Modal Tableau Systems with Blocking and Congruence Closure

Renate A. Schmidt[1]* and Uwe Waldmann[2]

[1] School of Computer Science, The University of Manchester, UK
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. Our interest in this paper are semantic tableau approaches closely related to bottom-up model generation methods. Using equality-based blocking techniques these can be used to decide logics representable in first-order logic that have the finite model property. Many common modal and description logics have these properties and can therefore be decided in this way. This paper integrates congruence closure, which is probably the most powerful and efficient way to realise reasoning with ground equations, into a modal tableau system with equality-based blocking. The system is described for an extension of modal logic **K** characterised by frames in which the accessibility relation is transitive and every world has a distinct immediate predecessor. We show the system is sound and complete, and discuss how various forms of blocking such as ancestor blocking can be realised in this setting. Though the investigation is focussed on a particular modal logic, the modal logic was chosen to show the most salient ideas and techniques for the results to be generalised to other tableau calculi and other logics.

1 Introduction

Tableau systems provide a natural and powerful form of reasoning widely used for non-classical logics, especially modal, description, and hybrid logics. In this paper the focus is on semantic tableau systems closely related to bottom-up model generation methods [4]. Using unrestricted blocking [20], which is an equality-based blocking technique, these can decide logics with the finite model property, representable in first-order logic [21, 22]. Many common modal and description logics have these properties and can therefore be decided using semantic tableau systems with equality-based blocking.

For many common modal and description logics there are ways to avoid the explicit use of equality in the tableau system [10, 2]. For more expressive logics, with nominals as in hybrid modal logics and description logics (nominals are distinguished propositional variables that hold at exactly one world), it becomes harder to avoid the explicit handling of equality (though not impossible [11]). For

* I am indebted to Christoph Weidenbach and Uwe Waldmann for hosting me at the Max-Planck-Institut für Informatik, Saarbrücken, during 2013–2014. Partial support from UK EPSRC research grant EP/H043748/1 is also gratefully acknowledged.

modal logics where the binary relations satisfy frame conditions expressible as first-order formulae with equality, explicit handling of equations is the easiest and sometimes the only known way to perform equality reasoning. Single-valuedness of a relation is an example of a frame condition expressed using equality. Another example is the following

$$(1) \qquad \forall x \exists y \forall z \Big(R(y,x) \wedge x \not\approx y \wedge \big((R(y,z) \wedge R(z,x)) \rightarrow (z \approx x \vee z \approx y) \big) \Big),$$

where $\approx$ denotes equality. This formula states that in the relation R every world has a *distinct immediate predecessor*. Provision for explicit equality reasoning is also necessary for tableau systems with equality-based blocking.

In semantic tableau systems explicit equality handling has been realised in a variety of ways. Using standard equality rules is conceptually easiest and most general, and is often used [6, 8, 20]. This approach leads to a combinatorial explosion of derived formulae to ensure all elements in the same equivalence class have the same information content. Many of these formulae are unneeded and fewer formulae are derived when using paramodulation-style rules, where the central idea is replacement of equals by equals [5, 8]. Ordered rewriting presents a further refinement and is significantly more efficient because equations are oriented by an ordering and then used to simplify the formulae. Ordered rewriting is used, e.g., in a semantic tableau system of [16] for the description logic $\mathcal{SHOI}$. Different equality reasoning methods have also been integrated into non-ground tableau and related approaches, e.g. [5, 8, 9].

In this paper we require efficient handling of *ground equations*. For this purpose congruence closure algorithms provide probably the most efficient algorithms [18]. The Nelson-Oppen congruence closure method [17] has been incorporated with Smullyan-type tableau system for first-order logic by [13]. Congruence closure algorithms have also been very successfully combined with the DPLL approach and are standardly integrated in SMT-solvers as theory reasoners for the theory of equations with uninterpreted function symbols [19].

The motivation of the present work is to combine congruence closure with semantic tableau systems for modal, description, and hybrid logics. Since it presents a general framework in which many existing congruence closure algorithms can be described (and in order to achieve more generality), we combine the *abstract congruence closure system* of [3] with our semantic tableau system. Our ultimate goal is to provide a general framework with general soundness and completeness results for developing and studying equality reasoning and blocking in semantic tableau systems. The tableau system we consider has been obtained in the tableau synthesis framework of [21], but in this framework equality is accommodated by the standard equality rules. In this paper we show how these can be replaced by abstract congruence closure rules.

The most closely related work is the aforementioned [13], because the flavour of the tableau systems we are concerned with is similar to that of Smullyan-type tableau systems for first-order logic. The key difference is the way in which we use the congruence closure algorithm: In [13], the congruence closure component is essentially a black box that is queried to check entailed equalities. In contrast,

we use the convergent term rewrite system produced by the abstract congruence closure algorithm also systematically to normalise the remaining tableau formulae. This means that duplication of formulae is avoided and that restrictions of the search space that depend on normalisation can be applied easily. In addition, we show that the ideas are not limited to a fixed set of the well-known tableau rules for first-order logic, but can be combined with special-purpose tableau systems of other logics having other kinds of tableau rules. Also related is [16] and the implementation of equality reasoning in METTEL-generated tableau provers [25], where ordered rewriting is used. This work does however not have the same level of generality as abstract congruence closure, and no soundness and completeness proofs are given.

Another important difference to [13], and many modal, description, and hybrid logic tableau systems, is the use of Skolem terms to represent witnesses, instead of constants. Skolem terms have significant advantages, especially when blocking is used and/or explicit equality reasoning is needed [16]. They provide a convenient and general-purpose technical device to keep track of existential quantifier dependencies between witnesses. In conjunction with rewriting or congruence closure fewer inferences need to be performed since, when rewriting a term, all occurrences of the term, also in the dependency information, are rewritten. As an example consider the labelled formulae $s_1 : \neg\Box\phi$ and $s_2 : \neg\Box\phi$, from which we can derive $f_{\neg\Box\phi}(s_1) : \neg\phi$ and $f_{\neg\Box\phi}(s_2) : \neg\phi$, where $f_{\neg\Box\phi}$ is the Skolem function associated with the modal formula $\neg\Box\phi$. If we later obtain the equation $s_1 \approx s_2$, then the witnesses $f_{\neg\Box\phi}(s_1)$ and $f_{\neg\Box\phi}(s_2)$ also become semantically equal. If we turn the equation $s_1 \approx s_2$ into a rewrite rule $s_1 \to s_2$ and use it for destructive replacement of labels, then even the formulae $f_{\neg\Box\phi}(s_1) : \neg\phi$ and $f_{\neg\Box\phi}(s_2) : \neg\phi$ become identical, so that one copy is deleted and is no longer available for tableau expansions. Without Skolem terms other forms of bookkeeping are needed and may require reapplication of witness-creating rules which is not needed in our setting. In the tableau synthesis framework more generality is achieved because Skolem terms allow the encoding of arbitrary first-order properties as tableau rules [21], including properties such as (1), which otherwise presents difficulties.

