# One-Pass Tableaux for Computation Tree Logic

Pietro Abate[1], Rajeev Goré[1], and Florian Widmann[1,2]

[1] The Australian National University
Canberra ACT 0200, Australia
[2] Logic and Computation Programme
Canberra Research Laboratory
NICTA Australia
{Pietro.Abate,Rajeev.Gore,Florian.Widmann}@anu.edu.au

**Abstract.** We give the first single-pass ("on the fly") tableau decision procedure for computational tree logic (CTL). Our method extends Schwendimann's single-pass decision procedure for propositional linear temporal logic (PLTL) but the extension is non-trivial because of the interactions between the branching inherent in CTL-models, which is missing in PLTL-models, and the "or" branching inherent in tableau search. Our method extends to many other fix-point logics like propositional dynamic logic (PDL) and the logic of common knowledge (LCK).

The decision problem for CTL is known to be EXPTIME-complete, but our procedure requires 2EXPTIME in the worst case. A similar phenomenon occurs in extremely efficient practical single-pass tableau algorithms for very expressive description logics with EXPTIME-complete decision problems because the 2EXPTIME worst-case behaviour rarely arises. Our method is amenable to the numerous optimisation methods invented for these description logics and has been implemented in the Tableau Work Bench (`twb.rsise.anu.edu.au`) without these extensive optimisations. Its one-pass nature also makes it amenable to parallel proof-search on multiple processors.

## 1 Introduction and Motivation

Propositional fix-point logics like propositional linear temporal logic (PLTL), computation tree logic (CTL) and full computation tree logic (CTL*) are useful for digital circuit verification [10] and reasoning about programs [18]. The usual route is to use model-checking to ensure that a given model satisfies certain desirable properties since this can be done in time linear in the size of the model. Model-checking cannot answer the general question of whether every model for a given set of formulae $\Gamma$ satisfies a certain property $\varphi$: that is model-checking cannot be used to perform automated deduction in the chosen fix-point logic. The decision problems for fix-point logics are typically at least PSPACE-complete (PLTL), and are often EXPTIME-complete (CTL) or even 2EXPTIME-complete (CTL*). These logics are all fragments of monadic second order logic whose decision problem has non-elementary complexity when restricted to (infinite) tree-models, meaning that its complexity is a tower of

N. Dershowitz and A. Voronkov (Eds.): LPAR 2007, LNAI 4790, pp. 32–46, 2007.

exponentials of a height determined by the size of the initial formula. Consequently, automated theorem provers tailored for specific fix-point logics are of importance in computer science.

The main methods for automating deduction in logics like PLTL and CTL are optimal tableau-based methods [24,6], optimal automata-based methods [22] and optimal resolution-based methods [7,4]. The inverse method has also been applied to simpler modal logics like K [23], but it is possible to view this approach as an automata-based method [2]. We are trying to obtain further details of a new method for PLTL which appears to avoid an explicit loop-check [8].

Most existing automated theorem provers for the PSPACE-complete fix-point logic PLTL are tableau-based [16,9,17,20], but resolution provers for PLTL have also been developed recently [13]. It is easy to construct examples where the (goal-directed) tableau-based provers out-perform the resolution provers, and vice versa [13], so both methods remain of interest. Theorem provers based on the always optimal automata-based methods which do not actually build the required automata [21] are still in their infancy because good optimisation techniques have not been developed to date.

For CTL, however, we know of no efficient implemented automated theorem provers, even though tableau-based [6,19], resolution-based [4] and automata-based [22] deduction methods for CTL are also known.

The simplest non-technical explanation is that proof-search in many modal logics requires some form of "loop check" to guarantee termination, but fix-point logics require a further test to distinguish a "good loop" that represents a path in a model from a "bad loop" that represents an infinite branch with no hope of ever giving a model. The harder the decision problem, the greater the difficulty of separating good loops from bad loops.

Most tableau-based methods for fix-point logics solve this problem using a two-pass procedure [24,5,6]. The first pass applies the tableau rules to construct a finite rooted cyclic graph. The second pass prunes nodes that are unsatisfiable because they contain contradictions like $\{p, \neg p\}$, and also remove nodes which give rise to "bad loops". The main practical disadvantage of such two-pass methods is that the cyclic graph built in the first pass has a size which is *always* exponential in the size of the initial formula. So the very act of building this graph immediately causes EXPTIME behaviour even in the average case.

One-pass tableau methods avoid this bottle-neck by building a rooted cyclic *tree* (where all cyclic edges loop back to ancestors) one branch at a time, using backtracking. The experience from one-pass tableaux for very expressive description logics [12] of similar worst-case complexity shows that their average case behaviour is often much better since the given formulae may not contain the full complexity inherent in the decision problem, particularly if the formula arises from real-world applications. Of course, there is no free lunch, since in the worst case, these one-pass methods may have significantly worse behaviour than the known optimal behaviour: 2EXPTIME than EXPTIME in the case of CTL for example. Moreover, the method for separating "good loops" from "bad loops" becomes significantly more complicated since it cannot utilise the global view

offered by a graph built during a previous pass. Ideally, we want to evaluate each branch on its own during construction, or during backtracking, using only information which is "local" to this branch since this allows us to explore these branches in parallel using multiple processors.

Implemented one-pass [17,20] and two-pass [16] tableau provers already exist for PLTL. A comparison between them [13] shows that the median running time for Janssen's highly optimised two-pass prover for PLTL is greater than the median running time for Schwendimann's not-so-optimised one-pass prover for PLTL [20] *for problems which are deliberately constructed to be easy for tableau provers*, indicating that the two-pass prover spends most of its time in the first pass building the cyclic graph. There is also a one-pass "tableau" method for propositional dynamic logic (PDL) [3] which constructs a rooted cyclic tree and uses a finite collection of automata, pre-computed from the initial formula, to distinguish "good loops" from "bad loops", but the expressive powers of PDL and CTL are orthogonal: each can express properties which cannot be expressed in the other. But we know of no one-pass tableau method for CTL or its extensions.

For many applications, the ability to exhibit a (counter) model for a formula $\varphi$ is just as important as the ability to decide whether $\varphi$ is a theorem. In digital circuit verification, for example, if the circuit does not obey a desired property expressed by a formula $\varphi$, then it is vital to exhibit a (CTL) counter-model which falsifies $\varphi$ so that the circuit can be modified.

Finally, the current Gentzen-style proof-theory of fix-point logics [14,15] requires either infinitary rules, or worst-case finitary branching rules, or "cyclic proofs" with sequents built from formula occurrences or "focussed formulae".

