LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Local Model Checking in the Modal Mu-Calculus

Local Model Checking

in the Modal Mu-Calculus

by

Colin Stirling and David Walker

ECS-LFCS-89-78
(also published as CSR-296-89)

LFCS Report Series

May 1989

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Local Model Checking in the Modal Mu-Calculus

Colin Stirling and David Walker
Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ, U.K.

1 Motivation

The modal mu-calculus, due to Pratt and Kozen [Pr, Ko], is a natural extension of dynamic logic. It is also one method of obtaining a branching time temporal logic from a modal logic [EL]. Furthermore, it extends Hennessy-Milner logic, thereby offering a natural temporal logic for Milner's CCS, and process systems in general. (Discussion of the uses of the mu-calculus for CCS can be found in [GS,Ho,La,St,Sti2].) Within this context we are especially interested in whether or not a particular state, or process, in a finite model satisfies a mu-calculus formula. This is a different enterprise from that addressed by Emerson and Lei [EL] who ask if a given formula is satisfiable in a given finite model. Their model checker appeals to standard approximation techniques for computing the set of states which satisfy a fixpoint formula. But then one has to compute *all* the states or processes in the model which satisfy that formula.

In this paper we present a local model checker for the mu-calculus, as a tableau system. It checks whether or not a particular state satisfies a formula. Instead of using approximation techniques there is an implicit use of fixpoint induction (inspired by [La]). A maximal fixpoint formula, in effect, expresses a safety property. One shows that the assumption that a state has such a property leads to no unforeseen consequences. In contrast, a minimal fixpoint formula expresses a liveness property. Therefore one has to establish that the property holds of a particular state. Formulae involving alternating fixpoints [EL] introduce subtleties. However the resulting tableau system is natural and an equivalent version of it has been implemented by Rance Cleaveland [Cl].

In section 2 we describe the syntax and semantics of the modal mu-calculus. A small extension to the calculus, the addition of propositional constants, is detailed in section 3. The model checker, presented as a tableau system, is given in section 4, while the proofs of its soundness, completeness and decidability are the topic of section 6. Finally, in section 5 we use the model checker to analyse a mutual exclusion algorithm when translated into CCS.

2 The modal mu-calculus

The set of formulae of the modal mu-calculus is defined by:

$$A ::= Z \mid Q \mid \neg A \mid A \wedge A \mid [a]A \mid \nu Z.A$$

where Z ranges over propositional variables, Q over atomic propositions, and a over a set of (action) labels. One restriction on $\nu Z.A$ is that each free occurrence of Z in A lies within the scope of an even number of negations. Derived operators are defined in the familiar way: $A \vee B$ is $\neg(\neg A \wedge \neg B)$; $\langle a\rangle A$ is $\neg[a]\neg A$; and $\mu Z.A$ is $\neg\nu Z.\neg A[Z := \neg Z]$, where $A[Z := \neg Z]$ is the result of substituting $\neg Z$ for each free occurrence of Z in A.

The mu-calculus, with action labels drawn from a set *Act*, is interpreted on labelled transition systems $\mathcal{T}$ which are pairs of the form $\mathcal{T} = (S, \{\xrightarrow{a} \mid a \in Act\})$. S (or $S_{\mathcal{T}}$) is a nonempty set of states, and for each $a \in Act$, $\xrightarrow{a}$ is a transition relation on states. We write $s \xrightarrow{a} s'$ instead of $(s, s') \in \xrightarrow{a}$. Labelled transition systems are popular structures for modelling concurrent systems, [Mi, Pn], including process algebras such as CCS. S is then a set (or algebra) of processes and $s \xrightarrow{a} s'$ means that process s may become s' by preforming the action a. In this context the mu-calculus can be viewed as a branching time temporal logic for CCS, a natural extension of the modal logic in [HM].

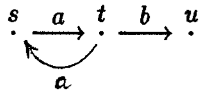
A model $\mathcal{M}$ for the mu-calculus is a pair $\mathcal{M} = (\mathcal{T}, V)$ where $\mathcal{T}$ (or $\mathcal{T}_{\mathcal{M}}$) is a transition system and V (or $V_{\mathcal{M}}$) is a valuation assigning sets of states to atomic propositions and variables: $V(Q) \subseteq S_{\mathcal{T}}$ and $V(Z) \subseteq S_{\mathcal{T}}$. We assume the customary updating notation: $V[S'/Z]$ is the valuation V' which agrees with V except that $V'(Z) = S'$. Finally the set of states satisfying A in a model $\mathcal{M} = (\mathcal{T}, V)$ is inductively defined as $\| A \|_V^{\mathcal{T}}$ (where for ease of notation we drop the index $\mathcal{T}$ which is assumed to be fixed):

$$\begin{aligned}
\| Z \|_V &= V(Z) \\
\| Q \|_V &= V(Q) \\
\| \neg A \|_V &= S_{\mathcal{T}} - \| A \|_V \\
\| A \wedge B \|_V &= \| A \|_V \cap \| B \|_V \\
\| [a] A \|_V &= \{ s \in S_{\mathcal{T}} \mid \forall s'.\ \text{if } s \xrightarrow{a} s' \text{ then } s' \in \| A \|_V \} \\
\| \nu Z. A \|_V &= \bigcup \{ S' \subseteq S_{\mathcal{T}} \mid S' \subseteq \| A \|_{V[S'/Z]} \}
\end{aligned}$$

The expected clause for the derived operator $\mu Z.$ is:

$$\| \mu Z. A \|_V = \bigcap \{ S' \subseteq S_{\mathcal{T}} \mid \| A \|_{V[S'/Z]} \subseteq S' \}$$