These advantages of Skolem terms carry over to tableau systems with congruence closure. Skolem terms however tend to clutter derivations when the nesting is deep, which is inconvenient when manually writing derivations. The present work does in a sense solve this problem, because in tableau systems with congruence closure the Skolem terms are abstracted away and hidden in the rewrite rules, thus resulting in more easily consumable presentations of the derivations. We also see how properties such as (1) can be encoded as tableau rules by using constants and dispersing the Skolem terms into the equational component.

Though the investigation is focussed on a particular modal logic, the logic was chosen to show the most salient ideas and techniques for the results to be generalised to tableau systems for other logics, including those obtainable by tableau synthesis. A presentation in its full generality would have obscured the main ideas.

Basic tableau rules:

$$\text{(cl)}\ \frac{s:\phi,\ s:\neg\phi}{\bot} \qquad \text{(}\bot\text{)}\ \frac{s:\bot}{\bot} \qquad \text{(}\neg\neg\text{)}\ \frac{s:\neg\neg\phi}{s:\phi}$$

$$\text{(}\alpha\text{)}\ \frac{s:\neg(\phi_1 \vee \ldots \vee \phi_k)}{s:\sim\phi_1,\ \ldots,\ s:\sim\phi_k} \qquad \text{(}\beta\text{)}\ \frac{s:\phi \vee \Psi}{s:\phi \mid s:\Psi}$$

$$\text{(}\Box\text{)}\ \frac{s:\Box\phi,\ R(s,t)}{t:\phi} \qquad \text{(}\neg\Box\text{)}\ \frac{s:\neg\Box\phi}{R(s,f_{\neg\Box\phi}(s)),\ f_{\neg\Box\phi}(s):\sim\phi}$$

$$\text{(ub)}\ \frac{}{s \approx t \mid s \not\approx t}$$

Paramodulation equality rules:

$$\text{(rfl)}\ \frac{s \not\approx s}{\bot} \qquad \text{(sym)}\ \frac{s \approx t}{t \approx s} \qquad \text{(prm)}\ \frac{s \approx t,\ G[s]}{G[t]}$$

Theory tableau rules:

$$\text{(tr)}\ \frac{R(s,t),\ R(t,u)}{R(s,u)}$$

$$\text{(dp1)}\ \frac{}{R(g(s),s)} \qquad \text{(dp2)}\ \frac{s \approx g(s)}{\bot} \qquad \text{(dp3)}\ \frac{R(g(s),t),\ R(t,s)}{t \approx s \mid t \approx g(s)}$$

Fig. 1. Tableau calculus Tab(ub) for **K(tr,dp)**. Ψ denotes a disjunction (with at least one disjunct). $\sim$ denotes complementation, i.e., $\sim\psi = \phi$ if $\psi = \neg\phi$, and $\sim\psi = \neg\psi$, otherwise. G denotes any tableau formula. $G[s]$ means s occurs as a subterm in G, and $G[t]$ denotes the formula obtained by replacing one occurrence of s with t.

The paper is structured as follows. To illustrate the main ideas of combining the abstract congruence closure system of [3] with semantic tableau systems, we consider a semantic tableau system for an extension of basic modal logic **K** characterised by frames in which the accessibility relation is transitive and where the frame condition (1) above holds. The logic, called **K(tr,dp)**, and its tableau system are introduced in Section 2. In Section 3 we show how congruence closure can be integrated into this system. We show soundness and completeness of the system in Sections 4 and 5, and describe in Section 6 how various forms of blocking, including ancestor blocking, can be realised. All proofs are omitted but can be found in [23].

2 Modal Logic K(tr,dp) and a Tableau System for It

We give a semantic definition of modal logic **K(tr,dp)**. **K(tr,dp)** is the propositional normal modal logic characterised by the class of relational structures (frames) (W, R), where W is a non-empty set and R is a binary relation defined over W, which is transitive and satisfies (1) as an additional *frame condition*. W represents the set of possible worlds and R is the accessibility relation over which the semantics of the necessitation operator $\Box$ and the possibility operator $\Diamond$ are defined.

Let Tab(ub) be the tableau system given by the rules in Figure 1. The rules operate on formulae of the form $\bot$, $s : \phi$, $R(s,t)$, $s \approx t$, $s \not\approx t$, where ϕ is a modal formula, and s and t are the *labels* interpreted as worlds in Kripke models. We refer to these formulae as the *tableau formulae* in **K(tr,dp)**. The labels s and t are terms of a freely generated term algebra over a signature Σ of constants (denoted by $a, b, \ldots$) and unary function symbols f_ψ and g for modal formulae ψ. The Skolem functions f_ψ and g provide a technical device to uniquely name the witnesses created in the rule ($\neg\Box$) for the diamond formulae (the $\neg\Box\phi$-formulae) and the rule (dp1) for the distinct predecessor property (1).

The (ub) rule is the unrestricted blocking rule, which will ensure the tableau system terminates for all finitely satisfiable formulae and constructs a finite model. More restricted forms of blocking are described in Section 6.

The frame conditions in the definition of **K(tr,dp)** were chosen so that the incorporation of congruence closure (in the next section) into the corresponding tableau rules exhibits as many different interesting aspects as possible. Transitivity is a common frame condition and the transitivity rule (tr) a well-known rule. Frame condition (1) is used as an example in [24] to illustrate tableau rule refinement techniques. The corresponding rules (dp1), (dp2) and (dp3) are instructive because they contain an equality predicate and Skolem terms in premise positions, which are important, more difficult cases for the combination with congruence closure.

Tableau systems are best suited for applications where models need to be found for satisfiable formulae. Given a formula ϕ, a semantic tableau system attempts to construct a model that realises the formula. The start state of the derivation is then the set $N_0 = \{a : \phi\}$, where a denotes a fresh constant in Σ; it represents the initial world of the model to be constructed (if this is possible). If in every branch of the derivation $\bot$ was derived then no model can exist and ϕ is unsatisfiable. Else, there will be a (possibly infinite) branch from which a model can be read off in the limit. E.g., for the formula $\Diamond\top \wedge \Box\Diamond p$ the following model may be constructed (there are others).

$$R(g(a), g(a)), \quad R(g(a), a), \quad R(a, g(a)), \quad R(a, a), \quad a : p$$

Without the unrestricted blocking rule (ub) an infinite model is constructed.