We present a one-pass tableau method for automating deduction in CTL which has the following properties:

Ease of implementation: although our tableau rules are cumbersome to describe and difficult to prove sound and complete, they are extremely easy to implement since they build a rooted cyclic tree as usual, and the only new operations they require are set intersection, set membership, and the operations of min/max on integers;

Ease of optimisation: our method can be optimised using techniques which have proved successful for (one-pass) tableaux for description logics [11];

Ease of generating counter-models and proofs: the soundness proof of our systematic tableau procedure for testing CTL-satisfiability immediately gives an effective procedure for turning an "open" tableau into a CTL-model;

Ease of generating proofs: our tableau calculus can be trivially turned into a cut-free Gentzen-style calculus with "cyclic proofs" where sequents are built from sets of formulae rather than multisets of formula *occurrences* or occurrences of "focused formulae". Moreover, unlike existing cut-free Gentzen-style calculi for fix-point logics [14,15] we can give an optimal, rather than worst-case, bound for the finitary version of the omega rule for CTL;

Potential for parallelisation: our current rules build the branches independently, but combine their results during backtracking, so it is possible to implement our procedure on a bank of parallel processors.

Working Prototype: we have implemented a (non-parallelised) prototype of our basic tableau method using the Tableau Work Bench (`twb.rsise.anu.edu.au`) which allows users to test arbitrary CTL formulae over the web.

Our work is one step toward the holy grail of an efficient tableau-based automated theorem prover for the full computational tree logic CTL* [19].

## 2 Syntax, Semantics and Hintikka Structures

**Definition 1.** *Let* AP *denote the set* $\{p_0, p_1, p_2, \dots\}$ *of propositional variables. The set* Fml *of all formulae of the logic CTL is inductively defined as follows:*

1. $\text{AP} \subset \text{Fml}$*;*
2. *if* $\varphi$ *is in* Fml*, then so are* $\neg\varphi$*,* $EX\varphi$*, and* $AX\varphi$*;*
3. *if* $\varphi$ *and* $\psi$ *are in* Fml*, then so are* $\varphi \wedge \psi$*,* $\varphi \vee \psi$*,* $E(\varphi\, U\, \psi)$*,* $A(\varphi\, U\, \psi)$*,* $A(\varphi\, B\, \psi)$*, and* $E(\varphi\, B\, \psi)$*.*

*A formula of the form* $EX\varphi$*,* $AX\varphi$*,* $E(\varphi\, U\, \psi)$*, or* $A(\varphi\, U\, \psi)$ *is called an EX-, AX-, EU-, or AU-formula, respectively. Let* FmlEU *and* FmlAU *denote the set of all EU- and AU-formulae, respectively.*

Implication, equivalence, and $\top$ are not part of the core language but can be defined as $\varphi \to \psi := \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$, and $\top := p_0 \vee \neg p_0$.

**Definition 2.** *A* transition frame *is a pair* $(W, R)$ *where* $W$ *is a non-empty set of worlds and* $R$ *is a total binary relation over* $W$ *(*i.e. $\forall w \in W.\, \exists v \in W.\, w\, R\, v$*).*

**Definition 3.** *Let* $(W, R)$ *be a transition frame. A* transition sequence $\sigma$ *in* $(W, R)$ *is an infinite sequence* $\sigma_0, \sigma_1, \sigma_2, \dots$ *of worlds in* $W$ *such that* $\sigma_i\, R\, \sigma_{i+1}$ *for all* $i \in \mathbb{N}$*. For* $w \in W$*, a* $w$-sequence $\sigma$ *in* $(W, R)$ *is a transition sequence in* $(W, R)$ *with* $\sigma_0 = w$*. For* $w \in W$*, let* $\mathcal{B}(w)$ *be the set of all* $w$*-sequences in* $(W, R)$ *(we assume that* $(W, R)$ *is clear from the context).*

**Definition 4.** *A* model $M = (W, R, L)$ *is a transition frame* $(W, R)$ *and a labelling function* $L : W \to 2^{\text{AP}}$ *which associates with each world* $w \in W$ *a set* $L(w)$ *of propositional variables true at world* $w$*.*

**Definition 5.** *Let* $M = (W, R, L)$ *be a model. The* satisfaction relation $\Vdash$ *is defined inductively as follows:*

$$
\begin{array}{ll}
M, w \Vdash p & \textit{iff } p \in L(w), \textit{ for } p \in \text{AP} \\
M, w \Vdash \neg\psi & \textit{iff } M, w \nVdash \psi \\
M, w \Vdash \varphi \wedge \psi & \textit{iff } M, w \Vdash \varphi \;\&\; M, w \Vdash \psi \\
M, w \Vdash \varphi \vee \psi & \textit{iff } M, w \Vdash \varphi \textit{ or } M, w \Vdash \psi \\
M, w \Vdash EX\varphi & \textit{iff } \exists v \in W.\, w\, R\, v \;\&\; M, v \Vdash \varphi \\
M, w \Vdash AX\varphi & \textit{iff } \forall v \in W.\, w\, R\, v \;\Rightarrow\; M, v \Vdash \varphi \\
M, w \Vdash E(\varphi\, U\, \psi) & \textit{iff } \exists \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}.\, [M, \sigma_i \Vdash \psi \;\&\; \forall j < i.\, M, \sigma_j \Vdash \varphi] \\
M, w \Vdash A(\varphi\, U\, \psi) & \textit{iff } \forall \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}.\, [M, \sigma_i \Vdash \psi \;\&\; \forall j < i.\, M, \sigma_j \Vdash \varphi] \\
M, w \Vdash E(\varphi\, B\, \psi) & \textit{iff } \exists \sigma \in \mathcal{B}(w).\, \forall i \in \mathbb{N}.\, [M, \sigma_i \Vdash \psi \;\Rightarrow\; \exists j < i.\, M, \sigma_j \Vdash \varphi] \\
M, w \Vdash A(\varphi\, B\, \psi) & \textit{iff } \forall \sigma \in \mathcal{B}(w).\, \forall i \in \mathbb{N}.\, [M, \sigma_i \Vdash \psi \;\Rightarrow\; \exists j < i.\, M, \sigma_j \Vdash \varphi]\ .
\end{array}
$$

**Definition 6.** *A formula* $\varphi \in \mathrm{Fml}$ *is* satisfiable *iff there is a model* $M = (W, R, L)$ *and some* $w \in W$ *such that* $M, w \Vdash \varphi$. *A formula* $\varphi \in \mathrm{Fml}$ *is* valid *iff* $\neg\varphi$ *is not satisfiable.*

**Definition 7.** *A formula* $\varphi \in \mathrm{Fml}$ *is in* negation normal form *if the symbol* $\neg$ *appears only immediately before propositional variables. For every formula* $\varphi \in \mathrm{Fml}$, *we can obtain a formula* $\mathrm{nnf}(\varphi)$ *in negation normal form by pushing negations inward as far as possible (*e.g. *by using de Morgan's laws) such that* $\varphi \leftrightarrow \mathrm{nnf}(\varphi)$ *is valid. We define* $\sim\varphi := \mathrm{nnf}(\neg\varphi)$.