A simple example is the model $\mathcal{M} = (\mathcal{T}, V)$ where $\mathcal{T}$ is



and $V(Q) = \emptyset$ for all atomic Q. Let R be the formula $\langle b \rangle \mathtt{true}$. Let A and B be the formulae

$$\begin{aligned}
A &\equiv \nu Z. \mu Y. \langle a \rangle ((R \wedge Z) \vee Y) \\
B &\equiv \mu Y. \nu Z. \langle a \rangle ((R \vee Y) \wedge Z)
\end{aligned}$$

Now

$$\begin{aligned}
\| A \|_V^{\mathcal{T}} &= \{s, t\} \\
\| B \|_V^{\mathcal{T}} &= \emptyset
\end{aligned}$$

The formula A expresses that on some a^ω path R holds infinitely often, while B expresses that on some a^ω path R holds almost always. In CCS, where states are processes, u represents the process $\mathbf{0}$ (*Nil*) which can preform no actions, while s and t are the processes

$$\begin{aligned}
s &= \text{fix} Z.\, a.\, (b.\mathbf{0} + a.\, Z) \\
t &= \text{fix} Z.\, b.\, \mathbf{0} + a.\, a.\, Z
\end{aligned}$$

Hence both processes s and t have the property expressed by A.

A model is *finite* if its set of states is finite. Our interest is in the particular question: does state, or process, s have the property expressed by the formula A in the finite model $\mathcal{M} = (\mathcal{T}, V)$, i.e. is $s \in \| A \|_V^{\mathcal{T}}$? A natural technique is to compute the set $\| A \|_V$, [EL], using approximation techniques when A contains fixpoint subformulae. For instance, using semantic approximants, if V is a valuation let $V_0 = V[S_{\mathcal{T}}/Z]$ and $V_{i+1} = V_i[\| A \|_{V_i} /Z]$. Then because the model is finite we know that

$$\| \nu Z.\, A \|_V = \bigcap_{i \geq 0} V_i(Z)$$

Also by finiteness we know that there is $i \geq 0$ such that $V_i(Z) = V_{i+1}(Z)$, and for such an i, $V_i(Z) = \| \nu Z.\, A \|_V$. Finally one just needs to check whether or not the required state s is in this set. (A minimal fixpoint formula $\neg \nu Z.\, A$ can be dealt with by computing either $S_{\mathcal{T}} - \| \nu Z.\, A \|_V$ or $\bigcup_{i \geq 0} V_i(Z)$ where $V_0 = V[\emptyset/Z]$ and $V_{i+1} = V_i[\| \neg A[Z := \neg Z] \|_{V_i} /Z]$.) But this technique is not intended to be sensitive to the fact that we are interested only in whether or not the particular state s lies in $\| A \|_V$.

An apparent localisation is to appeal, instead, to syntactic approximants. Let $(\nu Z.\, A)^0 = \mathtt{true}$ and $(\nu Z.\, A)^{i+1} = A[Z := (\nu Z.\, A)^i]$. Then again because of finiteness we know that

$$s \in \| \nu Z.\, A \|_V \text{ iff } \forall i \geq 0.\, s \in \| (\nu Z.\, A)^i \|_V$$

But again it is necessary to compute the complete fixpoint set, i.e. the set $S' = \| (\nu Z.\, A)^i \|_V$ where $\| (\nu Z.\, A)^i \|_V = \| (\nu Z.\, A)^{i+1} \|_V$. For there is no guarantee that if for some j, $s \in \| (\nu Z.\, A)^j \|_V \cap \| (\nu Z.\, A)^{j+1} \|_V$ then also $s \in \| \nu Z.\, A \|_V$.

An alternative, more local, approach to model checking (which does not depend on computing complete fixpoint sets) is to appeal to fixpoint induction. The idea is that $s \in \| \nu Z.\, A \|_V$ if the assumption that $s \in \| \nu Z.\, A \|_V$ *implies* $s \in \| A[Z := \nu Z.\, A] \|_V$; and in the case of a minimal fixpoint formula, $s \in \| \mu Y.\, A \|_V$ if the assumption that $s \notin \| \mu Y.\, A \|_V$ implies $s \in \| A[Y := \mu Y.\, A] \|_V$. This technique is used by Larsen [La] for a logic which disallows alternating fixpoints: each formula contains only maximal fixpoints or only minimal fixpoints. The major problem here, especially in the presence of formulae containing alternating fixpoints, is that of logically understanding assumptions of the form $s \in \| \nu Z.\, A \|_V$ and $s \notin \| \mu Y.\, A \|_V$ as well as the notion of implication. The simple local tableau technique which we offer below not only caters for the full modal mu-calculus but also has a natural logical interpretation. There is, however, a small cost: a need to extend the mu-calculus to include propositional constants and definition lists.

3 Adding constants and definition lists

The syntax of the mu-calculus is extended to embrace a family of propositional constant symbols. Associated with a constant U is a declaration of the form $U = A$ where A is a closed formula, possibly containing previously declared constant symbols. A *definition list* is a sequence Δ of declarations $U_1 = A_1, \ldots, U_n = A_n$ such that $U_i \neq U_j$ whenever $i \neq j$ and such that each constant occurring in A_i is one of $U_1, \ldots, U_{i-1}$. This means that a prefix of a definition list is itself a definition list. When Δ as above is such a list we let $dom(\Delta) = \{U_1, \ldots, U_n\}$ and $\Delta(U_i) = A_i$. Moreover, if Δ is a definition list, $U \notin dom(\Delta)$ and each constant occurring in A is in $dom(\Delta)$, then $\Delta \cdot U = A$ is the definition list which is the result of appending $U = A$ to Δ. A definition list Δ is *admissible for* B if every constant occurring in B is declared in Δ. In this circumstance we let B_Δ be the formula B in the 'environment' Δ (see Definition 1). The interpretation of formulae is now extended to formulae relative to admissible definition lists by, in effect, treating constants as variables.