We say a tableau calculus is *sound* when for a satisfiable set of tableau formulae any fully expanded tableau derivation has an open branch. A tableau derivation is *fully expanded* if all branches are either closed, or open and fully expanded. A tableau calculus is *refutationally complete* if for any unsatisfiable set of tableau formulae there is a closed tableau derivation. A tableau calculus is *constructively complete*, if for every open fully expanded branch a model exists, that can be read off from the branch.

By Tab we denote the calculus without the unrestricted blocking rule. The rules in Tab and Tab(ub) are the ones obtained by tableau synthesis and rule refinement [21, 24] from the semantic definition of **K(tr,dp)**, except we use paramodulation-style rules instead of the standard equality rules. With the appropriate adaptations of the proofs in [21, 24] for this, it follows that:

Theorem 1. *The tableau calculi Tab and Tab(ub) are sound and constructively complete for testing satisfiability of formulae (or sets of tableau formulae) in* ***K(tr,dp)****. They are also refutationally complete.*

3 Modal Tableau System with Congruence Closure

Congruence closure algorithms provide an efficient way to perform reasoning with ground equations and can be combined with DPLL algorithms, but also with tableau systems as we show in this section. A congruence closure algorithm transforms an arbitrary set of ground equations into an equivalent confluent and terminating ground rewrite system.[3] Checking whether two terms are semantically equivalent with respect to the original set of equations amounts to checking whether the normal forms of the two terms with respect to the rewrite system coincide. For efficiency reasons, it is useful to construct the rewrite system over a signature extended by a set of new constants symbols and to restrict to a specific form of *flat* rewrite rules.

Let K be a set of constant symbols (denoted by $c, d, \dots$) disjoint from Σ. A *D-rule* with respect to Σ and K is a rewrite rule of the form $h(c_1, \dots, c_k) \to c$, where $h \in \Sigma$, $k \geq 0$, and $c_i, c \in K$. A *C-rule* is a rewrite rule of the form $c \to c'$, where $c, c' \in K$.

In order to guarantee termination of the set of rewrite rules, we assume that $\succ$ is an arbitrary total and well-founded ordering on $\Sigma \cup K$ with the property that $f \succ c$ for every $f \in \Sigma$ and $c \in K$. We can extend $\succ$ to an ordering $\succ_T$ on arbitrary terms by defining $\succ_T$ as the Knuth-Bendix ordering with precedence $\succ$ and weight 1 for every function or constant symbol. The ordering $\succ_T$ is total and well-founded on ground terms over $\Sigma \cup K$ (even if Σ and/or K are infinite); moreover $c \succ c'$ implies $c \succ_T c'$ for $c, c' \in K$, and $t \succ_T c$ whenever $c \in K$ and t contains a symbol from Σ. These properties ensure that $t \succ_T t'$ holds for all generated rules $t \to t'$, and hence, that the set of rules terminates.

The inference rules in Figures 2 and 3 combine the tableau rules of the previous section with the abstract congruence closure rules of [3]. The integration is defined to be as modular as possible, to limit any problematic interactions and present a clean separation between the modal tableau formulae and the congruence closure rules. Let the calculus be named Tab(ub,cc).

A *tableau state* is a pair $N \,[\!]\, E$ of a set N of tableau formulae and a set E of D- and C-rules. E denotes the rewrite system being built. The inference rules have the general form:

$$(\rho) \qquad \frac{N \,[\!]\, E}{N_1 \,[\!]\, E_1 \mid \dots \mid N_k \,[\!]\, E_k}$$

with $k \geq 1$. In general, the inference process constructs a *derivation tree* in which the nodes are tableau states. A *branch* $\mathcal{B}$ in a tableau derivation is a sequence of pairs $N_0 \,[\!]\, E_0,\ N_1 \,[\!]\, E_1,\ \dots,\ N_i \,[\!]\, E_i,\ \dots$, where $N_0 \,[\!]\, E_0$ is the *start state*, and

[3] We refer to [1] for standard notions and notations in term rewriting.

Basic tableau rules:

$$\text{(cl)}\ \frac{N,\ c:\phi,\ c:\neg\phi\ [\!]\ E}{\bot\ [\!]\ E} \qquad (\bot)\ \frac{N,\ c:\bot\ [\!]\ E}{\bot\ [\!]\ E} \qquad (\neg\neg)\ \frac{N,\ c:\neg\neg\phi\ [\!]\ E}{N,\ c:\phi\ [\!]\ E}$$

$$(\alpha)\ \frac{N,\ c:\neg(\phi_1\vee\ldots\vee\phi_k)\ [\!]\ E}{N,\ c:\sim\phi_1,\ \ldots,\ c:\sim\phi_k\ [\!]\ E} \qquad (\beta)\ \frac{N,\ c:\phi\vee\Psi\ [\!]\ E}{N,\ c:\phi\ [\!]\ E\mid N,\ c:\Psi\ [\!]\ E}$$

$$(\Box)\ \frac{N,\ c:\Box\phi,\ R(c,d)\ [\!]\ E}{N,\ c:\Box\phi,\ R(c,d),\ d:\phi\ [\!]\ E}$$

$$(\neg\Box)\ \frac{N,\ c:\neg\Box\phi\ [\!]\ E}{N,\ R(c,d),\ d:\sim\phi\ [\!]\ E,\ f_{\neg\Box\phi}(c)\to d}$$ provided c is in E-normal form; d is a new constant

$$(\text{ub})\ \frac{N\ [\!]\ E}{N,\ c\approx d\ [\!]\ E\mid N,\ c\not\approx d\ [\!]\ E}$$ provided c and d are distinct constants in E-normal form

Theory tableau rules:

$$(\text{tr})\ \frac{N,\ R(c,d),\ R(d,d')\ [\!]\ E}{N,\ R(c,d),\ R(d,d'),\ R(c,d')\ [\!]\ E}$$

$$(\text{dp1})\ \frac{N\ [\!]\ E}{N,\ R(d,c)\ [\!]\ E,\ g(c)\to d}$$ provided c is in E-normal form; d is a new constant

$$(\text{dp2})\ \frac{N\ [\!]\ E,\ g(c)\to d}{\bot\ [\!]\ E,\ g(c)\to d}$$ provided c is the E-normal form of d

$$(\text{dp3})\ \frac{N,\ R(d,c'),\ R(c',c)\ [\!]\ E,\ g(c)\to d'}{\begin{array}{l}N,\ R(d,c'),\ R(c',c),\ c'\approx c\ [\!]\ E,\ g(c)\to d'\\ \quad\mid N,\ R(d,c'),\ R(c',c),\ c'\approx d\ [\!]\ E,\ g(c)\to d'\end{array}}$$ provided d is the E-normal form of d'

Fig. 2. Adapted tableau rules incorporating congruence closure.

each subsequent state $N_i\,[\!]\,E_i$ is obtained from $N_{i-1}\,[\!]\,E_{i-1}$ by the application of an inference rule. A branch is regarded as *closed*, as soon as $\bot$ is derived in an N_i. A branch is *open* when it is not closed.