Note that $E(\varphi\, B\, \psi) \leftrightarrow \neg A(\neg\varphi\, U\, \psi)$ and $A(\varphi\, B\, \psi) \leftrightarrow \neg E(\neg\varphi\, U\, \psi)$ are valid.

**Table 1.** Smullyan's $\alpha-$ and $\beta-$notation to classify formulae

| $\alpha$ | $\alpha_1$ | $\alpha_2$ |
|---|---|---|
| $\varphi \wedge \psi$ | $\varphi$ | $\psi$ |
| $E(\varphi\, B\, \psi)$ | $\sim\psi$ | $\varphi \vee EXE(\varphi\, B\, \psi)$ |
| $A(\varphi\, B\, \psi)$ | $\sim\psi$ | $\varphi \vee AXA(\varphi\, B\, \psi)$ |

| $\beta$ | $\beta_1$ | $\beta_2$ |
|---|---|---|
| $\varphi \vee \psi$ | $\varphi$ | $\psi$ |
| $E(\varphi\, U\, \psi)$ | $\psi$ | $\varphi \wedge EXE(\varphi\, U\, \psi)$ |
| $A(\varphi\, U\, \psi)$ | $\psi$ | $\varphi \wedge AXA(\varphi\, U\, \psi)$ |

**Proposition 8.** *In the notation of Table 1, the formulae of the form* $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ *and* $\beta \leftrightarrow \beta_1 \vee \beta_2$ *are valid.*

Note that some of these equivalences require the fact that the binary relation of every model is total.

**Definition 9.** *Let* $\phi \in \mathrm{Fml}$ *be a formula in negation normal form. The* closure $\mathrm{cl}(\phi)$ *of* $\phi$ *is the least set of formulae such that:*

1. *Each subformula of* $\phi$, *including* $\phi$ *itself, is in* $\mathrm{cl}(\phi)$;
2. *If* $E(\varphi\, B\, \psi)$ *is in* $\mathrm{cl}(\phi)$, *then so are* $EXE(\varphi\, B\, \psi)$ *and* $\varphi \vee EXE(\varphi\, B\, \psi)$;
3. *If* $A(\varphi\, B\, \psi)$ *is in* $\mathrm{cl}(\phi)$, *then so are* $AXA(\varphi\, B\, \psi)$ *and* $\varphi \vee AXA(\varphi\, B\, \psi)$;
4. *If* $E(\varphi\, U\, \psi)$ *is in* $\mathrm{cl}(\phi)$, *then so are* $EXE(\varphi\, U\, \psi)$ *and* $\varphi \wedge EXE(\varphi\, U\, \psi)$;
5. *If* $A(\varphi\, U\, \psi)$ *is in* $\mathrm{cl}(\phi)$, *then so are* $AXA(\varphi\, U\, \psi)$ *and* $\varphi \wedge AXA(\varphi\, U\, \psi)$.

*The* extended closure $\mathrm{ecl}(\phi)$ *of* $\phi$ *is defined as* $\mathrm{ecl}(\phi) := \mathrm{cl}(\phi) \cup \{\sim\varphi : \varphi \in \mathrm{cl}(\phi)\}$.

**Definition 10.** *A* structure $(W, R, L)$ [*for* $\varphi \in \mathrm{Fml}$] *is a transition frame* $(W, R)$ *and a labelling function* $L : W \to 2^{\mathrm{Fml}}$ *which associates with each world* $w \in W$ *a set* $L(w)$ *of formulae* [*and has* $\varphi \in L(v)$ *for some world* $v \in W$].

**Definition 11.** *A* pre-Hintikka structure $H = (W, R, L)$ [*for* $\varphi \in \mathrm{Fml}$] *is a structure* [*for* $\varphi$] *that satisfies the following conditions for every* $w \in W$ *where* $\alpha$ *and* $\beta$ *are formulae as defined in Table 1:*

$$\begin{aligned}
&\mathrm{H1} : \neg p \in L(w)\ \ (p \in \mathrm{AP}) \ \Rightarrow\ p \notin L(w);\\
&\mathrm{H2} : \alpha \in L(w) \ \Rightarrow\ \alpha_1 \in L(w)\ \&\ \alpha_2 \in L(w);\\
&\mathrm{H3} : \beta \in L(w) \ \Rightarrow\ \beta_1 \in L(w)\ \textit{or}\ \beta_2 \in L(w);\\
&\mathrm{H4} : EX\varphi \in L(w) \ \Rightarrow\ \exists v \in W.\, w\, R\, v\ \&\ \varphi \in L(v);\\
&\mathrm{H5} : AX\varphi \in L(w) \ \Rightarrow\ \forall v \in W.\, w\, R\, v \ \Rightarrow\ \varphi \in L(v).
\end{aligned}$$

*A* Hintikka structure $H = (W, R, L)$ [*for* $\varphi \in \mathrm{Fml}$] *is a pre-Hintikka structure* [*for* $\varphi$] *that additionally satisfies the following conditions:*

$$\mathrm{H6} : E(\varphi\, U\, \psi) \in L(w) \;\Rightarrow\; \exists \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}.\, [\psi \in L(\sigma_i)\ \&\ \forall j < i.\, \varphi \in L(\sigma_j)];$$
$$\mathrm{H7} : A(\varphi\, U\, \psi) \in L(w) \;\Rightarrow\; \forall \sigma \in \mathcal{B}(w).\, \exists i \in \mathbb{N}.\, \psi \in L(\sigma_i).$$

Although $H3$ captures the fix-point semantics of $E(\varphi\, U\, \psi)$ and $A(\varphi\, U\, \psi)$ by "locally unwinding" the fix-point, it does not guarantee a *least* fix-point which requires that $\psi$ has to be true eventually. We therefore additionally need $H6$ and $H7$ which act "globally". Note that $H2$ is enough to capture the correct behaviour of $E(\varphi\, B\, \psi)$ and $A(\varphi\, B\, \psi)$ as they have a *greatest* fix-point semantics.