Definition 1 If $\Delta : U_1 = A_1, \ldots, U_n = A_n$ is admissible for B then $\| B_\Delta \|_V =_{df} \| B \|_{V_n}$ where $V_0 = V$ and $V_{i+1} = V_i[\| A_{i+1} \|_{V_i} / U_{i+1}]$.

This interpretation accords with the expected meaning of B_Δ in terms of syntactic substitution.

Lemma 2 $\| B_{\Delta \cdot U = A} \|_V = \| (B[U := A])_\Delta \|_V$.

Proof: By induction on the structure of B. □

A corollary, invoked later, is that if U does not occur in B then $B_{\Delta \cdot U = A}$ has the same meaning as B_Δ.

4 The model checker

The model checker is a tableau system for testing whether or not a state s has the property expressed by a closed formula A in a finite model $\mathcal{M}$. As is common in tableau systems, the rules are inverse natural deduction type rules. Here they are built from 'sequents' of the form $s \vdash_\Delta^{\mathcal{M}} A$, proof-theoretic analogues of $s \in \| A_\Delta \|_V^{\mathcal{T}}$. Each rule is of the form

$$\frac{s \vdash_\Delta^{\mathcal{M}} A}{s_1 \vdash_{\Delta_1}^{\mathcal{M}} A_1 \ldots s_k \vdash_{\Delta_k}^{\mathcal{M}} A_k}$$

where $k > 0$, possibly with side conditions. The premise sequent $s \vdash_\Delta^{\mathcal{M}} A$ is the goal to be achieved while the consequents are the subgoals, which are determined by the structure of the model 'near s,' the definition list Δ and the structure of A. Often, in the sequel, the index $\mathcal{M}$ is dropped from the sequents. The intermediate use of definition lists is essential, as they keep track of the 'dynamically changing' subformulae as fixpoints are unrolled. This is the key to the technique. Condition $\mathcal{C}$, the side-condition on the constant rules, is explained later as it is a condition on proof trees, rather than on the particular sequents of the premises.

$$\frac{s \vdash_\Delta \neg\neg A}{s \vdash_\Delta A} \qquad \frac{s \vdash_\Delta A \wedge B}{s \vdash_\Delta A \quad s \vdash_\Delta B}$$

$$\frac{s \vdash_\Delta \neg(A \wedge B)}{s \vdash_\Delta \neg A} \qquad \frac{s \vdash_\Delta \neg(A \wedge B)}{s \vdash_\Delta \neg B}$$

$$\frac{s \vdash_\Delta [a]A}{s_1 \vdash_\Delta A \ldots s_n \vdash_\Delta A} \quad \{s_1, \ldots, s_n\} = \{s' \mid s \xrightarrow{a} s'\}$$

$$\frac{s \vdash_\Delta \neg[a]A}{s' \vdash_\Delta \neg A} \quad s \xrightarrow{a} s'$$

$$\frac{s \vdash_\Delta \nu Z. A}{s \vdash_{\Delta'} U} \quad \Delta' \text{ is } \Delta \cdot U = \nu Z. A$$

$$\frac{s \vdash_\Delta \neg\nu Z. A}{s \vdash_{\Delta'} U} \quad \Delta' \text{ is } \Delta \cdot U = \neg\nu Z. A$$

$$\frac{s \vdash_\Delta U}{s \vdash_\Delta A[Z := U]} \quad \mathcal{C} \text{ and } \Delta(U) = \nu Z. A$$

$$\frac{s \vdash_{\Delta} U}{s \vdash_{\Delta} \neg A[Z := \neg U]} \quad C \text{ and } \Delta(U) = \neg \nu Z.\, A$$

A *tableau* for $s \vdash^{\mathcal{M}} A$ is a maximal proof tree whose root is labelled with the sequent $s \vdash^{\mathcal{M}} A$ (where we omit the definition list when, as here, it is empty). The sequents labelling the immediate successors of a node labelled $s \vdash^{\mathcal{M}}_{\Delta} A$ are determined by an application of one of the rules, dependent on the structure of A. For simplicity we have allowed non-determinism in the result sequents in the cases of $\neg(A \wedge B)$ and $\neg[a]A$, rather than entangling proof trees with or-branching as well as and-branching. Maximality means that no rule applies to a sequent labelling a leaf of a tableau. The rules for booleans and modal operators are straightforward. New constants are introduced in the case of fixpoint formulae, while the rules for constants unroll the fixpoints they abbreviate when condition C holds. This condition is just that no node above the current premise, $s \vdash^{\mathcal{M}}_{\Delta} U$, in the proof tree is labelled $s \vdash^{\mathcal{M}}_{\Delta'} U$ for some Δ'. So failure of the condition, when there is a sequent $s \vdash^{\mathcal{M}}_{\Delta'} U$ above $s \vdash^{\mathcal{M}}_{\Delta} U$, enforces termination. In fact the presence of condition C guarantees that when $\mathcal{M}$ is finite any tableau for $s \vdash^{\mathcal{M}} A$ is of finite depth. Notice that all the rules are backwards sound. For example, in the case of the rule for maximal fixpoints, if Δ' is $\Delta \cdot U = \nu Z.\, A$ and $s \in \| U_{\Delta'} \|_V$, then by Lemma 2, $s \in \| \nu Z.\, A_{\Delta} \|_V$. Hence if the leaves of a (finite) tableau are *true*, i.e. if whenever $s \vdash_{\Delta} A$ labels a leaf, $s \in \| A_{\Delta} \|_V$, then so is the root.