The start state $N_0\,[\!]\,E_0$ is obtained by a preprocessing stage from the given set N of tableau formulae involving the exhaustive application of this rule

$$\text{Extension:}\quad \frac{N[t]\ [\!]\ E}{N[c]\ [\!]\ E,\ t\to c}$$ provided c is new and $t\to c$ is a D-rule

and the Simplification rule in Figure 3. Thus, N_0 is the flattened version of N and E_0 is the set of D-rules and C-rules defining all subterms occurring in N. If the Simplification rule is given precedence over the Extension rule there will be maximal sharing.

The inference rules in Figure 2 are adaptations of the basic tableau rules and the theory rules in Figure 1 to tableau formulae in normalised form. The rules manipulate the tableau formulae as before, one important difference though is the way witnesses are created. In the $(\neg\Box)$-rule the Skolem term $f_{\neg\Box\phi}(c)$ is created and added to the rewrite system in the new D-rule $f_{\neg\Box\phi}(c)\to d$, which defines it by a new constant d from K. On the left-hand side d represents the

Equality theory propagation rule:

$$\text{(id)}\ \frac{N,\ s \not\approx s \ [\!]\ E}{\bot \ [\!]\ E}$$

Congruence closure tableau rules:

Simplification: $\dfrac{N[t] \ [\!]\ E,\ t \to c}{N[c] \ [\!]\ E,\ t \to c}$

Orientation: $\dfrac{N,\ t \approx c \ [\!]\ E}{N \ [\!]\ E,\ t \to c}$ provided $t \succ_T c$

Deletion: $\dfrac{N,\ t \approx t \ [\!]\ E}{N \ [\!]\ E}$

Deduction: $\dfrac{N \ [\!]\ E,\ t \to c,\ t \to d}{N,\ c \approx d \ [\!]\ E,\ t \to d}$ provided $c \succ_T d$

Collapse: $\dfrac{N \ [\!]\ E,\ s[c] \to c',\ c \to d}{N \ [\!]\ E,\ s[d] \to c',\ c \to d}$ provided c is a proper subterm of s

Fig. 3. Congruence closure rules for equality reasoning.

newly created successor in the derived tableau formulae $R(c, d)$ and $d : \sim\phi$. The other basic tableau rules and the transitivity rule do not affect the rewrite system E, and are obvious adaptations from the rules in the previous system.

The (dp1)-rule is the other witness creating rule in the calculus and is adapted in the same way as the $(\neg\Box)$-rule. That is, a new D-rule is added that defines the new Skolem term $g(c)$ and its representative d in K. The rules (dp2) and (dp3) have Skolem terms in premise position. Because in the adapted tableau system Skolem terms can occur only in D-rules in the rewrite system the adaptations to normalised form involve look-ups in the rewrite system, see the third and fourth rules in Figure 2.

The paramodulation rules in Tab(ub) are replaced by the congruence closure rules listed in Figure 3. Their purpose is to build a rewrite system, normalise the tableau formulae via Simplification and Deletion, propagate derived equations via Deduction, and perform theory propagation steps. The only theory propagation rule is the (id) rule.

The congruence closure rules are based on the abstract congruence closure framework of [3]. We have added the requirement that $c \succ_T d$ to the Deduction rule in order to ensure that $t \to c$ is eliminated by the rule, and not $t \to d$ which is the smaller of the two. The Extension rule is not included since exhaustive extension and simplification is performed at the outset. This means only constants occur in N_0 of the start state and the rules are defined in such a way that no non-constant terms are introduced to the tableau formula part during the derivation. We note that if the optional Composition rule

Composition: $\dfrac{N \ [\!]\ E,\ t \to c,\ c \to d}{N \ [\!]\ E,\ t \to d,\ c \to d}$

is made a mandatory rule, then the side-conditions of rule (dp2) and rule (dp3) can be simplified respectively to $c = d$ and $d = d'$, because then, in general, both sides of the rewrite rules are maximally reduced.

We assume fairness for the construction of a derivation. This is important if branches can be infinite. The construction is *fair* if, when an inference is possible forever, then it is performed eventually.

Theorem 2. *The calculus Tab(ub,cc) is sound and constructively complete for testing satisfiability of sets of tableau formulae in* ***K(tr,dp)****. It is also refutationally complete.*

Formal proofs are given in the next two sections.

4 Semantics and Soundness

We define the semantics of formulae in Tab(ub,cc)-rules by an *interpretation* $\mathcal{I} = (U.^{\mathcal{I}})$, where U is a non-empty set and $\cdot^{\mathcal{I}}$ is the interpretation function mapping terms (labels) to elements in U, propositional variables to subsets of U, $\approx$ to the identity relation over U, and R to a relation over U that is transitive and satisfies property (1). The meaning of modal formulae in $\mathcal{I}$ is defined with respect to the structure $\mathcal{M} = (U, R^{\mathcal{I}}, v)$, where v is the restriction of $\cdot^{\mathcal{I}}$ to propositional variables. v defines the valuation of propositional variables and $\mathcal{M}$ is a Kripke structure. Satisfiability of modal formulae in $\mathcal{M}$ is now defined as usual by:

$$\mathcal{M}, x \models p \text{ iff } x \in v(p) \qquad \mathcal{M}, x \not\models \bot \qquad \mathcal{M}, x \models \neg\phi \text{ iff } \mathcal{M}, x \not\models \phi$$
$$\mathcal{M}, x \models \phi_1 \vee \ldots \vee \phi_k \text{ iff } \mathcal{M}, x \models \phi_i \text{ for some } i,\ 1 \leq i \leq k$$
$$\mathcal{M}, x \models \Box\phi \text{ iff for all } y,\ (x, y) \in R^{\mathcal{I}} \text{ implies } \mathcal{M}, y \models \phi$$

