal{B}$.

Case $R = \neg S$. For (1) suppose $a : \exists\neg S.\{a'\} \in \mathcal{B}$. Then $a : \neg\exists S.\{a'\} \in \mathcal{B}$ is obtained with the $(\exists\neg)$ rule. If $(\|a\|, \|a'\|) \notin (\neg S)^{\mathcal{I}(\mathcal{B})}$ then $(\|a\|, \|a'\|) \in S^{\mathcal{I}(\mathcal{B})}$ and by property (2) which holds by the induction hypothesis for $S \prec R$ we have that $a' : \neg\{a'\}$ is in $\mathcal{B}$. This concept together with $a' : \{a'\}$ implies the branch is closed. We reach a contradiction, so $(\|a\|, \|a'\|) \in (\neg S)^{\mathcal{I}(\mathcal{B})}$.

For property (2) suppose that $(\|a\|, \|a'\|) \in (\neg S)^{\mathcal{I}(\mathcal{B})}$ and $a : \neg\exists\neg S.D$ are in the branch $\mathcal{B}$. Then we have $(\|a\|, \|a'\|) \notin S^{\mathcal{I}(\mathcal{B})}$ and, hence, by the contrapositive of property (1) for $S \prec R$, $a : \exists S.\{a'\}$ is not in $\mathcal{B}$. Applying the $(\neg\exists\neg)$ rule to $a : \neg\exists\neg S.D$ we get $a' : \neg D \in \mathcal{B}$. □

LEMMA 5.3. *If $a : D \in \mathcal{B}$ then $\|a\| \in D^{\mathcal{I}(\mathcal{B})}$ for any concept D.*

PROOF. We prove the lemma by induction on the ordering $\prec$. We consider the following cases for the concept D.

Case $D = A$. We have $a : A \in \mathcal{B} \iff \|a\| \in A^{\mathcal{I}(\mathcal{B})}$ by definition of $A^{\mathcal{I}(\mathcal{B})}$.

Case $D = \{a'\}$. If $a : \{a'\} \in \mathcal{B}$ then, using the definition of $a^{\mathcal{I}(\mathcal{B})}$, $a^{\mathcal{I}(\mathcal{B})} = \|a\| = \|a'\| = a'^{\mathcal{I}(\mathcal{B})}$ and, consequently, $a^{\mathcal{I}(\mathcal{B})} \in \{a'^{\mathcal{I}(\mathcal{B})}\} = \{a'\}^{\mathcal{I}(\mathcal{B})}$.

Case $D = \neg D_0$. If $a : \neg D_0 \in \mathcal{B}$ then $a : D_0 \notin \mathcal{B}$ because otherwise $\mathcal{B}$ would have been closed by the $(\bot)$ rule. We have the following subcases.

If $D_0 = A$ we have $a : A \in \mathcal{B} \iff \|a\| \in A^{\mathcal{I}(\mathcal{B})}$ by definition of $A^{\mathcal{I}(\mathcal{B})}$.

[4] Removing double occurrences of the inverse operator in front of atoms is not essential for the proof but it simplifies matters a bit.

If $D_0 = \{a'\}$, as the rules ($\neg$sym) and (refl) have been applied, $a' : \{a'\}$ is in $\mathcal{B}$ and $a'^{\mathcal{I}(\mathcal{B})} = \|a'\|$. Similarly, $a^{\mathcal{I}(\mathcal{B})} = \|a\|$. Furthermore, because $a : \{a'\} \notin \mathcal{B}$ we have $a \not\sim_{\mathcal{B}} a'$, that is, $a \notin \|a'\| = a'^{\mathcal{I}(\mathcal{B})}$. Consequently, $\|a\| \notin \{\|a'\|\} = \{a'\}^{\mathcal{I}(\mathcal{B})}$.

In case $D_0 = \neg D_1$, if $a : \neg\neg D_1 \in \mathcal{B}$, then $a : D_1 \in \mathcal{B}$ by the ($\neg\neg$) rule. By the induction hypothesis, $\|a\| \in D_1^{\mathcal{I}(\mathcal{B})} = (\neg\neg D_1)^{\mathcal{I}(\mathcal{B})} = D^{\mathcal{I}(\mathcal{B})}$.

In case $D_0 = D_1 \sqcup D_2$ both $a : \neg D_1$ and $a : \neg D_2$ are in $\mathcal{B}$ by the ($\neg\sqcup$) rule. We also have $\neg D_1 \prec \neg(D_1 \sqcup D_2)$ and $\neg D_2 \prec \neg(D_1 \sqcup D_2)$. Hence, by the induction hypothesis $\|a\| \in (\neg D_1)^{\mathcal{I}(\mathcal{B})} \cap (\neg D_2)^{\mathcal{I}(\mathcal{B})} = (\neg(D_1 \sqcup D_2))^{\mathcal{I}(\mathcal{B})}$.

Suppose $D_0 = \exists R.D_1$ and let $\|a'\|$ be an arbitrary element of $\Delta^{\mathcal{I}(\mathcal{B})}$ such that $(\|a\|, \|a'\|) \in R^{\mathcal{I}(\mathcal{B})}$ (if there is no such element then there is nothing to prove). By Lemma 5.2 (2), $a' : \neg D_1 \in \mathcal{B}$. The induction hypothesis gives us $\|a'\| \notin D_1^{\mathcal{I}(\mathcal{B})}$. Finally, we obtain $\|a\| \in (\neg\exists R.D_1)^{\mathcal{I}(\mathcal{B})}$, because a' was chosen arbitrarily.

Case $D = D_0 \sqcup D_1$. If $a : D_0 \sqcup D_1 \in \mathcal{B}$ then either $a : D_0 \in \mathcal{B}$, or $a : D_1 \in \mathcal{B}$ by the ($\sqcup$) rule. Hence, either $\|a\| \in D_0^{\mathcal{I}(\mathcal{B})}$ or $\|a\| \in D_1^{\mathcal{I}(\mathcal{B})}$ by the induction hypothesis for $D_0 \prec D$ and $D_1 \prec D$. Thus, $\|a\| \in D_0^{\mathcal{I}(\mathcal{B})} \cup D_1^{\mathcal{I}(\mathcal{B})} = D^{\mathcal{I}(\mathcal{B})}$.

Case $D = \exists R.D_0$. If $a : \exists R.D_0 \in \mathcal{B}$ then $a' : D_0 \in \mathcal{B}$ and $a : \exists R.\{a'\} \in \mathcal{B}$ for some individual a' by the ($\exists$) rule. By the induction hypothesis, $\|a'\| \in D_0^{\mathcal{I}(\mathcal{B})}$. By Lemma 5.2 (1), we have $(\|a\|, \|a'\|) \in R^{\mathcal{I}(\mathcal{B})}$. That is, $\|a\| \in (\exists R.D_0)^{\mathcal{I}(\mathcal{B})}$. □

A direct consequence of these two lemmas is a stronger form of completeness of the tableau calculus that states existence of a model for an open branch. From this it follows that if the input set is unsatisfiable a closed tableau can be constructed for it.

THEOREM 5.4 (COMPLETENESS). *For any set of concepts $\mathcal{S}$ and any tableau $T_{\mathcal{ALBO}^{\mathsf{id}}}(\mathcal{S})$, if there is an open branch in $T_{\mathcal{ALBO}^{\mathsf{id}}}(\mathcal{S})$ then $\mathcal{S}$ is satisfiable in an $\mathcal{ALBO}^{\mathsf{id}}$-model. This implies $T_{\mathcal{ALBO}^{\mathsf{id}}}$ is a complete tableau calculus for testing satisfiability of concept expressions in $\mathcal{ALBO}^{\mathsf{id}}$.*

Thus, the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$ is sound and complete for reasoning in $\mathcal{ALBO}^{\mathsf{id}}$.

6. UNRESTRICTED BLOCKING

There are satisfiable concepts that can result in an infinite $T_{\mathcal{ALBO}^{\mathsf{id}}}$-tableau, where all open branches are infinite. The concept

$$\neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A$$

is such an example. Since the prefix $\neg\exists(Q' \sqcup \neg Q').\neg$ is equivalent to the universal modality, the concept $a : \exists Q.A$ is propagated to every individual a in every branch of the tableau. The concept $a : \exists Q.A$ 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.