**Proposition 12.** *A formula* $\varphi \in \mathrm{Fml}$ *in negation normal form is satisfiable iff there exists a Hintikka structure for* $\varphi$.

## 3 A One-Pass Tableau Algorithm for *CTL*

A *tableau algorithm* is a systematic search for a model of a formulae $\phi$. Its data structures are (upside-down) single-rooted finite trees – called *tableaux* – where each node is labelled with a set of formulae that is derived from the formula set of its parent according to some given rules (unless it is the root, of course). The algorithm starts with a single node that is labelled with the singleton set $\{\phi\}$ and incrementally expands the tableau by applying the rules mentioned before to its leaves. The result of the tableau algorithm is a tableau where no more rules can be applied. Such tableaux are called *expanded*. On any branch of the tableau, a node $t$ is an *ancestor* of a node $s$ iff $t$ lies above $s$ on the unique path from the root down to $s$.

An expanded tableau can be associated with a pre-Hintikka structure $H$ for $\phi$, and $\phi$ is satisfiable if and only if $H$ is a Hintikka structure for $\phi$. To be able to determine whether $H$ is a Hintikka structure, the algorithm stores additional information with each node of the tableau using *histories* and *variables* [20]. A history is a mechanism for collecting extra information during proof search and passing it from parents to children. A variable is a mechanism to propagate information from children to parents.

In the following, we restrict ourselves to the tableau algorithm for $CTL$.

**Definition 13.** *A* tableau node $x$ *is of the form* $(\Gamma :: \mathrm{HCr} :: \mathrm{mrk}, \mathrm{uev})$ *where:*

$\Gamma$ *is a set of formulae;*
HCr *is a list of the formula sets of some designated ancestors of* $x$*;*
mrk *is a boolean valued variable indicating whether the node is marked; and*
uev *is a partial function from formulae to* $\mathbb{N}_{>0}$.

The list HCr is the only history since its value in a node is determined by the parent node, whereas mrk and uev are variables since their values in a node are determined by the children. In the following we call tableau nodes just nodes when the meaning is clear.

Informally, the value of mrk at node $x$ is **true** if $x$ is "closed". Since repeated nodes cause "cycles" or "loops", a node that is not "closed" is not necessarily "open" as in traditional tableaux. That is, although we have enough information to detect that further expansion of the node will cause an infinite branch, we may not yet have enough information to determine the status of the node. Informally, if a node $x$ lies on such a "loop" in the tableau, and an "eventuality" $EU$- or $AU$-formula $\varphi$ appears on this loop but remains unfulfilled, then uev of $x$ is defined for $\varphi$ by setting $\mathrm{uev}(\varphi) = n$, where $n$ is the height of the highest ancestor of $x$ which is part of the loop.

We postpone the definition of a rule for a moment and proceed with the definition of a tableau.

**Definition 14.** *A* tableau *for a formula set* $\Gamma \subseteq \mathrm{Fml}$ *and a list of formula sets* HCr *is a tree of tableau nodes with root* $(\Gamma :: \mathrm{HCr} :: \mathrm{mrk}, \mathrm{uev})$ *where the children of a node* $x$ *are obtained by a single application of a rule to* $x$ *(*i.e. *only one rule can be applied to a node). A tableau is* expanded *if no rules can be applied to any of its leaves.*

Note that mrk and uev in the definition are not given but are part of the result as they are determined by the children of the root.

**Definition 15.** *The partial function* $\mathrm{uev}_\bot : \mathrm{Fml} \rightharpoonup \mathbb{N}_{>0}$ *is the constant function that is undefined for all formulae (*i.e. $\mathrm{uev}_\bot(\psi) = \bot$ *for all* $\psi \in \mathrm{Fml}$*).*

*Note 16.* In the following, we use $\Lambda$ to denote a set containing only propositional variables or their negations (*i.e.* $\varphi \in \Lambda \;\Rightarrow\; \exists p \in \mathrm{AP}.\, \varphi = p$ or $\varphi = \neg p$). To focus on the "important" parts of the rule, we use "$\cdots$" for the "unimportant" parts which are passed from node to node unchanged (*e.g.* $(\Gamma :: \cdots :: \cdots)$).

### 3.1 The Rules

*Terminal Rule.*

$$(id)\quad \frac{(\Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{} \;\{p, \neg p\} \subseteq \Gamma \text{ for some } p \in \mathrm{AP}$$

with mrk := **true** and uev := $\mathrm{uev}_\bot$. The intuition is that the node is "closed" so we pass this information up to the parent by putting mrk to **true**, and putting uev as undefined for all formulae.

*Linear ($\alpha$) Rules.*

$$(\wedge)\quad \frac{(\varphi \wedge \psi\ ;\ \Gamma :: \cdots :: \cdots)}{(\varphi\ ;\ \psi\ ;\ \Gamma :: \cdots :: \cdots)} \qquad (D)\quad \frac{(AX\Delta\ ;\ \Lambda :: \cdots :: \cdots)}{(EX(p_0 \vee \neg p_0)\ ;\ AX\Delta\ ;\ \Lambda :: \cdots :: \cdots)}$$

$$(EB)\quad \frac{(E(\varphi\, B\, \psi)\ ;\ \Gamma :: \cdots :: \cdots)}{(\sim\psi\ ;\ \varphi \vee EXE(\varphi\, B\, \psi)\ ;\ \Gamma :: \cdots :: \cdots)}$$

$$(AB)\quad \frac{(A(\varphi\, B\, \psi)\ ;\ \Gamma :: \cdots :: \cdots)}{(\sim\psi\ ;\ \varphi \vee AXA(\varphi\, B\, \psi)\ ;\ \Gamma :: \cdots :: \cdots)}$$

The $\wedge$-rule is standard and the $D$-rule captures the fact that the binary relation of a model is total by ensuring that every potential dead-end contains at least one $EX$-formula. The $EB$- and $AB$-rules capture the fix-point nature of the corresponding formulae according to Prop. 8. These rules do not modify the histories or variables at all.