A *successful* tableau for $s \vdash^{\mathcal{M}} A$ is a finite tableau in which every leaf is labelled by a sequent $t \vdash^{\mathcal{M}}_{\Delta} B$ fulfilling one of the following requirements:

(i)	$B = Q$ and $t \in V_{\mathcal{M}}(Q)$
(ii)	$B = \neg Q$ and $t \notin V_{\mathcal{M}}(Q)$
(iii)	$B = [a]C$
(iv)	$B = U$ and $\Delta(U) = \nu Z.C$

A successful tableau contains only true leaves. This is clear for leaves fulfilling (i) and (ii). Maximality of a tableau guarantees it for leaves satisfying (iii), because then $\{t' \mid t \xrightarrow{a} t'\} = \emptyset$. Of more interest is (iv): if $t \vdash^{\mathcal{M}}_{\Delta'} U$ labels a node in a tableau above a node labelled $t \vdash^{\mathcal{M}}_{\Delta} U$ where $\Delta(U) = \nu Z.\, A$, then indeed $t \in \| U_{\Delta} \|_{V_{\mathcal{M}}}$ (provided that the other leaves beneath $t \vdash^{\mathcal{M}}_{\Delta'} U$ are also true). An unsuccessful tableau has at least one false leaf, such as a leaf labelled $t \vdash^{\mathcal{M}}_{\Delta} Q$ where $t \notin V_{\mathcal{M}}(Q)$. Again, the most interesting failure is when a leaf is labelled $t \vdash^{\mathcal{M}}_{\Delta} U$ where $\Delta(U) = \neg \nu Z.\, A$ and above it is a node labelled $t \vdash^{\mathcal{M}}_{\Delta'} U$.

Tableau rules for the derived operators are just reformulations of some of the negation rules:

$$\frac{s \vdash_{\Delta} A \vee B}{s \vdash_{\Delta} A} \qquad \frac{s \vdash_{\Delta} A \vee B}{s \vdash_{\Delta} B}$$

$$\frac{s \vdash_{\Delta} \langle a \rangle A}{s' \vdash_{\Delta} A} \quad s \xrightarrow{a} s'$$

$$\frac{s \vdash_{\Delta} \mu Z.\, A}{s \vdash_{\Delta'} U} \quad \Delta' \text{ is } \Delta \cdot U = \mu Z.\, A$$

$$\frac{s \vdash_{\Delta} U}{s \vdash_{\Delta} A[Z := U]} \quad C \text{ and } \Delta(U) = \mu Z.\, A$$

If these operators were also taken as primitive (as in the case of normal forms) then the definition of successful tableau would be changed accordingly.

The two important theorems follow. Their proofs are given in section 6 below. For both we assume that $\mathcal{M}$ is finite. Theorem 4 affirms soundness and completeness, while Theorem 3 amounts to decidability (since there can be only a finite number of tableaux for $s \vdash^{\mathcal{M}} A$, up to renaming of constants).

Theorem 3 Every tableau for $s \vdash^{\mathcal{M}} A$ is finite.

Theorem 4 $s \vdash^{\mathcal{M}} A$ has a successful tableau if and only if $s \in \| A \|_{V_{\mathcal{M}}}$.

By employing more complex sequents the side condition C on the two constant rules can be replaced with a condition on sequents. Let an *extended sequent* have the form

$$\alpha \longrightarrow s \vdash_{\Delta} A$$

where α is a finite set of sequents, each of which is of the form $t \vdash_{\Delta'} U$: the idea is that α contains all sequents above $s \vdash_{\Delta} A$ whose formula is a constant. The rules earlier can be trivially expanded to extended sequents. Two sample examples are:

$$\frac{\alpha \longrightarrow s \vdash_{\Delta} A \wedge B}{\alpha \longrightarrow s \vdash_{\Delta} A \qquad \alpha \longrightarrow s \vdash_{\Delta} B}$$

$$\frac{\alpha \longrightarrow s \vdash_{\Delta} U}{\alpha, s \vdash_{\Delta} U \longrightarrow s \vdash_{\Delta} A[Z := U]} \quad s \vdash_{\Delta'} U \notin \alpha \text{ for any } \Delta' \text{ and } \Delta(U) = \nu Z.\, A$$

Now the side condition C is replaced by: $s \vdash_{\Delta'} U \notin \alpha$ for any Δ'. This simple reformulation of the rules is akin to the formalisation of sequent calculi from natural deduction systems. Recently Winskel [Wi] has discovered an alternative formulation of the tableau system which allows a clear semantic account to be given. We give a brief description of it and, by reformulating it using constants and definitions lists, show the equivalence of the two approaches.

Rather than extending the language with constants, given a model $\mathcal{M} = (\mathcal{T}, V)$ with $\mathcal{T} = (S, \{\xrightarrow{a} \mid a \in Act\})$, Winskel introduces a family of operators $\nu Z\{\vec{s}\}.$, one for each finite subset $\{\vec{s}\}$ of S. The interpretation is as follows.

$$\| \nu Z\{\vec{s}\}.\, A \|_V = \bigcup \{S' \subseteq S \mid S' \subseteq \{\vec{s}\} \cup \| A \|_{V[S'/Z]}\}$$

The crucial property of this family of operators is the following.

Fact 5 [Wi] Suppose that no variable other than Z occurs free in A and that $t \notin \{\vec{s}\}$. Then

$$t \in \| \nu Z\{\vec{s}\}.\, A \|_V \text{ iff } t \in \| A[Z := \nu Z\{\vec{s}\} \cup \{t\}.\, A] \|_V$$

Winskel gives a set of reduction rules for determining whether or not a state s satisfies a closed formula A. We reformulate these rules as a tableau system using constants.