Let us demonstrate how *standard loop checking* (in this case, subset ancestor blocking, see for example Baader et al. [2003]) detects a loop. As we already observed, satisfiability of the concept $\neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A$ corresponds to satisfiability of the concept $\Box\exists Q.A$ in the description logic $\mathcal{ALCO}^{\Box}$, which is $\mathcal{ALCO}$ with the universal modality $\Box$. As a calculus for $\mathcal{ALCO}^{\Box}$ we use the first group of rules from Figure 1 plus the following two rules (where a' in the left rule is uniquely associated with $a : \neg\Box C$).

$$(\neg\Box)\text{: } \frac{a : \neg\Box C}{a' : \neg C}\ (a' \text{ is new}) \qquad\qquad (\Box)\text{: } \frac{a : \Box C,\ \ a' : \{a'\}}{a' : C}$$

1.	$a_0 : \Box\exists Q.A$	given
2.	$a_0 : \{a_0\}$	(refl),1
3.	$a_0 : \exists Q.A$	($\Box$),1,2
4.	$a_1 : A$	($\exists$),3
5.	$a_0 : \exists Q.\{a_1\}$	($\exists$),3
6.	$a_1 : \{a_1\}$	(refl),4
7.	$a_1 : \exists Q.A$	($\Box$),1,6
8.	$a_0 \not\sim a_1$	Loop checking: $A \in \mathcal{L}(a_1) \setminus \mathcal{L}(a_0)$
9.	$a_2 : A$	($\exists$),7
10.	$a_1 : \exists Q.\{a_2\}$	($\exists$),7
11.	$a_2 : \{a_2\}$	(refl),9
12.	$a_2 : \exists Q.A$	($\Box$),1,11
13.	$a_1 \sim a_2$	Loop checking: $\mathcal{L}(a_1) \supseteq \mathcal{L}(a_2)$

Fig. 3. Standard loop checking mechanism in $\mathcal{ALCO}^{\Box}$.

Similar to the proofs in Section 5 it can be proved that this calculus is sound and complete for satisfiability problem in $\mathcal{ALCO}^{\Box}$.

Figure 3 gives a derivation in this calculus using loop checking based on standard subset ancestor blocking. For any individual a, let $\mathcal{L}(a)$ be a set of concepts associated with a in the current branch. More precisely, it is defined by

$$\mathcal{L}(a) \stackrel{\text{def}}{=} \{C \mid a : C \text{ is in the current branch, and } C \neq \{a\}\}.$$

After all the type-completing rules have been applied to all concept expressions labelled with a specific individual, loop checking tests are performed relative to an ancestor individual. Two loop checking tests are performed, namely in step 8 and step 13. Consider step 13. All the type-completing rules have been applied to all concept expressions of the form $a_2 : C$ and $a_1 : C$. Comparison of the sets of concepts $\mathcal{L}(a_1)$ and $\mathcal{L}(a_2)$ associated with a_1 and a_2 in the loop checking test shows that $\mathcal{L}(a_1)$ subsumes $\mathcal{L}(a_2)$. Thus they are in a subset relationship as indicated, and consequently the individuals a_1 and a_2 can be identified. At step 8 the sets $\mathcal{L}(a_0)$ and $\mathcal{L}(a_1)$ are not in a subset relationship because $a_0 : A$ is not present in the branch. The derivation therefore cannot yet stop, but does in step 13.

This example illustrates one of the simplest forms of standard loop checking used in description and modal logic tableau procedures. Other forms of loop-checking have been devised for different logics. One can classify the different existing loop-checking mechanisms as using a combination of these blocking techniques: subset or equality blocking, non-pairwise or pairwise blocking, ancestor or anywhere blocking, and static or dynamic blocking, see for example [Baader and Sattler 2001]. These techniques are all based on comparing sets of concepts labelled by some individuals (and in the case of pairwise blocking also sets of roles associated to predecessors). It is not clear whether these forms of loop checking are sufficient to handle role negation though.

Suppose that, at some node of a branch, all the type-completing rules have been applied to concepts labelled with individuals a_0 and a_1. Following the standard non-pairwise blocking procedure we can identify a_0 and a_1 if they both label the same set of concepts. However, it is not correct to start identifying the individuals at this point in the derivation as there could be, for example, a concept $\exists S.\neg\exists\neg R.C$ involving role negation, where R and C are expressions that are suitably complex. At a subsequent point the ($\exists$) rule is applied to this concept and, hence, the expression $a_2 : \neg\exists\neg R.C$, where a_2 is new, appears in the branch. This triggers the application of the ($\neg\exists\neg$) rule. Because of the form of R and $\neg C$ in the two branches it also triggers application of the ($\neg\exists$) rule and possibly other type-completing rules. After such applications, it can

1.	$a_0 : \exists Q_0.A$	given
2.	$a_0 : \exists Q_1.A$	given
3.	$a_0 : \exists Q_2.\neg\exists Q_2.\exists Q_1^{-1}.(A' \sqcup \neg A')$	given
4.	$a_0 : \neg\exists Q_2.\exists\neg Q_2.\neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	given
5.	$a_1 : A$	$(\exists)$,1
6.	$a_0 : \exists Q_0.\{a_1\}$	$(\exists)$,1
7.	$a_1 : \{a_1\}$	(refl),5
8.	$a_2 : A$	$(\exists)$,2
9.	$a_0 : \exists Q_1.\{a_2\}$	$(\exists)$,2
10.	$a_2 : \{a_2\}$	(refl),8
11.	$a_3 : \neg\exists Q_2.\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\exists)$,3
12.	$a_0 : \exists Q_2.\{a_3\}$	$(\exists)$,3
13.	$a_3 : \neg\exists\neg Q_2.\neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\exists)$,4,12
14.	$\blacktriangleright a_3 : \exists Q_2.\{a_2\}$	$(\neg\exists\neg)$,13,10
15.	$a_2 : \neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\exists)$,11,14
16.	$a_0 : \neg(A' \sqcup \neg A')$	$(\neg\exists^{-1})$,15,9
17.	Unsatisfiable	after a few steps
18.	$\blacktriangleright a_2 : \neg\neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\exists\neg)$,13,10
19.	$a_2 : \exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\neg)$,18
20.	$\blacktriangleright a_3 : \exists Q_2.\{a_1\}$	$(\neg\exists\neg)$,13,7
21.	$a_1 : \neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\exists)$,11,20
...		
22.	$\blacktriangleright a_1 : \neg\neg\exists Q_1^{-1}.(A' \sqcup \neg A')$	$(\neg\exists\neg)$,13,7
...		

Fig. 4. Global effect of the introduction of a new individual

happen that the labels are not identifiable (anymore) because their types have become distinguishable by some concept. That is, any introduction of a new individual in a tableau has, in general, a global effect on the provisional model constructed so far.

The example in Figure 4 illustrates this global effect. Here we are interested in the satisfiability of

$$(\exists Q_0.A) \sqcap (\exists Q_1.A) \sqcap (\exists Q_2.\forall Q_2.\forall Q_1^{-1}.\bot) \sqcap (\forall Q_2.\overline{\forall} Q_2.\forall Q_1^{-1}.\bot).$$

(Recall that $\forall Q_2.C = \neg\exists Q_2.\neg C$ and $\overline{\forall} Q_2.C = \neg\exists\neg Q_2.C$.) At step 10 none of the type-completing rules need to be applied to concepts labelled with a_1 and a_2. Although at this point a_1 and a_2 are labels of the same subconcepts of the given concepts, we cannot make them equal (using equality anywhere blocking). The reason is that in 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 a_1 and a_2 are now distinguished by the concept $\exists Q_1^{-1}.(A' \sqcup \neg A')$. For this example it is certainly not possible to use standard static blocking techniques to identify individuals (a_1 and a_2) although they are locally complete (that is, none of the type completing rules can be applied to them), because other individuals (a_0) can influence a_1 and a_2 via the right branch of the $(\neg\exists\neg)$ rule.

Both examples illustrate a reason for non-termination of $T_{\mathcal{ALCBO}}^{\mathsf{id}}$ is the possible infinite generation of labels. Although the rules respect the well-founded ordering $\prec$ they do it only with respect to expressions, not labelled expressions (assertions). Furthermore, it is easy to see that only applications of the $(\exists)$ rule generate new individuals in the branch. Thus, a reason that a branch can be infinite is the unlimited application of the $(\exists)$ rule; it is in fact the only possible reason. It is crucial therefore to have a

blocking mechanism that avoids infinite derivations by restricting the application of the ($\exists$) rule.

In order to turn the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$ into a terminating calculus for $\mathcal{ALBO}^{\mathsf{id}}$, we introduce unrestricted 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. That is, let $a < a'$, whenever the appearance of $a' : \{a'\}$ in the branch is strictly later than the appearance of $a : \{a\}$.