Satisfiability in $\mathcal{I}$ of tableau formulae and rewrite rules is defined by:

$$\mathcal{I} \models s : \phi \text{ iff } \mathcal{M}, s^{\mathcal{I}} \models \phi \qquad \mathcal{I} \models R(s, t) \text{ iff } (s^{\mathcal{I}}, t^{\mathcal{I}}) \in R^{\mathcal{I}}$$
$$\mathcal{I} \models s \approx t \text{ iff } s^{\mathcal{I}} = t^{\mathcal{I}} \qquad \mathcal{I} \models s \to t \text{ iff } s^{\mathcal{I}} = t^{\mathcal{I}} \qquad \mathcal{I} \models s \not\approx t \text{ iff } s^{\mathcal{I}} \neq t^{\mathcal{I}}$$

It is not difficult to show that each of Tab(ub,cc)-rule is sound, i.e., when each of the formulae in the premise $N \,[]\, E$ of a rule (ρ) is true in an interpretation $\mathcal{I}$ then all of the formulae in one of the conclusions $N_i \,[]\, E_i$ are true in $\mathcal{I}$.

It immediately follows that Tab(ub,cc) is sound, i.e., for any set N of tableau formulae for **K(tr,dp)**, there is an open, fully expanded branch in some derivation constructed using Tab(ub,cc). In fact, an open, fully expanded branch is found in any Tab(ub,cc)-derivation, since the calculus is proof confluent.

5 Completeness

In this section we prove that the calculus Tab(ub,cc) is constructively complete.

We need a condition that ensures that the tableau rules (and in particular the theory rules) do not interfere with the congruence closure rules. Accordingly we call a tableau rule

$$\frac{N \,[\!]\, E}{N_1 \,[\!]\, E_1 \mid \ldots \mid N_k \,[\!]\, E_k}$$

admissible, if for every $i \leq k$, either $N_i = \bot$, or the following conditions all hold:

(i) $E \subseteq E_i$,
(ii) $\{s \approx t \mid s \approx t \in N\} \subseteq N_i$,
(iii) $E_i \setminus E$ consists only of D-rules, and
(iv) all terms that occur in $N_i \setminus N$ are constants in K.

This means that admissible tableau rules retain all positive equational formulae, the only rules introduced during an inference step are D-rules and only constants from K are introduced. It is easy to check that the basic tableau rules and the theory rules are admissible.

For an open branch $\mathcal{B} = N_0 \,[\!]\, E_0,\ N_1 \,[\!]\, E_1,\ \ldots,\ N_i \,[\!]\, E_i,\ \ldots$ we define the set of all rules and equations on the branch by $E_\infty = \bigcup_{i\geq 0} E_i \cup \{s \approx t \mid s \approx t \in N_i\}$ and the set of persistent rules and equations on the branch by $E_* = \bigcup_{i\geq 0} \bigcap_{j\geq i} (E_j \cup \{s \approx t \mid s \approx t \in N_j\})$. (If $\mathcal{B}$ is finite, then E_* equals $E_i \cup \{s \approx t \mid s \approx t \in N_i\}$, where $N_i \,[\!]\, E_i$ is the last node of $\mathcal{B}$.)

To discuss the properties of E_∞ and E_*, we have to extend the ordering $\succ_T$ to an ordering on equations and rewrite rules. We define the ordering $\succ_E$ on equations and rewrite rules by mapping every equation $s \approx t$ to the multiset $\{s, s, t, t\}$, every rewrite rule $s \to t$ to the multiset $\{s, t\}$, and by comparing the resulting multisets using the multiset extension of $\succ_T$.

If E is a set of equations and rewrite rules and s and t are terms over $\Sigma \cup K$, we write $s \sim_E t$ if the equation $s \approx t$ is logically entailed by the equations and rewrite rules in E (where we do not distinguish between equations and rewrite rules). If E is a confluent and terminating set of rewrite rules, we write $s{\downarrow_E}$ for the E-normal form of s. Similarly we use the notation $F{\downarrow_E}$ and $N{\downarrow_E}$ for the normalisation of a formula F or of a set N of formulae with respect to E.

Lemma 1. *Let all basic and theory tableau rules be admissible. Let $\mathcal{B}$ be an open branch that is fully expanded with respect to the congruence closure rules. Then, E_∞ and E_* have the following properties:*

(i) All equations in E_∞ have the form $c \approx d$ with $c, d \in K$, and all rewrite rules in E_∞ are C-rules or D-rules.
(ii) E_* does not contain any equations, that is, $E_* = \bigcup_{i\geq 0} \bigcap_{j\geq i} E_j$.
(iii) $\bigcup_{i\geq 0} E_i$ and E_* are terminating.
(iv) E_* is confluent.
(v) If a term u is reducible by a rewrite rule in E_∞, then it is reducible by E_*.
(vi) If $u \sim_{E_i} v$, then $u \sim_{E_{i+1}} v$.
(vii) $u \sim_{E_\infty} v$ if and only if $u \sim_{E_*} v$ if and only if $u{\downarrow_{E_*}} = v{\downarrow_{E_*}}$.
(viii) If $u \sim_{E_i} v$, then $u \sim_{E_*} v$.

The *limit* of a branch is defined to be the tuple $N_\infty \,[]\, E_*$ with $N_\infty = \bigcup_{i\geq 0} N_i{\downarrow_{E_*}}$.

Let $\mathcal{F}_\mathcal{B}$ denote the set of all tableau formulae and rules on $\mathcal{B}$, i.e., $\mathcal{F}_\mathcal{B} = \bigcup_{i\geq 0}(N_i \cup E_i)$. And, let $\mathcal{T}_\mathcal{B}$ denote the set of all terms occurring in a branch $\mathcal{B}$, i.e., $\mathcal{T}_\mathcal{B} = \{s \mid s \text{ is a term over } \Sigma \cup K \text{ occurring in } \mathcal{F}_\mathcal{B}\}$.

For the rest of the section we assume $\mathcal{B}$ denotes any open, fully expanded branch in a Tab(ub,cc)-derivation.