*Universal Branching ($\beta$) Rules.*

$$(\vee)\quad \frac{(\varphi \vee \psi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{(\varphi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1) \mid (\psi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_2, \mathrm{uev}_2)}$$

$$(EU)\quad \frac{(E(\varphi\, U\, \psi)\ ;\ \Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{(\psi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1) \mid (\varphi\ ;\ EXE(\varphi\, U\, \psi)\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_2, \mathrm{uev}_2)}$$

$$(AU)\quad \frac{(A(\varphi\, U\, \psi))\ ;\ \Gamma :: \cdots :: \mathrm{mrk}, \mathrm{uev})}{(\psi\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_1, \mathrm{uev}_1) \mid (\varphi\ ;\ AXA(\varphi\, U\, \psi)\ ;\ \Gamma :: \cdots :: \mathrm{mrk}_2, \mathrm{uev}_2)}$$

with:

$$\mathrm{mrk} := \mathrm{mrk}_1\ \&\ \mathrm{mrk}_2$$

$$\mathrm{excl}_\phi(f)(\chi) := \begin{cases} \bot & \text{if } \chi = \phi \\ f(\chi) & \text{otherwise} \end{cases}$$

$$\mathrm{uev}'_1 := \begin{cases} \mathrm{uev}_1 & \text{for the } \vee\text{-rule} \\ \mathrm{excl}_{E(\varphi\, U\, \psi)}(\mathrm{uev}_1) & \text{for the } EU\text{-rule} \\ \mathrm{excl}_{A(\varphi\, U\, \psi)}(\mathrm{uev}_1) & \text{for the } AU\text{-rule} \end{cases}$$

$$\min_\bot(f, g)(\chi) := \begin{cases} \bot & \text{if } f(\chi) = \bot \text{ or } g(\chi) = \bot \\ \min(f(\chi), g(\chi)) & \text{otherwise} \end{cases}$$

$$\mathrm{uev} := \begin{cases} \mathrm{uev}_\bot & \text{if } \mathrm{mrk}_1\ \&\ \mathrm{mrk}_2 \\ \mathrm{uev}'_1 & \text{if } \mathrm{mrk}_2\ \&\ \text{not } \mathrm{mrk}_1 \\ \mathrm{uev}_2 & \text{if } \mathrm{mrk}_1\ \&\ \text{not } \mathrm{mrk}_2 \\ \min_\bot(\mathrm{uev}'_1, \mathrm{uev}_2) & \text{otherwise} \end{cases}$$

The $\vee$-rule is standard except for the computation of uev. The $EU$- and $AU$-rules capture the fix-point nature of the $EU$- and $AU$-formulae, respectively, according to Prop. 8. The intuitions of the definitions of the histories and variables are:

mrk: the value of the variable mrk is **true** if the node is "closed", so the definition of mrk just captures the "universal" nature of these rules whereby the parent node is closed if both children are closed.

excl: the definition of $\mathrm{excl}_\phi(f)(\psi)$ just ensures that $\mathrm{excl}_\phi(f)(\phi)$ is undefined.

$\mathrm{uev}'_1$: the definition of $\mathrm{uev}'_1$ ensures that its value is undefined for the principal formulae of the $EU$- and $AU$-rules.

$\min_\bot$: the definition of $\min_\bot$ ensures that we take the minimum of $f(\chi)$ and $g(\chi)$ only when both functions are defined for $\chi$.

uev: if both children are "closed" then the parent is also closed via mrk so we ensure that uev is undefined in this case. If only the right child is closed,

we take $\mathrm{uev}'_1$, which is just $\mathrm{uev}_1$ modified to ensure that it is undefined for the principal *EU*- or *AU*-formula. Similarly if only the left child is closed. Finally, if both children are unmarked, we define uev for all formulae that are defined in the uev of both children but map them to the minimum of their values in the children, and undefine the value for the principal formula.

*Existential Branching Rule.*

$$(EX)\ \frac{EX\varphi_1\ ;\ldots;EX\varphi_n\ ;\ EX\varphi_{n+1}\ ;\ldots;EX\varphi_{n+m}\ ;\ AX\Delta\ ;\ \Lambda \quad :: \mathrm{HCr} :: \mathrm{mrk}, \mathrm{uev}}{\begin{array}{l}\varphi_1\ ;\ \Delta \\ :: \mathrm{HCr}_1 :: \mathrm{mrk}_1, \mathrm{uev}_1\end{array} \mid \cdots \mid \begin{array}{l}\varphi_n\ ;\ \Delta \\ :: \mathrm{HCr}_n :: \mathrm{mrk}_n, \mathrm{uev}_n\end{array}}$$

where:

(1) $\{p, \neg p\} \not\subseteq \Lambda$
(2) $n + m \geq 1$
(3) $\forall i \in \{1, \ldots, n\}.\, \forall j \in \{1, \ldots, \mathrm{len}(\mathrm{HCr})\}.\, \{\varphi_i\} \cup \Delta \neq \mathrm{HCr}[j]$
(4) $\forall k \in \{n+1, \ldots, n+m\}.\, \exists j \in \{1, \ldots, \mathrm{len}(\mathrm{HCr})\}.\, \{\varphi_k\} \cup \Delta = \mathrm{HCr}[j]$

with:

$$\mathrm{HCr}_i := \mathrm{HCr}\ @\ [\{\varphi_i\} \cup \Delta] \text{ for } i = 1, \ldots, n$$

$$\mathrm{mrk} := \textstyle\bigvee_{i=1}^{n} \mathrm{mrk}_i \text{ or } \exists i \in \{1, \ldots, n\}.\, \exists \psi \in \{\varphi_i\} \cup \Delta.\, \bot \neq \mathrm{uev}_i(\psi) > \mathrm{len}(\mathrm{HCr})$$

$$\mathrm{uev}_k(\cdot) := j \in \{1, \ldots, len(\mathrm{HCr})\} \text{ such that } \{\varphi_k\} \cup \Delta = \mathrm{HCr}[j] \text{ for } k = n+1, \ldots, n+m$$

$$\mathrm{uev}(\psi) := \begin{cases} \mathrm{uev}_j(\psi) & \text{if } \psi \in \mathrm{FmlEU}\ \&\ \psi = \varphi_j \quad (j \in \{1, \ldots, n+m\}) \\ l & \text{if } \psi \in \mathrm{FmlAU} \cap \Delta\ \& \\ & \quad l = \max\{\mathrm{uev}_j(\psi) \neq \bot \mid j \in \{1, \ldots, n+m\}\} \\ \bot & \text{otherwise} \end{cases}$$
$$(\text{where } \max(\emptyset) := \bot)$$

Some intuitions are in order:

(1) The $EX$-rule is applicable if the parent node contains no $\alpha$- or $\beta$-formulae and $\Lambda$, which contains propositional variables and their negations only, contains no contradictions.
(2) Both $n$ and $m$ can be zero, but not together.
(3) If $n > 0$, then each $EX\varphi_i$ for $1 \leq i \leq n$ is not "blocked" by an ancestor, and has a child containing $\varphi_i; \Delta$, thereby generating the required $EX$-successor;
(4) If $m > 0$, then each $EX\varphi_k$ for $n+1 \leq k \leq n+m$ is "blocked" from creating its child $\varphi_k; \Delta$ because some ancestor does the job;

$\mathrm{HCr}_i$: is just the HCr of the parent but with an extra entry to extend the "history" of nodes on the path from the root down to the $i^{\text{th}}$ child.

mrk: captures the "existential" nature of this rule whereby the parent is marked if some child is closed or if some child contains a formula whose uev is defined and "loops" lower than the parent. Moreover, if $n$ is zero, then mrk is set to **false** to indicate that this branch is not "closed".