We add the following rule, called the *unrestricted blocking* rule, to the calculus.

$$\text{(ub):}\ \frac{a : \{a\},\ \ a' : \{a'\}}{a : \{a'\} \mid a : \neg\{a'\}}$$

Moreover, we require that the following conditions both hold.

(c1) If $a : \{a'\}$ appears in a branch and $a < a'$ then possible applications of the ($\exists$) rule to expressions of the form $a' : \exists R.C$ are not performed within the branch.

(c2) 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.

The intuition of the (ub) rule is that it conjectures whether two labels are equal or not. In the left branch two labels are set to be equal. If this does not lead to finding a model then the labels cannot be equal. This is the information carried by the right branch. Conditions (c1) and (c2) are important to restrict the application of the ($\exists$) rule. Condition (c1) specifies that it may only be applied to labelled expressions where the label is the smallest representative of an equivalence class. Condition (c2) says that from some point onwards in a branch blocking has been applied exhaustively before the application of the ($\exists$) rule.

We use the notation $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) for the extension of $T_{\mathcal{ALBO}^{\mathsf{id}}}$ with this blocking mechanism using the (ub) rule.

THEOREM 6.1. *$T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) is a sound and complete tableau calculus for $\mathcal{ALBO}^{\mathsf{id}}$.*

PROOF. The (ub) rule is sound in the usual sense (see definition at the beginning of Section 5). The blocking conditions (c1) and (c2) are sound in the sense that they cannot cause an open branch to become closed. Since adding a sound tableau rule to a tableau calculus preserves soundness and completeness of the calculus, the blocking rule and the blocking conditions can be safely added to *any* tableau calculus without endangering soundness or completeness. As $T_{\mathcal{ALBO}^{\mathsf{id}}}$ is sound and complete, it follows that $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) is sound and complete. □

7. TERMINATION THROUGH UNRESTRICTED BLOCKING

Next we prove termination. Since every finite set of concepts can be replaced by the conjunction of its elements, without loss of generality in this section and the next section, we restrict ourselves to a single input concept.

For every set X, let $\mathsf{Card}(X)$ denote the *cardinality* of X. Let $\preceq$ be the reflexive closure of the ordering $\prec$ defined in Section 5. For every concept D we define $\mathsf{sub}(D) \stackrel{\mathsf{def}}{=} \{D' \mid D' \preceq D\}$.

Let C be the given input concept and n be the length of the concept C. Suppose $\mathcal{B}$ is an arbitrary open branch in a $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) tableau for C and $\mathcal{I}(\mathcal{B})$ be the model constructed from $\mathcal{B}$ as described in Section 5.

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

LEMMA 7.1. *$\#^{\exists}(\|a\|)$ is finite for every $\|a\| \in \Delta^{\mathcal{I}(\mathcal{B})}$.*

PROOF. Suppose not, that is, suppose $\#^{\exists}(\|a\|)$ is infinite. Hence, there is an infinite sequence of concepts $a_0 : \exists R_0.D_0, a_1 : \exists R_1.D_1, \ldots$ with each $a_i \in \|a\|$ to which the ($\exists$) rule has been applied in the branch $\mathcal{B}$. Because the ($\exists$) rule is restricted from being applied to role assertions (that is, concepts of the form $a : \exists R.\{a'\}$) we can assume none of the D_i are singletons. Consequently, each concept $\exists R_i.D_i$ belongs to $\mathsf{sub}(C)$ (which is finite). Hence, since the above sequence is infinite, there must be infinitely many a_i in it. Without loss of generality, we can assume $a < a_0 < a_1 < \cdots$. By Condition (c2), there is a node in $\mathcal{B}$ from which onwards all possible applications of the (ub) rule are performed before any application of the ($\exists$) rule. Because $a < a_0 < a_1 < \cdots$ there is an a_i such that $a_i : \{a_i\}$ appears in a next node in $\mathcal{B}$. Since the priority of the (refl) rule is higher than the priority of the ($\exists$) rule, the ($\exists$) rule is not applied to $a_i : \exists R_i.D_i$ before this node. This means the (ub) rule is applied to $a : \{a\}$ and $a_i : \{a_i\}$ in $\mathcal{B}$ before any next application of the ($\exists$) rule. Because $\mathcal{B}$ is open and $a_i \in \|a\|$, this application cannot introduce $a : \neg\{a_i\}$ into $\mathcal{B}$. Thus, $a : \{a_i\}$ is introduced into $\mathcal{B}$ by this application of the (ub) rule. Therefore, because of Condition (c1), a_i is immediately blocked for any application of the ($\exists$) rule. In particular, the ($\exists$) rule is not allowed to be applied to the concept $a_i : \exists R_i.D_i$ which is a contradiction with our assumptions about the sequence $a_0 : \exists R_0.D_0, a_1 : \exists R_1.D_1, \ldots$. □

Let $\#^{\exists}(\mathcal{B})$ denote the number of applications of the ($\exists$) rule in a branch $\mathcal{B}$ and let k be the number of individuals occurring in the input concept assertion $a : C$. Since the ($\exists$) rule is the only rule that generates new individuals, $\#^{\exists}(\mathcal{B})$ is in fact the number of generated individuals in the branch $\mathcal{B}$ and the number of all individuals occurring in $\mathcal{B}$ is $k + \#^{\exists}(\mathcal{B})$. For the number of all individuals occurring in $\mathcal{B}$ we introduce the notation $i(\mathcal{B})$, that is, $i(\mathcal{B}) \stackrel{\text{def}}{=} k + \#^{\exists}(\mathcal{B})$.

Let $M(\mathcal{B})$ be the least cardinal, which is the upper bound of all $\#^{\exists}(\|a\|)$, where $\|a\|$ ranges all the equivalence classes from $\Delta^{\mathcal{I}(\mathcal{B})}$. That is:

$$M(\mathcal{B}) \stackrel{\text{def}}{=} \sup\{\#^{\exists}(\|a\|) \mid \|a\| \in \Delta^{\mathcal{I}(\mathcal{B})}\}.$$

The following lemma establishing the upper bound for $\#^{\exists}(\mathcal{B})$ is easy to prove.

LEMMA 7.2. $\#^{\exists}(\mathcal{B}) \leq M(\mathcal{B}) \cdot \mathsf{Card}(\Delta^{\mathcal{I}(\mathcal{B})})$.

Furthermore, the following lemma holds.

LEMMA 7.3. *The number of concepts in $\mathcal{B}$ does not exceed $n \cdot i(\mathcal{B}) + 2n \cdot i(\mathcal{B})^2$. In particular, if $\#^{\exists}(\mathcal{B})$ is finite then $\mathcal{B}$ is finite.*

PROOF. Let $D \in \mathcal{B}$. Then D has one of the following forms:

$$\begin{aligned}
&D = a : D', \quad \text{where } D' \text{ is a concept in } \mathsf{sub}(C),\\
&D = a : \{a'\},\\
&D = a : \neg\{a'\},\\
&D = a : \exists R.\{a'\}, \quad \text{where } R \text{ is a role in } \mathsf{sub}(C),\\
&D = a : \neg\exists R.\{a'\}, \quad \text{where } R \text{ is a role in } \mathsf{sub}(C).
\end{aligned}$$

The number of individuals and concepts in $\mathsf{sub}(C)$ is bounded by n and the number of roles in $\mathsf{sub}(C)$ is strictly less than n. Hence, an upper bound for the number of concepts in a branch $\mathcal{B}$ can be given as: $n \cdot i(\mathcal{B}) + i(\mathcal{B})^2 + i(\mathcal{B})^2 + (n-1) \cdot i(\mathcal{B})^2 + (n-1) \cdot i(\mathcal{B})^2 = n \cdot i(\mathcal{B}) + 2n \cdot i(\mathcal{B})^2$. □

Clearly if $\Delta^{\mathcal{I}(\mathcal{B})}$ is finite then $M(\mathcal{B}) = \max\{\#^{\exists}(\|a\|) \mid \|a\| \in \Delta^{\mathcal{I}(\mathcal{B})}\}$. Therefore, as a consequence of Lemmas 7.2 and 7.3, we obtain the following statement.

COROLLARY 7.4. *Let $\mathcal{B}$ be an open branch in a $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) tableau for a concept C. Then, $\mathcal{B}$ is finite iff $\Delta^{\mathcal{I}(\mathcal{B})}$ is finite.*

LEMMA 7.5. *Assume that the input concept C is satisfiable in a model $\mathcal{J}$. Then there is an open branch $\mathcal{B}$ in $T_{\mathcal{ALBO}^{\mathsf{id}}}(ub)(C)$ such that* $\mathsf{Card}(\Delta^{\mathcal{I}(\mathcal{B})}) \leq \mathsf{Card}(\Delta^{\mathcal{J}})$.