Lemma 2. *Formulae and terms have the following properties.*

(i) *If* $s : \phi \in \mathcal{F}_\mathcal{B}$ *then* $s{\downarrow_{E_*}} : \phi \in N_\infty$.
(ii) *If* $R(s,t) \in \mathcal{F}_\mathcal{B}$ *then* $R(s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \in N_\infty$.
(iii) *If* $s \approx t \in \mathcal{F}_\mathcal{B}$ *then* $s{\downarrow_{E_*}} \approx t{\downarrow_{E_*}} \in N_\infty$.
(iv) *If* $s \not\approx t \in \mathcal{F}_\mathcal{B}$ *then* $s{\downarrow_{E_*}} \not\approx t{\downarrow_{E_*}} \in N_\infty$.

Lemma 3. N_∞ *has the following properties.*

(i) *Let* F *be a formula of the form* $R(s,t)$, $s \not\approx t$ *(where* $s \neq t$*),* $s : \Box\phi$, $s : p$ *or* $s : \neg p$. *Then,* $F \in N_\infty$ *implies there is an index* i *such that for all* $j \geq i$, $F \in N_j$.
(ii) *a.* *If* $s : \neg\neg\phi \in N_\infty$, *then there is an index* i *and an* $s' \in \mathcal{T}_\mathcal{B}$ *such that* $s' : \neg\neg\phi \in N_i$, $s' : \phi \in N_{i+1}$, *and* $s'{\downarrow_{E_*}} = s$.
b. *If* $s : \neg(\phi_1 \vee \ldots \vee \phi_k) \in N_\infty$, *then there is an index* i *and an* $s' \in \mathcal{T}_\mathcal{B}$ *such that* $s' : \neg(\phi_1 \vee \ldots \vee \phi_k) \in N_i$, $\{s' : {\sim}\phi_1,\ \ldots,\ s' : {\sim}\phi_k\} \subseteq N_{i+1}$, *and* $s'{\downarrow_{E_*}} = s$.
c. *If* $s : \phi_1 \vee \ldots \vee \phi_k \in N_\infty$, *then there is an index* i, *an* l *with* $1 \leq l \leq k$ *and an* $s' \in \mathcal{T}_\mathcal{B}$ *such that* $s' : \phi_1 \vee \ldots \vee \phi_k \in N_i$, $s' : \phi_l \in N_{i+1}$, *and* $s'{\downarrow_{E_*}} = s$.
d. *If* $s : \neg\Box\phi \in N_\infty$, *then there is an index* i, *a* $d \in K$ *and an* $s' \in \mathcal{T}_\mathcal{B}$ *such that* $s' : \neg\Box\phi \in N_i$, $\{R(s',d),\ d : {\sim}\phi\} \subseteq N_{i+1}$, $f_{\neg\Box\phi}(s') \to d \in E_{i+1}$, *and* $s'{\downarrow_{E_*}} = s$.

Properties (i) and (ii) can be combined to show N_∞ is a kind of Hintikka set:

Lemma 4. *(i)* *If* $s : \neg\neg\phi \in N_\infty$ *then* $s : \phi \in N_\infty$.
(ii) *If* $s : \neg(\phi_1 \vee \ldots \vee \phi_k) \in N_\infty$ *then* $\{s : {\sim}\phi_1,\ \ldots,\ s : {\sim}\phi_k\} \subseteq N_\infty$.
(iii) *If* $s : \phi_1 \vee \ldots \vee \phi_k \in N_\infty$ *then* $s : \phi_l \in N_\infty$ *for some* l, $1 \leq l \leq k$.
(iv) *If* $s : \neg\Box\phi \in N_\infty$ *then* $\{R(s,d),\ d : {\sim}\phi\} \subseteq N_\infty$ *for some* d *such that* $f_{\neg\Box\phi}(s) \sim_{E_*} d$.
(v) *If* $s : \Box\phi \in N_\infty$ *and* $R(s,t) \in N_\infty$ *then* $t : \phi \in \mathcal{F}_\mathcal{B}$ *and* $t : \phi \in N_\infty$.

Lemma 5. *If* $\{R(c,d), R(d,d')\} \subseteq N_\infty$ *then* $R(c,d') \in N_\infty$.

Next we show any open, fully expanded branch $\mathcal{B}$ induces a certain *canonical interpretation*, denoted by $\mathcal{I}(\mathcal{B})$. We define $\mathcal{I}(\mathcal{B})$ to be the interpretation $(U^{\mathcal{I}(\mathcal{B})}, \cdot^{\mathcal{I}(\mathcal{B})})$ with $U^{\mathcal{I}(\mathcal{B})} = \{s{\downarrow_{E_*}} \mid s \in \mathcal{T}_\mathcal{B}\}$ and $\cdot^{\mathcal{I}(\mathcal{B})}$ the homomorphic extension of the following.

$$s^{\mathcal{I}(\mathcal{B})} = s{\downarrow_{E_*}} \quad \text{if } s \in \mathcal{T}_\mathcal{B} \qquad p^{\mathcal{I}(\mathcal{B})} = \{s{\downarrow_{E_*}} \mid s : p \in \mathcal{F}_\mathcal{B}\}$$
$$R^{\mathcal{I}(\mathcal{B})} = \{(s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \mid R(s,t) \in \mathcal{F}_\mathcal{B}\} \qquad \approx^{\mathcal{I}(\mathcal{B})} = \{(s{\downarrow_{E_*}}, s{\downarrow_{E_*}}) \mid s \in \mathcal{T}_\mathcal{B}\}$$

$U^{\mathcal{I}(\mathcal{B})}$ is not empty, since every input set is non-empty and contains at least one term. We have that:

$$\begin{aligned} x \in (\neg\phi)^{\mathcal{I}(\mathcal{B})} &\text{ iff } x \in U^{\mathcal{I}(\mathcal{B})} \setminus \phi^{\mathcal{I}(\mathcal{B})} \\ x \in (\phi_1 \vee \ldots \vee \phi_k)^{\mathcal{I}(\mathcal{B})} &\text{ iff } x \in \phi_1^{\mathcal{I}(\mathcal{B})} \cup \ldots \cup \phi_k^{\mathcal{I}(\mathcal{B})} \\ x \in (\Box\phi)^{\mathcal{I}(\mathcal{B})} &\text{ iff for all } y \in U^{\mathcal{I}(\mathcal{B})} \text{ if } (x,y) \in R^{\mathcal{I}(\mathcal{B})} \text{ then } y \in \phi^{\mathcal{I}(\mathcal{B})} \end{aligned}$$

and

$$\begin{aligned} &\mathcal{I}(\mathcal{B}) \models s : \phi \text{ iff } s{\downarrow_{E_*}} \in \phi^{\mathcal{I}(\mathcal{B})} && \mathcal{I}(\mathcal{B}) \models s \approx t \text{ iff } (s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \in \approx^{\mathcal{I}(\mathcal{B})} \\ &\mathcal{I}(\mathcal{B}) \models R(s,t) \text{ iff } (s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \in R^{\mathcal{I}(\mathcal{B})} && \mathcal{I}(\mathcal{B}) \models s \not\approx t \text{ iff } (s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \notin \approx^{\mathcal{I}(\mathcal{B})}. \end{aligned}$$

$\mathcal{I}(\mathcal{B})$ is thus an interpretation. Our aim now is to show $\mathcal{I}(\mathcal{B})$ is a **K(tr,dp)**-model for each tableau formula on an open, fully expanded branch $\mathcal{B}$.