$\text{uev}_k$: for $n+1 \leq k \leq n+m$ the $k^{\text{th}}$ child is blocked by a proxy child higher in the branch. For every such $k$ we set $\text{uev}_k$ to be the *constant* function which maps every formula to the level of this proxy child. Note that this is just a temporary function used to define uev as explained next.

$\text{uev}(\psi)$: for an $EU$-formula $\psi = E(\psi_1 \, U \, \psi_2)$ such that there is a principal formula $EX\varphi_i$ with $\varphi_i = \psi$, we take uev of $\psi$ from the child if $EX\psi$ is "unblocked", or set it to be the *level* of the proxy child higher in the branch if it is "blocked". For an $AU$-formula $\psi = A(\psi_1 \, U \, \psi_2) \in \Delta$, we put uev to be the maximum of the defined values from the real children and the levels of the proxy children. For all other formulae, we put uev to be undefined. The intuition is that a defined $\text{uev}(\psi)$ tells us that there is a "loop" which starts at the parent and eventually "loops" up to some blocking node higher up on the current branch. The actual value of $\text{uev}(\psi)$ tells us the level of the proxy because we cannot distinguish whether this "loop" is "good" or "bad" until we backtrack up to that level.

Note that the $EX$-rule and the *id*-rule are mutually exclusive since their side-conditions cannot be simultaneously true.

### 3.2 Fullpaths, Virtual Successors and Termination of Proof Search

**Definition 17.** *Let $G = (W, R)$ be a directed graph (*e.g. *a tableau where $R$ is just the child-of relation between nodes). A* [full]path $\pi$ *in $G$ is a finite* [*infinite*] *sequence $x_0, x_1, x_2, \ldots$ of nodes in $W$ such that $x_i \, R \, x_{i+1}$ for all $x_i$ except the last node if $\pi$ is finite. For $x \in W$, an* $x$-[full]path $\pi$ *in $G$ is a* [*full*]*path in $G$ that has $x_0 = x$.*

**Definition 18.** *Let $x = (\Gamma :: \text{HCr} :: \text{mrk}, \text{uev})$ be a tableau node, $\varphi$ a formula, and $\Delta$ a set of formulae. We write $\varphi \in x$ $[\Delta \subseteq x]$ to denote $\varphi \in \Gamma$ $[\Delta \subseteq \Gamma]$. The elements* HCr, mrk, *and* uev *of $x$ are denoted by* $\text{HCr}_x$, $\text{mrk}_x$, *and* $\text{uev}_x$, *respectively. The node $x$ is* marked *iff* $\text{mrk}_x$ *is set to* **true**.

**Definition 19.** *Let $x$ be an $EX$-node in a tableau $T$ (*i.e. *an $EX$-rule was applied to $x$). Then $x$ is also called a* state *and the children of $x$ are called* pre-states. *Using the notation of the $EX$-rule, an $EX$-formula $EX\varphi_i \in x$ is* blocked *iff $n+1 \leq i \leq n+m$. For every blocked $EX\varphi_i \in x$ there exists a unique pre-state $y$ on the path from the root of $T$ to $x$ such that $\{\varphi_i\} \cup \Delta$ equals the set of formulae of $y$. We call $y$ the* virtual successor *of $EX\varphi_i$. For every not blocked $EX\varphi_i$ of $x$, the* successor *of $EX\varphi_i$ is the $i^{\text{th}}$ child of the $EX$-rule.*

Note that a state is just another term for an $EX$-node, whereas a pre-state can be any type of node (it may even be a state).

**Proposition 20 (Termination).** *Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form. Any tableau $T$ for a node $(\{\phi\} :: \cdots :: \cdots)$ is a finite tree, hence the procedure that builds a tableau always terminates.*

*Proof.* It is obvious that $T$ is a tree. Although it is not as trivial as it might seem to be at first sight, it is not too hard to show that every node in $T$ can contain only formulae of the extended closure $\mathrm{ecl}(\phi \wedge EX(p_0 \vee \neg p_0))$. It is obvious that $\mathrm{ecl}(\phi \wedge EX(p_0 \vee \neg p_0))$ is finite. Hence, there are only a finite number of different sets that can be assigned to the nodes, in particular the pre-states. As the $EX$-rule guarantees that all pre-states on a path possess different sets of formulae, there can only be a finite number of pre-states on a path. Furthermore, from any pre-state, there are only a finite number of consecutive nodes on a path until we reach a state. As every state in a path is followed by a pre-state and there are only a finite number of pre-states, all paths in $T$ must be finite. This, the obvious fact that every node in $T$ has finite degree, and König's lemma complete the proof. □

### 3.3 Soundness and Completeness

Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form and $T$ an expanded tableau with root $r = (\{\phi\} :: [] :: \mathrm{mrk}, \mathrm{uev})$: that is, the initial formula set is $\{\phi\}$ and the initial HCr is the empty list.

**Theorem 21.** *If $r$ is not marked, then there exists a Hintikka structure for $\phi$.*

**Theorem 22.** *For every marked node $x = (\Gamma :: \cdots :: \cdots)$ in $T$, the formula $\bigwedge_{\varphi \in \Gamma} \varphi$ is not satisfiable. In particular, if $r$ is marked, then $\phi$ is not satisfiable.*

Detailed proofs can be found in the extended version of this paper (`users.rsise.anu.edu.au/~florian`).