PROOF. By induction on the application of the rules in $T_{\mathcal{ALBO}^{\mathsf{id}}}(ub)(C)$ we choose an appropriate branch $\mathcal{B}$ and construct a model $\mathcal{J}'$ that differs from $\mathcal{J}$ only in the interpretations of individuals occurring in $\mathcal{B}$ but not occurring in C, so that $\mathcal{J}' \models a : D$ for every concept $a : D$ from $\mathcal{B}$.

More precisely, let $\Delta^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} \Delta^{\mathcal{J}}$ and $A^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} A^{\mathcal{J}}$ and $Q^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} Q^{\mathcal{J}}$ for every concept symbol A and role symbol Q. Additionally let $a^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} a^{\mathcal{J}}$ for every individual a occurring in C. Our aim is to find a branch $\mathcal{B}$ in $T_{\mathcal{ALBO}^{\mathsf{id}}}(ub)(C)$ that is given by a sequence of nodes $\mathcal{S}_0, \ldots, \mathcal{S}_n, \ldots$ and, for each n, to extend the interpretation $\mathcal{J}'$ to the individuals in $\mathcal{S}_n$ such that $\mathcal{J}' \models a : D$ for each $a : D$ in $\mathcal{S}_n$. It follows by an induction argument that $\mathcal{J}' \models a : D$ for each $a : D$ from $\mathcal{B}$.

For the base case we choose the root node of $T_{\mathcal{ALBO}^{\mathsf{id}}}(ub)(C)$ as the initial node $\mathcal{S}_0$ of the branch $\mathcal{B}$. That is, $\mathcal{S}_0 = \{a_0 : C\}$, where a_0 is a new individual. Since C is satisfiable in $\mathcal{J}$, there is an element w in $C^{\mathcal{J}'}$. We let $a_0^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} w$.

The induction step involves considering cases for each of the rules in the $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) calculus. We consider only two rules.

Case of the ($\exists$) rule. Suppose $a : \exists R.D$ is in node $\mathcal{S}_n$ and $a : \exists R.\{a'\}$ and $a' : D$ are the conclusions of the rule. We set $\mathcal{S}_{n+1} \stackrel{\mathsf{def}}{=} \mathcal{S}_n \cup \{a : \exists R.\{a'\}, a' : D\}$. By the induction hypothesis we have that $\mathcal{J}' \models a : \exists R.D$. That is, there is an element w in $\Delta^{\mathcal{J}'}$ such that $(a^{\mathcal{J}'}, w) \in R^{\mathcal{J}'}$ and $w \in D^{\mathcal{J}'}$. We set $a'^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} w$.

Case of the (ub) rule. Suppose $a : \{a\}$ and $a' : \{a'\}$ are in node $\mathcal{S}_n$. By the induction hypothesis $a^{\mathcal{J}'}$ and $a'^{\mathcal{J}'}$ are both defined. We have two cases: either $a^{\mathcal{J}'} = a'^{\mathcal{J}'}$ or $a^{\mathcal{J}'} \neq a'^{\mathcal{J}'}$. We let $\mathcal{S}_{n+1} \stackrel{\mathsf{def}}{=} \mathcal{S}_n \cup \{a : \{a'\}\}$ in the first case and $\mathcal{S}_{n+1} \stackrel{\mathsf{def}}{=} \mathcal{S}_n \cup \{a : \neg\{a'\}\}$ in the second case.

Thus, we have constructed an appropriate branch $\mathcal{B}$. In order to complete the construction of $\mathcal{J}'$ we set $a^{\mathcal{J}'} \stackrel{\mathsf{def}}{=} a^{\mathcal{J}}$ for every individual a that does not occur in the constructed branch $\mathcal{B}$. Clearly, $\mathcal{J}' \models a : D$ for each $a : D$ from $\mathcal{B}$. $\mathcal{B}$ is open since the set of all concepts from $\mathcal{B}$ is satisfiable in the model $\mathcal{J}'$. Now let f be a function mapping $\Delta^{\mathcal{J}'}$ ($= \Delta^{\mathcal{J}}$) to $\Delta^{\mathcal{I}(\mathcal{B})}$ and be defined by:

$$f(w) \stackrel{\mathsf{def}}{=} \begin{cases} \|a\|, & \text{if } w = a^{\mathcal{J}'} \text{ for some individual } a \text{ in } \mathcal{B}, \\ \text{arbitrary in } \Delta^{\mathcal{I}(\mathcal{B})}, & \text{otherwise.} \end{cases}$$

By construction of $\mathcal{J}'$ if $w = a^{\mathcal{J}'} = a'^{\mathcal{J}'}$ for two different individuals a and a' from the branch $\mathcal{B}$ then $a : \{a'\} \in \mathcal{B}$ and, hence, $\|a\| = \|a'\|$. The function f is thus defined correctly. Furthermore, f is onto $\Delta^{\mathcal{I}(\mathcal{B})}$ because $\Delta^{\mathcal{I}(\mathcal{B})} = \{\|a\| \mid a : \{a\} \in \mathcal{B}\}$. □

Recall the model bounding function $\mu_0(n)$ obtained for $\mathcal{ALBO}^{\mathsf{id}}$ in Theorem 3.4: $\mu_0(n) = 9(n\lfloor\log(3n+1)\rfloor) \cdot 8^n$. Combining Lemma 7.5 with Theorem 3.4 we obtain the following theorem.

THEOREM 7.6. *Let C be a satisfiable concept with length n. Then in every $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub)-tableau for C, there exists an open branch $\mathcal{B}$ such that* $\mathsf{Card}(\Delta^{\mathcal{I}(\mathcal{B})}) \leq \mu_0(n)$.

1.	$a_0 : \neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A$	given
2.	$a_0 : \{a_0\}$	(refl),1
3.	$a_0 : \neg\exists Q'.\neg\exists Q.A$	($\neg\exists\sqcup$),1
4.	$a_0 : \neg\exists\neg Q'.\neg\exists Q.A$	($\neg\exists\sqcup$),1
5.	$\blacktriangleright a_0 : \exists Q'.\{a_0\}$	($\neg\exists\neg$),4,2
6.	$a_0 : \neg\neg\exists Q.A$	($\neg\exists$),3,5
7.	$a_0 : \exists Q.A$	($\neg\neg$),6
8.	$a_0 : \exists Q.\{a_1\}$	($\exists$),7
9.	$a_1 : A$	($\exists$),7
10.	$a_1 : \{a_1\}$	(refl),9
11.	$\blacktriangleright a_0 : \exists Q'.\{a_1\}$	($\neg\exists\neg$),4,10
12.	$a_1 : \neg\neg\exists Q.A$	($\neg\exists$),3,11
13.	$a_1 : \exists Q.A$	($\neg\neg$),12
14.	$a_1 : \exists Q.\{a_2\}$	($\exists$),13
15.	$a_2 : A$	($\exists$),13
16.	$a_2 : \{a_2\}$	(refl),9
17.	$\blacktriangleright a_0 : \{a_1\}$	(ub),2,10
...	**Non-terminating**	Similarly to 11–17 with a_2 in place of a_1
18.	$\blacktriangleright a_0 : \neg\{a_1\}$	(ub),2,10
...		
19.	$\blacktriangleright a_1 : \neg\neg\exists Q.A$	($\neg\exists\neg$),4,10
...		
20.	$\blacktriangleright a_0 : \neg\neg\exists Q.A$	($\neg\exists\neg$),4,2
...		

Fig. 5. Effect of omitting Condition (c2)

THEOREM 7.7 (TERMINATION). *$T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) is a terminating tableau calculus for satisfiability in $\mathcal{ALBO}^{\mathsf{id}}$.*

PROOF. In any $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub)-tableau, every closed branch is finite. Suppose there is an open branch in the tableau. By the Completeness Theorem (Theorem 5.4) this means the input concept is satisfiable. By Theorem 7.6 and Corollary 7.4 there is a finite open branch in the tableau. □

Notice that Condition (c2) is essential for ensuring termination of a $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) derivation. Figure 5 shows that without (c2) the $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) tableau for the concept

$$\neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A$$

would not terminate because new individuals are generated more often than the equality conjectures made via the (ub) rule. Furthermore, if Condition (c1) is omitted, then any $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) tableau for the same concept does not terminate either. Thus, Conditions (c1) and (c2) are both necessary for termination of the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub).