Lemma 6. *(i) If $R(s,t) \in \mathcal{F}_\mathcal{B}$ then $(s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \in R^{\mathcal{I}(\mathcal{B})}$.*
(ii) If $(s{\downarrow_{E_*}}, t) \in R^{\mathcal{I}(\mathcal{B})}$ and $s : \Box\phi \in \mathcal{F}_\mathcal{B}$ then $t : \phi \in \mathcal{F}_\mathcal{B}$.

Lemma 7. *If $s : \phi \in \mathcal{F}_\mathcal{B}$ then $s{\downarrow_{E_*}} \in \phi^{\mathcal{I}(\mathcal{B})}$.*

Lemma 6(i) and Lemma 7 imply that every tableau formula of the form $R(s,t)$ and $s : \phi$ occurring on an open, fully expanded branch $\mathcal{B}$ is reflected in $\mathcal{I}(\mathcal{B})$, i.e., holds in $\mathcal{I}(\mathcal{B})$. Next we show that all equations and inequations on $\mathcal{B}$ are reflected in $\mathcal{I}(\mathcal{B})$.

Lemma 8. *(i) If $s \approx t \in \mathcal{F}_\mathcal{B}$ or $s \to t \in \mathcal{F}_\mathcal{B}$ then $(s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \in \approx^{\mathcal{I}(\mathcal{B})}$.*
(ii) If $s \not\approx t \in \mathcal{F}_\mathcal{B}$ then $(s{\downarrow_{E_*}}, t{\downarrow_{E_*}}) \notin \approx^{\mathcal{I}(\mathcal{B})}$.

It remains to show:

Lemma 9. *(i) If $(x,y) \in R^{\mathcal{I}(\mathcal{B})}$ and $(y,z) \in R^{\mathcal{I}(\mathcal{B})}$ then $(x,z) \in R^{\mathcal{I}(\mathcal{B})}$.*
(ii) $R^{\mathcal{I}(\mathcal{B})}$ satisfies the frame condition (1).

Finally, we can conclude:

Lemma 10. *The interpretation $\mathcal{I}(\mathcal{B})$ is a **K(tr,dp)**-model for each tableau formula on the branch $\mathcal{B}$.*

Consequently, if for a finite set of tableau formulae an open, fully expanded branch $\mathcal{B}$ can be constructed, then the input set is satisfiable, because the canonical interpretation $\mathcal{I}(\mathcal{B})$ is a **K(tr,dp)**-model. This means the tableau calculus Tab(ub,cc) is constructively complete, from which it immediately follows that it is also refutationally complete. This completes the proof of Theorem 2.

Table 1. Side-conditions for restricted forms of blocking. $\tau(s) = \{\psi \mid s : \psi \in N_i\}$, where N_i denotes the set of tableau formulae in the current state.

Name Suffix	Restriction
ancestor	s is a proper subterm of t
predecessor	$t = f_\psi(s)$ for some ψ, or $t = g(s)$
equality	$\tau(s) = \tau(t)$
subset	$\tau(s) \subseteq \tau(t)$
noS	$\{s, t\} \not\subseteq S$, where S is a finite set of terms
exists	$s : \Diamond\psi$, $t : \Diamond\psi$
δ^*	the leading symbol of t is a function symbol and occurs in the rules (i.e., f_ϕ and g)

6 Ancestor Blocking and Other Forms of Blocking

For many modal, description and hybrid logics that have the finite model property, termination of a tableau calculus can be enforced by using blocking. The unrestricted blocking rule (ub) (in Fig. 1) permits to introduce a case analysis for arbitrary pairs of terms s and t that are identified and merged. It is obvious that this rule can also be used together with congruence closure, see Fig. 2; in fact, since any relevant term is represented by some constant in K in E-normal form, it is sufficient to consider such constants. For many modal logics, however, more restricted forms of blocking are sufficient to guarantee termination. The question is how these restrictions can be checked in our setting.

Common restricted forms of blocking are equality (or subset) predecessor blocking, equality (or subset) ancestor blocking, anywhere blocking, dynamic blocking, pair-wise blocking and pattern-based blocking (c.f., e.g. [2, 12]). These can be emulated by imposing restrictions on the application of the (ub) rule and using appropriate search strategies [15, 16, 14]. Table 1 gives examples of some restrictions that may sensibly be imposed on the (ub)-rule in tableau systems without congruence closure. Restricting the application of the (ub)-rule by the ancestor condition is what is known as *sound ancestor blocking*, restricting it by both the ancestor and the equality conditions is what is known as *sound ancestor equality blocking* [15]. In this way each combination of conditions in the table defines a blocking rule. The (ub-noS)-rule excludes the terms in S (a fixed, finite set of terms) from involvement in any blocking steps. If S is taken to be the set of terms occurring in the initially given set of tableau formulae, then blocking is applied only to terms created during the inference process. An alternative way of achieving this is to use the (ub-δ^*)-rule. If this rule is applied eagerly immediately after the application of a witness-creating rule, then this emulates the use of the (δ^*)-rule of [7]. E.g., the (δ^*)-version of the ($\neg\Box$)-rule is:

$$\frac{s : \neg\Box\phi}{R(s, t_0),\ t_0 : {\sim}\phi \mid \ldots \mid R(s, t_n),\ t_n : {\sim}\phi \mid R(s, f_{\neg\Box\phi}(s)),\ f_{\neg\Box\phi}(s) : {\sim}\phi},$$

where $t_0, \ldots, t_n$ are all the terms occurring in the current state.

The rules for tableau systems combined with congruence closure corresponding to these restricted forms of blocking are appropriate restrictions of the (ub)-rule in Figure 2. In all cases the adaptation is routine.