### 3.4 A Fully Worked Example

As an example, consider the formula $E(p_1 \, U \, p_2) \wedge \neg E(\top \, U \, p_2)$ which is obviously not satisfiable. Converting the formula into negation normal form gives us $E(p_1 \, U \, p_2) \wedge A(\bot \, B \, p_2)$. Hence, any expanded tableau with root $E(p_1 \, U \, p_2) \wedge A(\bot \, B \, p_2)$ should be marked.

Figure 1 and Fig. 2 show such a tableau where the root node is node (1) in Fig. 1 and where Fig. 2 shows the sub-tableau rooted at node (5). Each node is classified as a $\rho$-node if rule $\rho$ is applied to that node in the tableau. The unlabelled edges go from states to pre-states. Dotted frames indicate that the sub-tableaux at these nodes are not shown because they are very similar to sub-tableaux of other nodes: that is node (6a) behaves the same way as node (3a). Dots "$\cdots$" indicate that the corresponding values are not important because they are not needed to calculate the value of any other history or variable. The partial function $UEV$ maps the formula $E(p_1 \, U \, p_2)$ to 1 and is undefined otherwise as explained below. The history $HCR$ is defined as $HCR := [\{E(p_1 \, U \, p_2), A(\bot \, B \, p_2)\}]$.

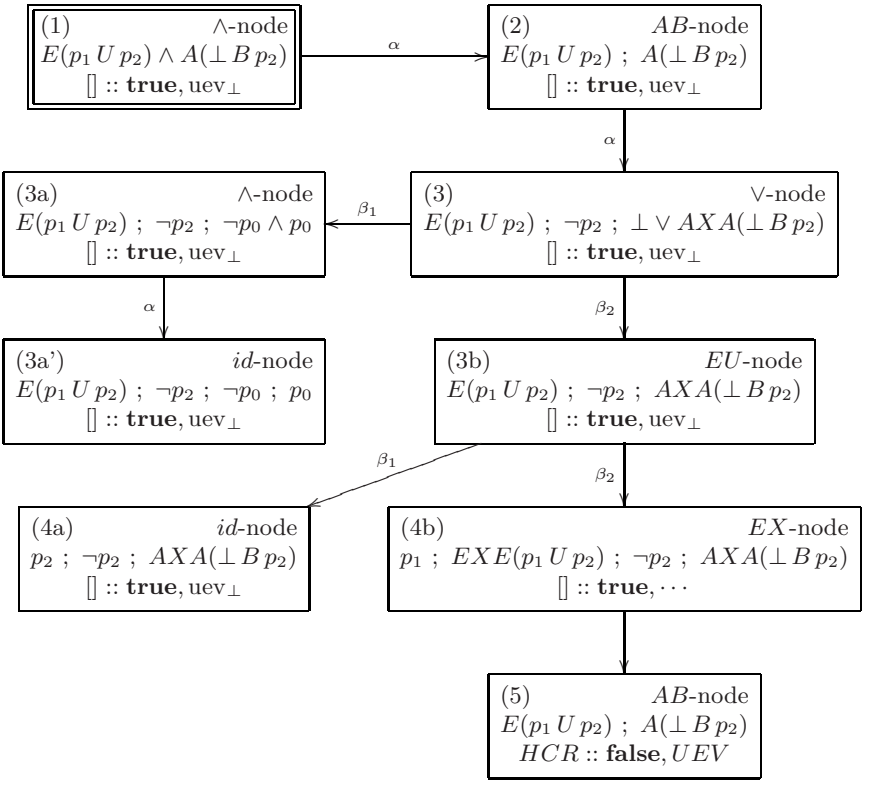


**Fig. 1.** An example: a tableau for $E(p_1 U p_2) \wedge A(\bot B p_2)$

The marking of the nodes (1) to (4a) in Fig. 1 with **true** is straightforward. Note that $\bot$ is just an abbreviation for $\neg p_0 \wedge p_0$ to save some space and make things easier for the reader; the tableau procedure as described in this paper does not know about the symbol $\bot$. It is, however, not a problem to adapt the rules so that the tableau procedure can handle $\top$ and $\bot$ directly. For node (5), our procedure constructs the tableau shown in Fig. 2. The leaf (7b) is an $EX$-node, but it is "blocked" from creating the desired successor containing $\{E(p_1 U p_2), A(\bot B p_2)\}$ because there is a $j \in \mathbb{N}$ such that $\mathrm{HCr}_{7\mathrm{b}}[j] = HCR[j] = \{E(p_1 U p_2), A(\bot B p_2)\}$: namely $j = 1$. Thus the $EX$-rule computes $UEV(E(p_1 U p_2)) = 1$ as stated above and also puts $\mathrm{mrk}_{7\mathrm{b}} :=$ **false**. As the nodes (7a) and (6a) are marked, the function $UEV$ is passed on to the nodes (6b), (6), and (5) according to the corresponding $\beta$- and $\alpha$-rules.

The crux of our procedure happens at node (4b) which is an $EX$-node with $\mathrm{HCr}_{4\mathrm{b}} = []$ and hence $len(\mathrm{HCr}_{4\mathrm{b}}) = 0$. The $EX$-rule therefore finds a child node (5) and a formula $E(p_1 U p_2)$ in it such that $1 = UEV(E(p_1 U p_2)) = \mathrm{uev}_5(E(p_1 U p_2)) > len(\mathrm{HCr}_{4\mathrm{b}}) = 0$. That is, node (4b) "sees" a child (5) that "loops lower", meaning that node (5) is the root of an "isolated" subtree which

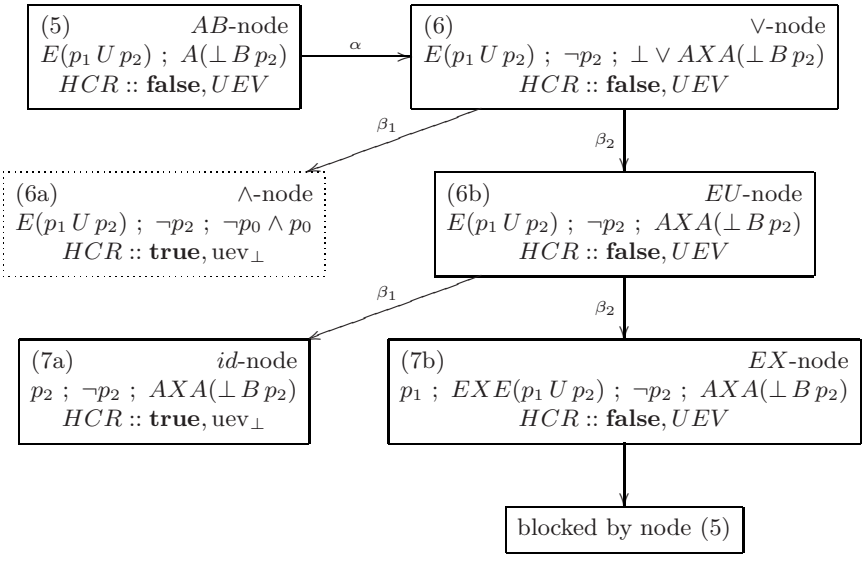


**Fig. 2.** An example: a tableau for $E(p_1\,U\,p_2) \wedge A(\bot\,B\,p_2)$ (continued)

does not fulfil its eventuality $E(p_1\,U\,p_2)$. Thus the $EX$-rule sets $\mathrm{mrk}_{4\mathrm{b}} = \mathbf{true}$, marking (4b) as "closed". The propagation of **true** to the root is then just via simple $\beta$- and $\alpha$-rule applications.