8. A COMPLEXITY-OPTIMAL TABLEAU DECISION PROCEDURE

Our estimations, in this section, for the complexity of tableau procedures based on the $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) calculus rely on the following two observations. First, from Theorem 7.6 it follows that, in a decision procedure based on $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub), we can ignore any branch $\mathcal{B}$ where the cardinality of $\Delta^{\mathcal{I}(\mathcal{B})}$ is larger than $\mu_0(n)$. Any such branch is either closed or there is an open branch with a smaller number of classes of equal individuals.

Second, Lemmas 7.2 and 7.3 provide a clue for estimating the number of derivation steps (that is, the number of rule applications) in a shortest open branch $\mathcal{B}$. Since the

maximal number of premises of a rule is two, and none of the rules are applied to the same set of premises more than once, if N is the number of concepts in $\mathcal{B}$, then the number of derivation steps in $\mathcal{B}$ is less than or equal to N^2. Thus, the following lemma holds.

LEMMA 8.1. *Let $\mathcal{B}$ be a finite open branch in $T_{\mathcal{ALBO}^{\mathsf{id}}}(\text{ub})(C)$. Then the number of derivation steps in $\mathcal{B}$ does not exceed*

$$(*) \qquad \Big(n \cdot (k + M(\mathcal{B}) \cdot \mu_0(n)) + 2n \cdot (k + M(\mathcal{B}) \cdot \mu_0(n))^2\Big)^2 .$$

In order to define a complexity-optimal tableau decision procedure, let us define the following derivation strategy.

> *Avoid huge branch strategy*: Limit the number of derivation steps in every branch of a tableau derivation to the bound $(*)$ in Lemma 8.1 and do not expand branches that become longer.

Whereas n and k depend only on the given concept C, the value $M(\mathcal{B})$ depends also on the particular strategy of applying the (ub) rule. In fact, the following lemma holds.

LEMMA 8.2. *In a branch $\mathcal{B}$ let M' be the number of individuals occurring strictly before the node from which point onwards, before any application of the* $(\exists)$ *rule, all applications of the* (ub) *rule have been performed in accordance with Condition (c2). Let n' be the number of concepts of the form $\exists R.D$ in* $\mathsf{sub}(C)$. *Then $M(\mathcal{B}) \leq M' + n'$.*

PROOF (SKETCH). It can be seen from the proof of Lemma 7.1 that, after the node where exhaustive applications of the (ub) rule started, the number of applications of the $(\exists)$ rule to concepts labelled by individuals from the same equivalence class in the branch $\mathcal{B}$ is restricted by n'. The number of applications of the $(\exists)$ rule to concepts before this node is restricted by the number of individuals present before that node, that is, by M'. □

We note that $n' \leq n$, but, depending on the strategy used, M' can have any value. However, if a tableau strategy ensures that M' is at most exponential in the length of the given concept C then the number of derivation steps in the branch $\mathcal{B}$ is also at most exponential in the length of C. This is in particular true if Condition (c2) is satisfied for the *first* node of the tableau. In this case, $M' = 0$. This proves the following theorem.

THEOREM 8.3. *The calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}(\text{ub})$ provides a non-deterministic tableau procedure running in* NExpTime*, when it uses the 'avoid huge branch strategy' together with the tableau expansion strategy ensuring that before any application of the* $(\exists)$ *rule all possible applications of the* (ub) *rule have been performed.*

This gives a complexity optimal tableau decision procedure for $\mathcal{ALBO}^{\mathsf{id}}$.

9. DETERMINISTIC DECISION PROCEDURES

We have presented a sound, complete and terminating tableau calculus for $\mathcal{ALBO}^{\mathsf{id}}$. The calculus defines the rules, we have defined how a tableau derivation may be constructed and how rules may be applied, but it has not been specified in concrete terms how the tableau derivation must be constructed and how the rules must be applied. The presented tableau calculi define only non-deterministic decision procedures.

When implementing a calculus as a deterministic decision procedure an important issue therefore is to decide how to perform the search without losing the crucial properties of soundness, completeness and termination. As all the rules in the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}(\text{ub})$ are sound, preserving soundness is not problematic. Care is needed for preserving completeness and termination. In particular, it is crucial that the search is

performed 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 $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) may not defer the use of an applicable rule indefinitely. In our context fairness must be understood 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 makes both the branch selection, and the selection of expressions to which to apply a rule, in a fair way. Then soundness, completeness and termination carry over from the calculus and we have:

THEOREM 9.1. *Any fair tableau procedure based on* $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) *is a decision procedure for* $\mathcal{ALBO}^{\mathsf{id}}$ *and all its sublogics.*

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

$$\neg\Big((\exists(Q' \sqcup \neg Q').\neg\exists Q_0.A) \sqcup (\neg\exists Q_1.\neg\exists Q_0.A)\Big),$$

or equivalently $(\Box\exists Q_0.A) \sqcap (\exists Q_1.\forall Q_0.\neg A)$, is not satisfiable. Figure 6 gives a depth-first left-to-right derivation that is unfair and does not terminate. It can be seen that the derivation is infinite because the application of the ($\exists$) rule to $a_0 : \exists Q_1.\neg\exists Q_0.A$ is deferred forever and, consequently, a contradiction is not found. This illustrates the importance of fairness for completeness.

The next example illustrates the importance of fairness for termination. The concept

$$\neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A,$$

or equivalently $\Box\exists Q.A$, is satisfiable. The derivation in Figure 7 is obtained with a depth-first *right-to-left* strategy. However, the repeated selection of the right branch at (ub) choice points, in particular, causes all the individuals in the branch to be pairwise non-equal. The concept $a : \exists Q.A$ re-appears repeatedly for every individual a 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 right-most branch get ignored. In the branches right-most with respect to the (ub) rule, it is as if blocking with the (ub) rule has never been applied.

Recall from Section 4, a tableau calculus is terminating iff the tableau constructed for any finite concept set $\mathcal{S}$ is finite whenever $T(\mathcal{S})$ is closed or contains a finite open branch if the tableau is open. For a satisfiable concept set this means it is possible to construct *a* finite, fully expanded, open branch. Not all branches in an open terminating tableau need to be finite though.

An immediate way of obtaining a fair deterministic decision procedure for a terminating tableau is to use breadth-first search. Another possibility is to use depth-first search with iterative deepening. The idea of iterative deepening search is that the tableau is expanded up to a fixed depth. If a closed tableau is found or if an open fully expanded branch is found then the derivation process stops. If not, then the depth is increased and the tableau is constructed up to this depth, and so on. A benefit of breadth-first and depth-first iterative deepening strategies is the generation of small models.

Additionally, depth-first search combined with the 'avoid huge branch strategy' provides a fair, deterministic decision procedure.

THEOREM 9.2. *The following are deterministic decision procedures for* $\mathcal{ALBO}^{\mathsf{id}}$ *and all its sublogics:*

(*1*) *Any fair tableau procedure based on* $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) *using a breadth-first search strategy.*