We only consider the adaptation for the case of ancestor blocking explicitly. In our framework, the only terms that occur in the left-hand side N of a tableau state $N \,[\!]\, E$ are constants from K. The syntactical subterm test must therefore be replaced by checking whether some terms represented by these constants are subterms of each other. The following lemma shows how this property can be tested efficiently. We assume that the Deduction rule and Collapse rule have been applied exhaustively, so that E is left-reduced (that is, no left-hand side of a rewrite rule is a subterm of the left-hand side of another rewrite rule); moreover we know that E is terminating by construction.

Lemma 11. *Let E be a set of C- and D-rules that is terminating and left-reduced. Let $G_E = (\mathcal{V}, \mathcal{E})$ be a directed graph, such that the vertex set $\mathcal{V}$ equals K, and such that there is an edge from c to c' in $\mathcal{E}$ whenever E contains a D-rule $h(\dots, c', \dots) \to c$ or a C-rule $c' \to c$. Let c_s and c_t be two distinct constants in K in E-normal form. Then, the following two properties are equivalent:*

(i) There exist terms s and t such that s is a proper subterm of t, and c_s and c_t are the E-normal forms of s and t.
(ii) c_s is reachable from c_t in G_E.

The adapted ancestor blocking rule is:

$$\text{(ub-ancestor)}\ \frac{N \;[\!]\; E}{N,\ c_s \approx c_t \;[\!]\; E \mid N,\ c_s \not\approx c_t \;[\!]\; E}$$

provided c_s and c_t are distinct constants from K in E-normal form; N does not contain an inequation $c_s \not\approx c_t$; and c_s is reachable from c_t in G_E.

Computing the set of all reachable vertices in a directed graph for some given initial node can be done in linear time, for instance by using breadth-first search. To find an arbitrary pair c_s, c_t that satisfies all the side conditions of the ancestor blocking rule, we could naively repeat breadth-first search for each potential initial node and test the remaining properties for every pair until we find a pair that satisfies all properties. This gives a quadratic time algorithm.

Lemma 11 shows that every pair of terms s, t such that s is a subterm of t corresponds to a pair of constants c_s, c_t in E-normal form such that c_s is reachable from c_t in G_E, and vice versa. This correspondence, however, is not one-to-one. In general, several pairs of terms are mapped to the same pair of constants, so that the number of constant pairs that could be considered in a tableau derivation is usually smaller than the number of term pairs.

We note that for the logic **K(tr,dp)** it would not make sense to use predecessor blocking.

7 Conclusion

This paper has presented an abstract semantic tableau system with abstract ways of handling both blocking and equality. The focus has been on showing how the abstract congruence closure system of [3] can be combined with a semantic tableau system for a modal logic. In contrast to earlier work, we use a "white box" integration, so that the abstract congruence closure is not only used to check entailed equalities, but also to normalise tableau formulae, so that logically equivalent formulae are eliminated. The particular modal tableau system was chosen to illustrate the most important ideas of integrating congruence closure so that the integration can be extended to other tableau systems for other modal, description, and hybrid logics. We believe the case study is general enough to work out how to combine congruence closure with Smullyan-type tableau rules for first-order logic, or incorporate it into bottom-up model generation and hypertableau methods. The ideas are also applicable in tableau systems obtained in the tableau synthesis framework of [21]. The only case that we have not considered is tableau rules with inequalities in premise position; for first-order representable logics this is without loss of generality, because equivalent tableau systems always exist without such occurrences. It remains to generalise the proofs for all these cases, which will be future work.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
3. L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 31(2):129–168, 2003.
4. P. Baumgartner and R. A. Schmidt. Blocking and other enhancements for bottom-up model generation methods. In *Proc. IJCAR'06*, volume 4130 of *LNAI*, pages 125–139. Springer, 2006.
5. B. Beckert. Semantic tableaux with equality. *Journal of Logic and Computation*, 7(1):39–58, 1997.
6. T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007.
7. F. Bry and R. Manthey. Proving finite satisfiability of deductive databases. In *Proc. CSL'87*, volume 329 of *LNCS*, pages 44–55. Springer, 1988.
8. A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 611–706. Elsevier, 2001.
9. M. Giese. Superposition-based equality handling for analytic tableaux. *Journal of Automated Reasoning*, 38(1-3):127–153, 2007.
10. R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer, 1999.
11. M. Kaminski. *Incremental Decision Procedures for Modal Logics with Nominals and Eventualities.* PhD thesis, Universität des Saarlandes, Germany, 2012.

12. M. Kaminski and G. Smolka. Hybrid tableaux for the difference modality. *Electronic Notes in Theoretical Computer Science*, 231:241–257, 2009.
13. T. Käufl and N. Zabel. Cooperation of decision procedures in a tableau-based theorem prover. *Revue d'Intelligence Artificielle*, 4(3):99–126, 1990.
14. M. Khodadadi. *Exploration of Variations of Unrestricted Blocking for Description Logics*. PhD thesis, The University of Manchester, UK, 2015.
15. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. An abstract tableau calculus for the description logic $\mathcal{SHOI}$ using unrestricted blocking and rewriting. In *Proc. DL'12*, volume 846 of *CEUR Workshop Proc.* CEUR-WS.org, 2012.
16. M. Khodadadi, R. A. Schmidt, and D. Tishkovsky. A refined tableau calculus with controlled blocking for the description logic $\mathcal{SHOI}$. In *Proc. TABLEAUX'13*, volume 8123 of *LNCS*, pages 188–202. Springer, 2013.
17. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.
18. R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4):557–580, 2007.
19. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.
20. R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *Proc. ISWC'07 + ASWC'07*, volume 4825 of *LNCS*, pages 438–451. Springer, 2007.
21. R. A. Schmidt and D. Tishkovsky. Automated synthesis of tableau calculi. *Logical Methods in Computer Science*, 7(2):1–32, 2011.
22. R. A. Schmidt and D. Tishkovsky. Using tableau to decide description logics with full role negation and identity. *ACM Transactions on Computational Logic*, 15(1), 2014.
23. R. A. Schmidt and U. Waldmann. Modal tableau systems with blocking and congruence closure. eScholar Report uk-ac-man-scw:268816, University of Manchester, 2015.
24. D. Tishkovsky and R. A. Schmidt. Refinement in the tableau synthesis framework, 2013. arXiv e-Print 1305.3131v1 [cs.LO].
25. D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL2. In *Proc. JELIA'12*, volume 7519 of *LNCS*, pages 492–495. Springer, 2012.