Fix a model $\mathcal{M}$ as above. We modify the notion of definition list introduced in section 3 as follows. A *definition list* is of the form

$$\Delta = \langle U_1 = (A_1, J_1), \ldots, U_n = (A_n, J_n) \rangle$$

where $U_i \neq U_j$ if $i \neq j$, each A_i is a closed formula of the form $\nu Z.B$ or $\neg\nu Z.B$ which may contain $U_1, \ldots, U_{i-1}$, and each $J_i \subseteq S$. As before Δ is admissible for B if every constant occurring in B is declared in Δ. If Δ is admissible for B then

$$\| B_\Delta \|_V = \| B \|_{V_n}$$

where $V_0 = V$ and for $i < n$, $V_{i+1} = V_i[\| (A_{i+1}, J_{i+1}) \|_{V_i} / U_{i+1}]$ where

$$\begin{array}{rcl} \|(\nu Z.C, J)\|_V & = & \bigcup\{S' \mid S' \subseteq J \cup \|C\|_{V[S'/Z]}\} \\ \|(\neg\nu Z.C, J)\|_V & = & S - \|(\nu Z.C, J)\|_V \end{array}$$

The modified tableau system has the same rules for boolean and modal operators as the original system. In place of the constant introduction and unfolding rules it has the following rules.

$$\frac{s \vdash_\Delta \nu Z.A}{s \vdash_{\Delta'} U} \quad \Delta' \text{ is } \Delta \cdot U = (\nu Z.A, \emptyset)$$

$$\frac{s \vdash_\Delta \neg\nu Z.A}{s \vdash_{\Delta'} U} \quad \Delta' \text{ is } \Delta \cdot U = (\neg\nu Z.A, \emptyset)$$

$$\frac{s \vdash_\Delta U}{s \vdash_{\Delta'} A[Z := U]} \quad \Delta(U)_1 = \nu Z.A,\ s \notin \Delta(U)_2$$

$$\frac{s \vdash_\Delta U}{s \vdash_{\Delta'} \neg A[Z := \neg U]} \quad \Delta(U)_1 = \neg\nu Z.A,\ s \notin \Delta(U)_2$$

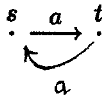
where in the last two rules Δ' is $\Delta[(\Delta(U)_1, \Delta(U)_2 \cup \{s\})/U]$.

The analogues of Theorems 3 and 4 above hold for this modified system. The proof of Theorem 3 goes over unchanged, while the crucial observation in the proof of Theorem 4 is the Fact above which shows that the two constant unfolding rules are both forwards and backwards sound. Completeness of the tableau system then follows by an easy simplification of the proof for the original system given in section 6, while soundness follows immediately from the fact that if $s \vdash_\Delta U$ is a leaf and $\Delta(U) = (\nu Z.A, J)$, then $s \in J$ and so $s \in \| U_\Delta \|_V$.

Cleaveland [Cl] has given another formulation of the tableau system which dispenses with the use of constants but at the cost of a complex subformula test. His proofs also rely on an observation similar to Fact 5 above.

5 Applications

We begin with two examples to illustrate the tableau method. Suppose $\mathcal{M} = (\mathcal{T}, V)$ is the model where $\mathcal{T}$ may be pictured as



and $V(Q) = \{t\}$. Consider the formulae

$$\begin{array}{rcl} A & \equiv & \nu Z.\mu Y.[a]((Q \wedge Z) \vee Y) \\ B & \equiv & \mu Y.\nu Z.[a]((Q \vee Y) \wedge Z) \end{array}$$

which in $\mathcal{M}$ express, respectively, that on all paths Q holds infinitely often, and that on all paths Q holds almost always. We present a successful tableau for $s \vdash^{\mathcal{M}} A$ and show that every tableau for $t \vdash^{\mathcal{M}} B$ is unsuccessful.

In the following successful tableau for $s \vdash^{\mathcal{M}} A$,

$$\begin{array}{rcl} \Delta_1 & = & (U_1 = A) \\ \Delta_2 & = & \Delta_1 \cdot (U_2 = A_1) \\ \Delta_3 & = & \Delta_2 \cdot (U_3 = A_1) \end{array}$$

where $A_1 = \mu Y. [a]((Q \wedge U_1) \vee Y)$.

$$\begin{array}{c}
s \vdash A \\ \hline
s \vdash_{\Delta_1} U_1 \\ \hline
s \vdash_{\Delta_1} A_1 \\ \hline
s \vdash_{\Delta_2} U_2 \\ \hline
s \vdash_{\Delta_2} [a]((Q \wedge U_1) \vee U_2) \\ \hline
t \vdash_{\Delta_2} (Q \wedge U_1) \vee U_2 \\ \hline
t \vdash_{\Delta_2} Q \wedge U_1 \\ \hline
\begin{array}{cc}
t \vdash_{\Delta_2} Q &
\begin{array}{c}
t \vdash_{\Delta_2} U_1 \\ \hline
t \vdash_{\Delta_2} A_1 \\ \hline
t \vdash_{\Delta_3} U_3 \\ \hline
t \vdash_{\Delta_3} [a]((Q \wedge U_1) \vee U_3 \\ \hline
s \vdash_{\Delta_3} (Q \wedge U_1) \vee U_3 \\ \hline
s \vdash_{\Delta_3} U_3 \\ \hline
s \vdash_{\Delta_3} [a]((Q \wedge U_1) \vee U_3) \\ \hline
t \vdash_{\Delta_3} (Q \wedge U_1) \vee U_3 \\ \hline
t \vdash_{\Delta_3} Q \wedge U_1 \\ \hline
\begin{array}{cc}
t \vdash_{\Delta_3} Q & t \vdash_{\Delta_3} U_1
\end{array}
\end{array}
\end{array}
\end{array}$$