1.	$a_0 : \neg\,(\exists(Q' \sqcup \neg Q').\neg\exists Q_0.A \sqcup \neg\exists Q_1.\neg\exists Q_0.A)$	given
2.	$a_0 : \{a_0\}$	(refl),1
3.	$a_0 : \neg\exists(Q' \sqcup \neg Q').\neg\exists Q_0.A$	($\neg\sqcup$),1
4.	$a_0 : \neg\neg\exists Q_1.\neg\exists Q_0.A$	($\neg\sqcup$),1
5.	$a_0 : \exists Q_1.\neg\exists Q_0.A$	($\neg\neg$),4
6.	$a_0 : \neg\exists Q'.\neg\exists Q_0.A$	($\neg\exists\sqcup$),3
7.	$a_0 : \neg\exists\neg Q'.\neg\exists Q_0.A$	($\neg\exists\sqcup$),3
8.	$\blacktriangleright a_0 : \exists Q'.\{a_0\}$	($\neg\exists\neg$),7,2
9.	$a_0 : \neg\neg\exists Q_0.A$	($\neg\exists$),8,6
10.	$a_0 : \exists Q_0.A$	($\neg\neg$),9
11.	$a_1 : A$	($\exists$),10
12.	$a_0 : \exists Q_0.\{a_1\}$	($\exists$),10
13.	$a_1 : \{a_1\}$	(refl),11
14.	$\blacktriangleright a_0 : \exists Q'.\{a_1\}$	($\neg\exists\neg$),7,13
15.	$a_1 : \neg\neg\exists Q_0.A$	($\neg\exists$),14,6
16.	$a_1 : \exists Q_0.A$	($\neg\neg$),15
17.	$\blacktriangleright a_0 : \{a_1\}$	(ub),2,13
18.	$a_2 : \neg\exists Q_0.A$	($\exists$),5
19.	$a_0 : \exists Q_1.\{a_2\}$	($\exists$),5
20.	$a_2 : \{a_2\}$	(refl),18
21.	$\blacktriangleright a_0 : \exists Q'.\{a_2\}$	($\neg\exists\neg$),7,20
22.	$a_2 : \neg\neg\exists Q_0.A$	($\neg\exists$),6,21
23.	Unsatisfiable	($\bot$),18,22
24.	$\blacktriangleright a_2 : \neg\neg\exists Q_0.A$	($\neg\exists\neg$),7,20
25.	Unsatisfiable	($\bot$),18,24
26.	$\blacktriangleright a_0 : \neg\{a_1\}$	(ub),2,13
27.	$a_2 : A$	($\exists$),16
28.	$a_1 : \exists Q_0.\{a_2\}$	($\exists$),16
29.	$a_2 : \{a_2\}$	(refl),27
30.	$\blacktriangleright a_0 : \exists Q'.\{a_2\}$	($\neg\exists\neg$),7,29
31.	$a_2 : \neg\neg\exists Q_0.A$	($\neg\exists$),30,6
32.	$a_2 : \exists Q_0.A$	($\neg\neg$),31
...	**Non-terminating**	Repetition of 16–32
33.	$\blacktriangleright a_2 : \neg\neg\exists Q_0.A$	($\neg\exists\neg$),7,29
34.		Similarly to 30–32
35.	$\blacktriangleright a_1 : \neg\neg\exists Q_0.A$	($\neg\exists\neg$),7,13
36.		Similarly to 14–34
37.	$\blacktriangleright a_0 : \neg\neg\exists Q_0.A$	($\neg\exists\neg$),7,2
38.		Similarly to 8–36

Fig. 6. An infinite derivation, due to unfair selection of concepts

(*2*) *Any fair tableau procedure based on* $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) *using a depth-first search strategy with iterative deepening.*
(*3*) *Any fair tableau procedure based on* $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) *using a depth-first left-to-right strategy and the 'avoid huge branch strategy'.*

Whether a depth-first left-to-right strategy without exploiting the model bound gives a decision procedure for $\mathcal{ALBO}^{\mathsf{id}}$ is open. In the context of the two-variable fragment for first-order logic Reker [2011] shows that depth-first left-to-right derivations need not terminate if the unrestricted blocking mechanism is used and termination is difficult to ensure. It is shown that, when constructing a tableau in a depth-first manner, that

1.	$a_0 : \neg\exists(Q' \sqcup \neg Q').\neg\exists Q.A$	given
2.	$a_0 : \{a_0\}$	(refl),1
3.	$a_0 : \neg\exists Q'.\neg\exists Q.A$	($\neg\exists\sqcup$),1
4.	$a_0 : \neg\exists\neg Q'.\neg\exists Q.A$	($\neg\exists\sqcup$),1
5.	$\blacktriangleright a_0 : \neg\neg\exists Q.A$	($\neg\exists\neg$),4,2
6.	$a_0 : \exists Q.A$	($\neg\neg$),5
7.	$a_1 : A$	($\exists$),6
8.	$a_0 : \exists Q.\{a_1\}$	($\exists$),6
9.	$a_1 : \{a_1\}$	(refl),7
10.	$\blacktriangleright a_1 : \neg\neg\exists Q.A$	($\neg\exists\neg$),4,9
11.	$a_1 : \exists Q.A$	($\neg\neg$),10
12.	$\blacktriangleright a_0 : \neg\{a_1\}$	(ub),2,9
13.	$a_2 : A$	($\exists$),11
14.	$a_1 : \exists Q.\{a_2\}$	($\exists$),11
...	**Non-terminating**	Repetition of 7–14
15.	$\blacktriangleright a_0 : \{a_1\}$	(ub),2,9
...		Never expanded
16.	$\blacktriangleright a_0 : \exists Q'.\{a_1\}$	($\neg\exists\neg$),4,9
...		Never expanded
17.	$\blacktriangleright a_0 : \exists Q'.\{a_0\}$	($\neg\exists\neg$),4,2
...		Never expanded

Fig. 7. An infinite derivation, due to unfair selection of branches

very early blocking of terms is crucial for achieving termination. A sample derivation is given in which an infinite model is constructed in the left-most branch even though the example is finitely satisfiable. The example can be transferred to $\mathcal{ALBO}^{\mathsf{id}}$ thus leaving open whether there are strategies for the $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) calculus to force left-most open branches to be always finitely bounded.

10. DISCUSSION

An important contribution of this paper is the unrestricted blocking mechanism based on the unrestricted blocking rule. It is conceptually similar but more generic than the approach of reusing domain terms in, for example, [Bry and Manthey 1988; Bry and Torge 1998], where it is used to compute domain minimal models. The following rule

$$\frac{a : \exists R.C}{a : \exists R.\{a_0\},\ \ a_0 : C \mid \cdots \mid a : \exists R.\{a_n\},\ \ a_n : C \mid a : \exists R.\{a'\},\ \ a' : C},$$

where $a_0, \ldots, a_n$ are all the individuals occurring in the current branch and a' is a fresh individual, is an adaptation of the δ^*-rule from Bry et al. [1988; 1998] to the language of description logics. Every tableau derivation with this rule can be rewritten to an equivalent tableau derivation using the unrestricted blocking rule but not the other way around.

An advantage of the unrestricted blocking mechanism is that it allows to separate proofs of termination of tableau calculi from proofs of their soundness and completeness. In the description, modal and hybrid logic literature blocking mechanisms are usually tightly integrated with the tableau rules and the tableau procedure. Soundness and completeness of the procedure is proved taking into account the given blocking mechanism. This creates intricacies and conceptual dependencies in the proofs. In contrast, blocking based on the (ub) rule is not tied to a specific logic. It can be added to any sound and complete tableau calculus without losing soundness and completeness.

Exploiting this we have shown that *any* tableau procedure based on a tableau calculus extended with this blocking mechanism is guaranteed to terminate if some additional general conditions are satisfied [Schmidt and Tishkovsky 2008]. These conditions are the blocking conditions (c1) and (c2) and fairness of a particular tableau strategy used in the tableau procedure, which is essential for the method to work. In addition, the effective finite model property must hold and it must be proved by a filtration argument correlating in an appropriate way with the original tableau calculus. The present paper provides a different kind of proof for termination to that in Schmidt and Tishkovsky [2008]. The proof here can be extended to show termination of tableau calculi equipped with the unrestricted blocking mechanism for logics for which the effective model property may not hold but the *finite model property* still holds. Proving the finite model property can be involved but proofs of it may already be known. This means that known finite model property results can be readily exploited to define terminating tableau calculi. We have introduced methods for systematically developing semantic tableau calculi for modal and related logics [Schmidt 2009; Schmidt and Tishkovsky 2011]. Adding unrestricted blocking turns many of the generated tableau calculi into terminating ones.

The way the blocking mechanism is defined in our calculus means that from some point onwards it needs to be used eagerly, as is required by Condition (c2). This restriction leaves room for different tableau expansion strategies to be used. For example, a tableau procedure can postpone applications of the (ub) rule or apply the rule in a non-eager way while the provisional model constructed in a branch is not too large. Subtle measures and strategies can be devised (for instance, based on run time or available memory) for determining when to start applying the blocking rule eagerly.

Eager application of the blocking rule produces small models. This is beneficial because small counter-examples can be produced whereas models generated by standard blocking mechanisms are generally larger. With smaller models memory consumption can be significantly smaller leading to quicker answers and improved performance.

On the other hand the blocking rule is a cut rule. Though analytic, excessive use of cuts in tableau derivations can degrade performance. However, the presence of cut rules does not necessarily lead to loss in performance. Cut rules generally add power to a calculus and can make proofs shorter. If cut rules are applied in controlled ways then decision procedures can also gain in performance in comparison to procedures without cut rules, as is demonstrated in the area of DPLL-based SAT-solving. It can thus be advantageous to control the application of the blocking rule via appropriate heuristics.

Another possibility is to limit the application of blocking. Existing standard blocking mechanisms only identify individuals if certain blocking conditions are true. In particular, they are based on inclusion or equality tests involving sets of expressions that must normally be constructed in a particular way. Individuals are identified only implicitly through the use of status variables that identify individuals as blocked and blockable. The expansion is assumed to be depth-first and the rules are assumed to be applied in a certain order. The unrestricted blocking mechanism is less restrictive and is monotone: there is no blocking test, any individual is blockable and blocked individuals remain blocked in a branch. There are conditions for the application of the unrestricted blocking rule but they are designed to be as weak and permissive as possible. As is shown in this paper none of the conditions can be omitted.