### 3.5 The One-Pass Algorithm and Its Complexity

Most tableau-based algorithms apply the rules in a particular order: namely, apply all the $\alpha$- and $\beta$-rules until none are applicable, and then apply the $EX$-rule once. Of course, no more rules are applied if the $id$-rule is applicable to close the branch. We have designed the rules so that they naturally capture this strategy, thereby giving a non-deterministic algorithm for constructing/traversing the tableau by just applying any one of the rules that are applicable. By fixing an arbitrary rule order and an arbitrary formula order, we can safely determinise this algorithm.

The use of histories and variables gives rise to an algorithm that constructs and traverses a tableau (deterministically) at the same time. On its way down the tableau, it constructs the set of formulae and the histories of a node by using information from the parent node; and on its way up, it synthesises the variables of a node according to the values of the variables of its children. Both steps are described by the rule that is applied to the node.

As soon as the algorithm has left a node on its way up, there is no need to keep the node in memory, it can safely be reclaimed as all important information has been passed up by the variables. Hence, the algorithm requires just one pass. Moreover, at any time, it only has to keep the current branch of the tableau in

memory. The final result of the decision procedure can be obtained by looking at the variable mrk of the root which is the last node that has its variables set.

Of course, it is not always necessary to build the entire tableau. If, for example, the first child of an $EX$-rule is marked, the algorithm can mark the parent without having to look at the other children. (It is easy to see that if a node $x$ is marked then the value of $\text{uev}_x$ is irrelevant and does not need to be calculated.) Dually, if the left child of a $\beta$-node is unmarked and has $\text{uev}_\perp$ then there is no need to explore the right child since we can safely say that the parent is unmarked and has $\text{uev}_\perp$. Other optimisations are possible and some of them are incorporated in our implementation in the Tableau Work Bench (`twb.rsise.anu.edu.au/twbdemo`), a generic tableau engine designed for rapid prototyping of (propositional) tableau calculi [1]. The high-level code of the prover for $CTL$ is also visible there using the special input language designed for the TWB.

**Theorem 23.** *The tableau algorithm for CTL outlined in this paper runs in double exponential deterministic time and needs exponential space.*

A detailed proof can be found in the extended version of this paper (`users.rsise.anu.edu.au/~florian`).

## References


1. Abate, P.: The Tableau Workbench: a framework for building automated tableau-based theorem provers. PhD thesis, The Australian National University (2006)
2. Baader, F., Tobies, S.: The inverse method implements the automata approach for modal satisfiability. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 92–106. Springer, Heidelberg (2001)
3. Baader, F.: Augmenting concept languages by transitive closure of roles: an alternative to terminological cycles. Technical Report, DFKI (1990)
4. Basukoski, A., Bolotov, A.: A clausal resolution method for branching time logic ECTL+. Annals of Mathematics and Artificial Intelligence 46(3), 235–263 (2006)
5. Ben-Ari, M., Manna, Z., Pnueli, A.: The temporal logic of branching time. In: Proceedings of Principles of Programming Languages (1981)
6. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. J. of Computer and System Sci. 30, 1–24 (1985)
7. Fisher, M., Dixon, C., Peim, M.: Clausal temporal resolution. ACM Transactions on Computational Logic (2001)
8. Gaintzarain, J., Hermo, M., Lucio, P., Navarro, M., Orejas, F.: A Cut-Free and Invariant-Free Sequent Calculus for PLTL. In: 6th EACSL Annual Conference on Computer Science and Logic (to appear, 2007)
9. Gough, G.: Decision procedures for temporal logics. Master's thesis, Dept. of Computer Science, University of Manchester, England (1984)
10. Grumberg, O., Clarke, E.M., Peled, D.A.: Model checking. MIT Press, Cambridge (1999)
11. Horrocks, I., Patel-Schneider, P.F.: Optimising description logic subsumption. Journal of Logic and Computation 9(3), 267–293 (1999)

12. Horrocks, I., Sattler, U.: A tableaux decision procedure for SHOIQ. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, pp. 448–453 (2005)
13. Hustadt, U., Konev, B.: TRP++: A temporal resolution prover. In: Collegium Logicum, Kurt Gödel Society, pp. 65–79 (2004)
14. Jäger, G., Alberucci, L.: About cut elimination for logics of common knowledge. Ann. Pure Appl. Logic 133(1-3), 73–99 (2005)
15. Jäger, G., Kretz, M., Studer, T.: Cut-free common knowledge. Journal of Applied Logic (to appear)
16. Janssen, G.: Logics for Digital Circuit Verification: Theory, Algorithms, and Applications. PhD thesis, Eindhoven University of Technology, The Netherlands (1999)
17. Kesten, Y., Manna, Z., McGuire, H., Pnueli, A.: A decision algorithm for full propositional temporal logic. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 97–109. Springer, Heidelberg (1993)
18. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer, Heidelberg (1992)
19. Reynolds, M.: Towards a CTL* tableau. In: Ramanujam, R., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 384–395. Springer, Heidelberg (2005)
20. Schwendimann, S.: A new one-pass tableau calculus for PLTL. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 277–291. Springer, Heidelberg (1998)
21. Vardi, M.Y., Pan, G., Sattler, U.: BDD-based decision procedures for K. Journal of Applied Non-Classical Logics 16(1-2), 169–208 (2006)
22. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer Systems and Science (1986)
23. Voronkov, A.: How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi. ACM Trans. on Comp. Logic 2(2) (2001)
24. Wolper, P.: Temporal logic can be more expressive. Inf. and Cont. 56, 72–99 (1983)