In the following unsuccessful tableau for $s \vdash^{\mathcal{M}} B$,

$$\begin{array}{rcl} \Delta_1 & = & (U_1 = B) \\ \Delta_2 & = & \Delta_1 \cdot (U_2 = B_1) \\ \Delta_3 & = & \Delta_2 \cdot (U_3 = B_1) \end{array}$$

where $B_1 = \nu Z.\, [a]((Q \lor U_1) \land Z)$.

$$
\begin{array}{c}
t \vdash B \\ \hline
t \vdash_{\Delta_1} U_1 \\ \hline
t \vdash_{\Delta_1} B_1 \\ \hline
t \vdash_{\Delta_2} U_2 \\ \hline
t \vdash_{\Delta_2} [a]((Q \lor U_1) \land U_2) \\ \hline
s \vdash_{\Delta_2} (Q \lor U_1) \land U_2 \\ \hline
\begin{array}{cc}
\begin{array}{c}
s \vdash_{\Delta_2} Q \lor U_1 \\ \hline
s \vdash_{\Delta_2} U_1 \\ \hline
s \vdash_{\Delta_2} B_1 \\ \hline
s \vdash_{\Delta_3} U_3 \\ \hline
s \vdash_{\Delta_3} [a]((Q \lor U_1) \land U_3) \\ \hline
t \vdash_{\Delta_3} (Q \lor U_1) \land U_3 \\ \hline
\begin{array}{cc}
\begin{array}{c}
t \vdash_{\Delta_3} Q \lor U_1 \\ \hline
t \vdash_{\Delta_3} Q
\end{array}
&
\begin{array}{c}
t \vdash_{\Delta_3} U_3 \\ \hline
t \vdash_{\Delta_3} [a]((Q \lor U_1) \land U_3) \\ \hline
s \vdash_{\Delta_3} (Q \lor U_1) \land U_3 \\ \hline
\begin{array}{cc}
\begin{array}{c}
s \vdash_{\Delta_3} Q \lor U_1 \\ \hline
s \vdash_{\Delta_3} U_1
\end{array}
&
s \vdash_{\Delta_3} U_3
\end{array}
\end{array}
\end{array}
\end{array}
&
\begin{array}{c}
s \vdash_{\Delta_2} U_2 \\
\vdots
\end{array}
\end{array}
\end{array}
$$

An important area of application of the model checker is to Milner's CCS [Mi]. An equivalent version of the checker has been implemented by Rance Cleaveland [Cl] in the Concurrency Workbench (a joint UK SERC venture between Sussex and Edinburgh Universities [CPS]). The operational semantics of CCS is given in terms of labelled transition systems. However, there is more than one transition system associated with CCS according to whether or not the τ action is observable. This distinction is marked by the differing transition relations $\xrightarrow{a}$ and $\xRightarrow{a}$ for $a \in \mathit{Act}$. In fact, the action sets differ too: there is the relation $\xrightarrow{\tau}$ but not $\xRightarrow{\tau}$; and there is the relation $\xRightarrow{\varepsilon}$, meaning zero or more silent moves, but not $\xrightarrow{\varepsilon}$. Thus, there are two different Hennessy-Milner logics for CCS [HM], each characterising the appropriate (strong or weak) bisimulation equivalence. Their extension to include fixpoints preserves this characterisation [Sti2]. These are sublanguages of the modal mu-calculus—for their sole atomic sentence is the constant `true`.

We now offer a more substantial example: an analysis of Knuth's mutual exclusion algorithm [Kn] when translated into CCS. Knuth's algorithm is given by the concurrent composition of the two programs when $i = 1$ and $i = 2$, and where j is the index of the other program:

```
while true do
begin
        ⟨ noncritical section ⟩ ;
        $L_0$: $c_i := 1$ ;
        $L_1$: if $k = i$ then goto $L_2$ ;
        if $c_j \neq 0$ then goto $L_1$ ;
        $L_2$: $c_i := 2$ ;
        if $c_j = 2$ then goto $L_0$ ;
        $k := i$ ;
        ⟨ critical section ⟩ ;
        $k := j$ ;
        $c_i := 0$ ;
end ;
```

The variable c_1 (c_2) of program one (two) may take the values $0, 1$ or 2; initially its value is 0. When translated into CCS [Mi,Wa], the algorithm, assuming the initial value of k to be 1, becomes the agent $Knuth$ below. For the example we let capital letters range over CCS processes (states of the CCS transition system). Here we are assuming that τ is not observable (so the transition relations are of the form $\overset{a}{\Longrightarrow}$). Each program variable is represented a family of agents. Thus the variable k with current value 1 is represented as an agent $K1$ which may perform actions corresponding to the reading of the value 1 and the writing of the values 1 and 2 by the two programs. The agents are:

$$Knuth =_{df} (P_1 \mid P_2 \mid K1 \mid C_1 0 \mid C_2 0)\backslash L$$

where L is the union of the sorts of the variables and

$$\begin{aligned}
K1 &=_{df} kw1.\,K1 + kw2.\,K2 + \overline{kr1}.\,K1 \\
K2 &=_{df} kw1.\,K1 + kw2.\,K2 + \overline{kr2}.\,K2
\end{aligned}$$

$$\begin{aligned}
C_1 0 &=_{df} c_1 w0.\,C_1 0 + c_1 w1.\,C_1 1 + c_1 w2.\,C_1 2 + \overline{c_1 r0}.\,C_1 0 \\
C_1 1 &=_{df} c_1 w0.\,C_1 0 + c_1 w1.\,C_1 1 + c_1 w2.\,C_1 2 + \overline{c_1 r1}.\,C_1 1 \\
C_1 2 &=_{df} c_1 w0.\,C_1 0 + c_1 w1.\,C_1 1 + c_1 w2.\,C_1 2 + \overline{c_1 r2}.\,C_1 2
\end{aligned}$$