Whether appropriate blocking tests can be developed to obtain a depth-first decision procedure for $\mathcal{ALBO}^{\mathsf{id}}$ remains open. The main challenge in devising either a dynamic or static blocking test is the possible global effect of derivation steps on the provisional model contained in a branch. This effect is partly demonstrated by the example in Figure 4. Analysis of the proof of the effective finite model property for Boolean modal logic in Gargov and Passy [1990] suggests that, when identifying two elements of a

model, relations between each of these two elements and all other elements of the model must be taken into account. If this is not the case the filtration method used by Gargov and Passy may produce models with deficiencies, called "conflicts". The global effect in the example in Figure 4 caused by the introduction of a new individual is essentially of the same nature: it provides evidence that blocking conditions must take into consideration relations of each particular individual with all other individuals in the branch. The definition of a "star" in the proof of the effective finite model property for the two-variable fragment of first-order logic in Mortimer [1975] also gives an indication of how complex such conditions may have to be for $\mathcal{ALBO}^{\mathsf{id}}$. An algorithm to check such blocking conditions is not easy to define and it is far from clear whether a depth-first procedure with such a blocking test would necessarily be more efficient than a depth-first iterative deepening implementation of the approach presented in this paper.

Though standard blocking mechanisms are defined significantly differently, they can be viewed as specialisations and adaptations of unrestricted blocking. Standard blocking amounts to the restricted application of blocking, sometimes without change of the status of blocked individuals (static blocking), sometimes with change in the status of blocked individuals (dynamic blocking). Restricting applications of blocking reduces the number of branching nodes and the overall search space can be significantly smaller. A particular blocking test may need to be realised by an algorithm complicated enough to create considerable overhead. Ultimately, the trade-off between the resources required for blocking tests and excessive branching created as a result of applications of the blocking rule is inescapable and needs to be carefully considered. While investigations in this direction are ongoing, in this paper we presented the unrestricted blocking mechanism as a generic technique to provide termination for any logic with the finite model property.

Different standard blocking mechanisms are needed for different logics. While one blocking mechanism might work for a particular logic it can be incomplete for a more expressive logic. The advantage of unrestricted blocking is automatic soundness and completeness for any logic. Analysis of the termination proof in Schmidt and Tishkovsky [2008] however suggests that, in the case of a particular logic, an appropriate blocking test can be extracted from the definition of a filtrated model used in the proof of the effective finite model property for the logic. This opens a promising direction for further research into devising specialised blocking tests and blocking mechanisms in a systematic way.

Compared to the tableau calculi of the conference paper [Schmidt and Tishkovsky 2007] there are a few differences in the calculus $T_{\mathcal{ALBO}^{\mathsf{id}}}$. Besides the presence of rules for the identity role in $T_{\mathcal{ALBO}^{\mathsf{id}}}$, the essential difference is that the calculi in the conference version include this additional congruence rule:

$$\text{(bridge)}: \frac{a' : \{a''\}, \quad a : \exists R.\{a'\}}{a : \exists R.\{a''\}}.$$

The results of the present paper show that this rule is superfluous for completeness of $T_{\mathcal{ALBO}^{\mathsf{id}}}$. (We remark the (bridge) rule is sound but not derivable in $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub).)

Our use of the equality constraints $a : \{a\}$ in the tableau rules is non-standard and employs an idea proposed and implemented in the METTEL tableau prover [Tishkovsky 2005; Hustadt et al. 2006; Tishkovsky et al. 2011]. Normally these labelled concepts would be deleted by standard simplification rules since $a : \{a\}$ is valid. This would not be correct in our calculus because we use these equality constraints as *domain predicates* for keeping track of the individuals *essential* for instantiation by the γ rules in the calculus, and ultimately, for keeping track of the individuals essential for the construction of the model in a tableau branch. Here, by γ rules we mean the

rules for universally quantified expressions creating, by repeated application, all instantiations of individuals occurring in conclusions but not in the premise (that is, the implicit universally quantified first-order variables). The γ rules of the $T_{\mathcal{ALBO}^{\mathsf{id}}}$(ub) calculus are the ($\neg\exists\neg$) rule and the (ub) rule. Naively these rules can be defined to use all individuals occurring in the input set and all individuals introduced by the ($\exists$) rule for instantiation but this would create unnecessarily many instances and would mean the search space becomes unnecessarily large. The ($\neg\exists$) rule does not require domain predication because this is already achieved by the second premise of the rule. Domain predication can be obtained in other ways by using an explicit concept name representing the domain as is done in Hustadt and Schmidt [1999], Schmidt [2006b], and Baumgartner and Schmidt [2006]. If preferred, domain predication can be omitted and captured by appropriate side-conditions, but then the (refl) rule and the γ rules need to be reformulated and the proofs adapted slightly.

Although the proofs in this paper are relatively concise, they are not trivial. They are based on the idea that the finite model property for $\mathcal{ALBO}^{\mathsf{id}}$ and termination property of the proposed tableau calculus are of the same nature, which is an important contribution. In particular, it was proved that a finite model (if it exists) can be generated by applying the unrestricted blocking rule eagerly within the branch.

While our tableau approach is in line with existing semantic tableau approaches described in the description, modal and hybrid logic literature, our presentation departs in significant ways from other presentations. To make the approach and the results as general as possible, we have presented it in terms of an abstract deduction calculus given by a set of inference rules without making any assumptions about how branch selection is performed, how tableau derivations are constructed, in which order the rules are applied, which individuals are blockable and the need for using status variables. Another difference is that we view a tableau to be a (tree) derivation rather than a model as is the tendency in the description logic literature. In our context models are in general not tree models or even tree-like, because $\mathcal{ALBO}^{\mathsf{id}}$ does not have the tree-model property. The unrestricted blocking rule can identify any pair of individuals meaning that any structural properties, like being a tree model, are easily lost. This shift in perspective is useful for developing tableau approaches for description logics without a form of tree model property.

Practical considerations such as branch selection order, rule application order, search strategies and heuristics are important considerations when embarking on an implementation. We touched on practical aspects and described general notions of fairness providing minimal conditions to guarantee sound, complete and terminating deterministic tableau procedures. 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.

The unrestricted blocking mechanism has been implemented in METTEL tableau prover [Tishkovsky 2005; Tishkovsky et al. 2011], its successor, the prover generator METTEL^2 [Tishkovsky et al. 2012], and the MSPASS first-order resolution theorem prover. Tests with various description logic problems and problems in the TPTP library show that the general termination mechanism by means of the unrestricted blocking rule does work in practice [Baumgartner and Schmidt 2008]. A more thorough discussion of these implementations and other practical aspects, let alone performance results, is beyond the scope of the paper.

11. 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 $\mathcal{ALB}$, $\mathcal{ALBO}$ or $\mathcal{ALBO}^{\mathsf{id}}$. The present paper closes this gap. The introduced blocking mechanism opens the way for improving the OWL 2 standard to cover larger decidable fragments of first-order logic and larger fragments of natural language than it currently does, with obvious benefits for applications of the semantic web.

ACKNOWLEDGMENTS

We thank Hilverd Reker for useful discussions. The work was supported by research grants EP/D056152/1, EP/F068530/1 and EP/H043748/1 of the UK EPSRC.

REFERENCES

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2003. *Description Logic Handbook*. Cambridge University Press.

Franz Baader and Ulrike Sattler. 2001. An Overview of Tableau Algorithms for Description Logics. *Studia Logica* 69, 1 (2001), 5–40. http://dx.doi.org/10.1023/A:1013882326814

Peter Baumgartner and Renate A. Schmidt. 2006. Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06) (Lecture Notes in Artificial Intelligence)*, Ulrich Furbach and Natarajan Shankar (Eds.), Vol. 4130. Springer, 125–139.

Peter Baumgartner and Renate A. Schmidt. 2008. Blocking and Other Enhancements for Bottom-Up Model Generation Methods. (2008). Extended version of [Baumgartner and Schmidt 2006]. Manuscript.

Patrick Blackburn. 2000. Internalizing Labelled Deduction. *Journal of Logic and Computation* 10, 1 (2000), 137–168.

François Bry and Rainer Manthey. 1988. Proving Finite Satisfiability of Deductive Databases. In *Proceedings of the 1st Workshop on Computer Science Logic (CSL'87) (Lecture Notes in Computer Science)*, Egon Börger, Hans Kleine Büning, and Michael M. Richter (Eds.), Vol. 329. Springer, 44–55.

François Bry and Sunna Torge. 1998. A Deduction Method Complete for Refutation and Finite Satisfiability. In *Proceedings of the 6th European Conference on Logics in Artificial Intelligence (JELIA'98) (Lecture Notes in Computer Science)*, Jürgen Dix, Luis Fariñas del Cerro, and Ulrich Furbach (Eds.), Vol. 1489. Springer, 1–17.

Hans De Nivelle and Ian Pratt-Hartmann. 2001. A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality. In *Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR'01) (Lecture Notes in Artificial Intelligence)*, Rajeev Goré, Alexander Leitsch, and Tobias Nipkow (Eds.), Vol. 2083. Springer, 211–225.

Hans De Nivelle, Renate A. Schmidt, and Ullrich Hustadt. 2000. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL* 8, 3 (2000), 265–292.

Luis Fariñas del Cerro and Olivier Gasquet. 2002. A General Framework for Pattern-Driven Modal Tableaux. *Logic Journal of the IGPL* 1, 1 (2002), 51–83.

Melvin Fitting. 1972. Tableau Methods of Proof for Modal Logics. *Notre Dame Journal of Formal Logic* 13, 2 (1972), 237–247.

George Gargov and Solomon Passy. 1990. A Note on Boolean Modal Logic. In *Mathematical Logic: Proceedings of the 1988 Heyting summerschool*, Petio P. Petkov (Ed.). Plenum Press, 299–309.

George Gargov, Solomon Passy, and Tinko Tinchev. 1987. Modal Environment for Boolean Speculations. In *Mathematical Logic and its Applications: Proceedings of the 1986 Gödel Conference*, Dimiter Skordev (Ed.). Plenum Press, 253–263.

Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. 1997. On the Decision Problem for Two-Variable First-Order Logic. *Bulletin of the Section of Logic* 3 (1997), 53–69.

Ullrich Hustadt and Renate A. Schmidt. 1999. On the Relation of Resolution and Tableaux Proof Systems for Description Logics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, Thomas Dean (Ed.). Morgan Kaufmann, 110–115.

Ullrich Hustadt and Renate A. Schmidt. 2000. Issues of Decidability for Description Logics in the Framework of Resolution. In *Automated Deduction in Classical and Non-Classical Logics (Lecture Notes in Artificial Intelligence)*, Ricardo Caferra and Gernot Salzer (Eds.), Vol. 1761. Springer, 191–205.

Ullrich Hustadt, Renate A. Schmidt, and Christoph Weidenbach. 1999. MSPASS: Subsumption Testing with SPASS. In *Proceedings of the 1999 International Workshop on Description Logics (DL'99)*, Patrick Lambrix, Alexander Borgida, Maurizio Lenzerini, Ralf Möller, and Peter F. Patel-Schneider (Eds.). Linköping University, 136–137.

Ullrich Hustadt, Dmitry Tishkovsky, Frank Wolter, and Michael Zakharyaschev. 2006. Automated reasoning about metric and topology (System description). In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA'06) (Lecture Notes in Artificial Intelligence)*, Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa (Eds.), Vol. 4160. Springer, 490–493.

Carsten Lutz and Ulrike Sattler. 2002. The Complexity of Reasoning with Boolean Modal Logics. In *Advances in Modal Logics, Vol. 3*, Frank Wolter, Heinrich Wansing, Maarten de Rijke, and Michael Zakharyaschev (Eds.). CSLI Publications.

Carsten Lutz and Dirk Walther. 2004. PDL with Negation of Atomic Programs. In *Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR'04) (Lecture Notes in Computer Science)*, David A. Basin and Michaël Rusinowitch (Eds.), Vol. 3097. Springer, 259–273.

Fabio Massacci. 2001. Decision Procedures for Expressive Description Logics with Intersection, Composition, Converse of Roles and Role Identity.. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Bernhard Nebel (Ed.). Morgan Kaufmann, 193–198.

Michael Mortimer. 1975. On languages with two variables. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 21 (1975), 135–140.

Boris Motik, Rob Shearer, and Ian Horrocks. 2009. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research (JAIR)* 36 (2009), 165–228.

OWL. 2004. Web Ontology Language (OWL). http://www.w3.org/2004/OWL. (2004).

OWL 2. 2009. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. http://www.w3.org/TR/owl2-syntax/. (2009).

Hilverd Reker. 2011. *Tableau-Based Reasoning for Decidable Fragments of First-Order Logic*. Ph.D. Dissertation. School of Computer Science, The University of Manchester, UK.

Alexandre Riazanov and Andrei Voronkov. 1999. Vampire. In *Proceedings of the 16th Conference on Automated Deduction (CADE-16) (Lecture Notes in Artificial Intelligence)*, Harald Ganzinger (Ed.), Vol. 1632. Springer, 292–296.

Renate A. Schmidt. 2006a. Developing Modal Tableaux and Resolution Methods via First-Order Resolution. In *Advances in Modal Logic, Vol. 6*, Guido Governatori, Ian M. Hodkinson, and Yde Venema (Eds.). College Publications, 1–26.

Renate A. Schmidt. 2006b. Solvability with Resolution of Problems in the Bernays-Schönfinkel Class. In *Deduction and Applications (Dagstuhl Seminar Proceedings)*, Franz Baader, Peter Baumgartner, Roberto Nieuwenhuis, and Andrei Voronkov (Eds.). IBFI, Schloss Dagstuhl, Germany, 16. Abstract of Dagstuhl presentation of joint results with U. Hustadt (Oct. 2005).

Renate A. Schmidt. 2009. A New Methodology for Developing Deduction Methods. *Annals of Mathematics and Artificial Intelligence* 55, 1–2 (2009), 155–187.

Renate A. Schmidt, Ewa Orlowska, and Ullrich Hustadt. 2004. Two Proof Systems for Peirce Algebras. In *Relational and Kleene-Algebraic Methods in Computer Science: 7th International Seminar on Relational Methods in Computer Science and 2nd International Workshop on Applications of Kleene Algebra (RelMiCS'03) (Lecture Notes in Computer Science)*, Rudolf Berghammer, Bernhard Möller, and Georg Struth (Eds.), Vol. 3051. Springer, 238–251.

Renate A. Schmidt and Dmitry Tishkovsky. 2007. Using Tableau to Decide Expressive Description Logics with Role Negation. In *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC'07) (Lecture Notes in Computer Science)*, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.), Vol. 4825. Springer, 438–451.

Renate A. Schmidt and Dmitry Tishkovsky. 2008. A General Tableau Method for Deciding Description Logics, Modal Logics and Related First-Order Fragments. In *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08) (Lecture Notes in Computer Science)*, Alessandro Armando, Peter Baumgartner, and Gilles Dowek (Eds.), Vol. 5195. Springer, 194–209.

Renate A. Schmidt and Dmitry Tishkovsky. 2011. Automated Synthesis of Tableau Calculi. *Logical Methods in Computer Science* 7, 2:6 (2011), 1–32.

Stephan Schulz. 2002. E: A Brainiac Theorem Prover. *AI Communications* 15, 2–3 (2002), 111–126.

Dmitry Tishkovsky. 2005. The MetTeL prover. http://www.mettel-prover.org/. (2005).

Dmitry Tishkovsky, Renate A. Schmidt, and Mohammad Khodadadi. 2011. MetTeL: A Tableau Prover with Logic-Independent Inference Engine. In *Proceedings of the 20th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'11) (Lecture Notes in Computer Science)*, Kai Brünnler and George Metcalfe (Eds.), Vol. 6793. Springer, 242–247.

Dmitry Tishkovsky, Renate A. Schmidt, and Mohammad Khodadadi. 2012. The Tableau Prover Generator MetTeI^2. In *Proceedings of the 13th European Conference on Logics in Artificial Intelligence (JELIA'12) (Lecture Notes in Artificial Intelligence)*, Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin (Eds.), Vol. 7519. Springer, 492–495.

Stephan Tobies. 2001. *Decidability Results and Practical Algorithms for Logics in Knowledge Representation*. Ph.D. Dissertation. RWTH Aachen, Germany.

Christoph Weidenbach, Renate A. Schmidt, Thomas Hillenbrand, Rostislav Rusev, and Dalibor Topic. 2007. System Description: SPASS Version 3.0. In *Proceedings of the 21st Conference on Automated Deduction (CADE-21) (Lecture Notes in Artificial Intelligence)*, Frank Pfenning (Ed.), Vol. 4603. Springer, 514–520.