$$\begin{aligned}
C_2 0 &=_{df} c_2 w0.\,C_2 0 + c_2 w1.\,C_2 1 + c_2 w2.\,C_2 2 + \overline{c_2 r0}.\,C_2 0 \\
C_2 1 &=_{df} c_2 w0.\,C_2 0 + c_2 w1.\,C_2 1 + c_2 w2.\,C_2 2 + \overline{c_2 r1}.\,C_2 1 \\
C_2 2 &=_{df} c_2 w0.\,C_2 0 + c_2 w1.\,C_2 1 + c_2 w2.\,C_2 2 + \overline{c_2 r2}.\,C_2 2
\end{aligned}$$

$$\begin{aligned}
P_1 &=_{df} \tau.\,P_{11} + \tau.\,\mathbf{0} \\
P_{11} &=_{df} \overline{c_1 w1}.\,req_1.\,P_{12} \\
P_{12} &=_{df} kr1.\,P_{14} + kr2.\,P_{13} \\
P_{13} &=_{df} c_2 r0.\,P_{14} + c_2 r1.\,P_{12} + c_2 r2.\,P_{12} \\
P_{14} &=_{df} \overline{c_1 w2}.\,P_{15} \\
P_{15} &=_{df} c_2 r0.\,P_{16} + c_2 r1.\,P_{16} + c_2 r2.\,P_{17} \\
P_{16} &=_{df} \overline{kw1}.\,enter_1.\,exit_1.\,\overline{kw2}.\,\overline{c_1 w0}.\,P_1 \\
P_{17} &=_{df} \overline{c_1 w1}.\,P_{12}
\end{aligned}$$

$$\begin{array}{lll}
P_2 & =_{df} & \tau.\, P_{21} + \tau.\, \mathbf{0} \\
P_{21} & =_{df} & \overline{c_2w1}.\, req_2.\, P_{22} \\
P_{22} & =_{df} & kr2.\, P_{24} + kr1.\, P_{23} \\
P_{23} & =_{df} & c_1r0.\, P_{24} + c_1r1.\, P_{22} + c_1r2.\, P_{22} \\
P_{24} & =_{df} & \overline{c_2w2}.\, P_{25} \\
P_{25} & =_{df} & c_1r0.\, P_{26} + c_1r1.\, P_{26} + c_1r2.\, P_{27} \\
P_{26} & =_{df} & \overline{kw2}.\, enter_2.\, exit_2.\, \overline{kw1}.\, \overline{c_2w0}.\, P_2 \\
P_{27} & =_{df} & \overline{c_2w1}.\, P_{22}
\end{array}$$

Some remarks on this representation may be helpful. The critical section of process P_i, where $i = 1$ or 2, is modelled as a pair of actions $enter_i$ and $exit_i$ representing, respectively, entry to and exit from the critical section. The noncritical section of each process is modelled as a summation, one summand of which represents the possibility that the process may halt, the other that it may proceed to request execution of its critical section. An action req_i appears in the definition of P_i. Its occurrence indicates that process P_i has 'just' indicated that it wishes to execute its critical section (by setting c_i to true). The reason for including these 'probes' will become clear below. Note also the presence of the agents P_{i7} and the way in which the statement **goto** L_0 is represented. The reason for this choice is that only the first $\overline{c_iw1}$ action (setting c_i to 1) is considered as signifying the initiation of an attempt by process i to execute its critical section.

The agent $Knuth$ has sort $K = \{enter_i, exit_i, req_i \mid i = 1, 2\}$. We introduce two derived modal operators:

$$\begin{array}{lll}
[K]A & \equiv & \bigwedge_{a\in K}[a]A \\
\langle K\rangle A & \equiv & \bigvee_{a\in K}\langle a\rangle A
\end{array}$$

We consider two questions. Firstly, does the algorithm preserve mutual exclusion? And secondly, is the algorithm live (in the sense that if a process requests execution of its critical section it will eventually enter its critical section)? We express these questions as follows.

1. We say that Knuth's algorithm *preserves mutual exclusion* iff

$$Knuth \models \texttt{PME}$$

where PME ('preserves mutual exclusion') is the following formula:

$$\nu Z.\, ((\neg(\langle exit_1\rangle\texttt{true} \wedge \langle exit_2\rangle\texttt{true})) \wedge [K]Z)$$

2. We say that Knuth's algorithm *is live* iff

$$Knuth \models \texttt{IL}$$

where IL is the formula

$$\nu Z.\, ([req_1]\texttt{EICS1} \wedge [req_2]\texttt{EICS2}) \wedge [K]Z$$

where for $i = 1, 2$, EICSi ('eventually in critical section i') is the formula

$$\mu Y.\, [\varepsilon]((\langle exit_i\rangle\texttt{true} \vee ([K]Y \wedge \langle K\rangle\texttt{true}))$$

Some clarifying remarks may be helpful.

(i) Process i is 'in its critical section' if P_i reaches a state in which it may perform the action $exit_i$. The formula PME is satisfied by an agent P of sort K iff for any $s \in K^*$ and agent P', if $P \overset{s}{\Longrightarrow} P'$ then $P' \not\models \langle exit_1 \rangle \mathtt{true} \wedge \langle exit_2 \rangle \mathtt{true}$. Thus $Knuth \models$ PME iff it never reaches a state with both P_1 and P_2 in their critical sections.

(ii) $P \models$ EICSi iff there are no sequence $\langle a_j \mid j < \omega \rangle \in K^\omega$ and no sequence $\langle Q_j \mid j < \omega \rangle$ of agents such that $Q_0 = Q$ and for all j, $Q_j \overset{a_j}{\Longrightarrow} Q_{j+1}$ and $Q_j \not\models \langle exit_i \rangle \mathtt{true}$. Thus $Knuth \models$ IL iff for $i = 1, 2$, there is no path on which occur infinitely-many visible actions and on which there is a 'probe' req_i (indicating that P_i has requested execution of its critical section) which is not followed by a corresponding action $enter_i$.

Using the Concurrency Workbench we have verified that Knuth's algorithm preserves mutual exclusion and is live (for more details see [Wa]). The process $Knuth$ consists of a number of agents in parallel. A more enterprising model checker would try to verify liveness and safety properties of $Knuth$ by verifying appropriate subproperties of its components. Proof rules for structured model checking for the modal sublanguage of the mu-calculus are presented in [Sti1]. We hope that these rules can be extended to the full mu-calculus.

6 Proofs of termination, soundness and completeness

We now prove the main results, theorems 3 and 4. First a little notation.

If B is a formula then $\mathcal{C}(B)$ is the set of constants occurring in B. Recall from section 4 that a tableau is a maximal proof tree with root labelled $s \vdash^{\mathcal{M}} A$. Given two nodes n and n' in a tableau with n' an immediate successor of n, we say that the sequent $s' \vdash_{\Delta'} B'$ labelling n' *succeeds* the sequent $s \vdash_\Delta B$ labelling n. Also, given two nodes n and n' in a tableau labelled $s \vdash_\Delta U$ and $s' \vdash_{\Delta'} U'$ respectively, we say that $s' \vdash_{\Delta'} U'$ *$\mathcal{C}$-succeeds* $s \vdash_\Delta U$ iff there is a sequence $\langle n_1, \ldots, n_k \rangle$ of nodes such that $n_1 = n$, $n_k = n'$, for $1 \leq i < k$, n_{i+1} is an immediate successor of n_i, and for $1 < i < k$, the formula of the sequent labelling n_i is not a constant.

Next we define a useful nonnegative integer measure, the *degree*, $d(B)$, of a closed formula B:

$$d(Q) = 0 \qquad d(\neg Q) = 0$$

$$\begin{aligned} d(U) &= 0 \\ d(\neg\neg B) &= 1 + d(B) \end{aligned}$$

$$\begin{aligned} d(B \wedge C) &= 1 + \max\{d(B), d(C)\} & \qquad d(\neg(B \wedge C) &= 1 + \max\{d(\neg B), d(\neg C)\} \\ d([a]B) &= 1 + d(B) & d(\neg([a]B) &= 1 + d(\neg B) \\ d(\nu Z.\, B) &= 1 + d(B[Z := U]) & d(\neg \nu Z.\, B) &= 1 + d(\neg B[Z := \neg U]) \end{aligned}$$

We extend this definition to sequents as follows:

$$d(s \vdash_\Delta B) = \begin{cases} d(B) & \text{if } B \text{ is not a constant} \\ d(\Delta(B)) & \text{otherwise} \end{cases}$$

Lemma 3.1 (i) If $s' \vdash_{\Delta'} B'$ succeeds $s \vdash_{\Delta} B$ and B' is not a constant, then $d(s' \vdash_{\Delta'} B') < d(s \vdash_{\Delta} B)$.

(ii) If $s' \vdash_{\Delta'} U'$ $\mathcal{C}$-succeeds $s \vdash_{\Delta} U$, then either $U' \in \mathcal{C}(\Delta(U)) \cup \{U\}$, or $d(s' \vdash_{\Delta'} U') < d(s \vdash_{\Delta} U)$ and $\mathcal{C}(\Delta(U')) \subseteq \mathcal{C}(\Delta(U)) \cup \{U\}$.

(iii) Suppose Δ is a prefix of Δ' and $U \in dom(\Delta)$. Then for any s, s', $d(s \vdash_{\Delta} U) = d(s' \vdash_{\Delta'} U)$.

Proof: (i) By inspection of the the tableau rules and the definition of degree.

(ii) Suppose $\Delta(U) = \nu Z.B$. Then either U' is a subformula of $B[Z := U]$, when $U' \in \mathcal{C}(\Delta(U)) \cup \{U\}$, or U' is introduced as $\nu Z'.C$ ($\neg\nu Z'.C$) which is a subformula of $B[Z := U]$, in which case $d(\nu Z'.C) < d(s \vdash_{\Delta} U)$ and $\mathcal{C}(\nu Z'.C) \subseteq \mathcal{C}(\Delta(U)) \cup \{U\}$ (and similarly for $\neg\nu Z'.C$).

(iii) Immediate from the definition. □

We now prove the termination theorem.

Theorem 3 Every tableau for $s \vdash^{\mathcal{M}} A$ is finite.

Proof: We omit the index $\mathcal{M}$.

Suppose there is an infinite tableau τ for $s \vdash A$. Since τ is finite-branching, there is an infinite path π through τ. Let $\sigma = \langle s_i \vdash_{\Delta_i} A_i \mid i < \omega \rangle$ be the sequence of sequents labelling the nodes of π. Since for each i, $s_{i+1} \vdash_{\Delta_{i+1}} A_{i+1}$ succeeds $s_i \vdash_{\Delta_i} A_i$, from Lemma 3.1(i) it follows that for infinitely many i, A_i is a constant. Also, since $\mathcal{M}$ is finite, no one constant appears infinitely often on π.

Consider the subsequence $\sigma' = \langle s'_i \vdash_{\Delta'_i} U_i \mid i < \omega \rangle$ of σ consisting of those sequents whose formulae are constants. Note that for each i, $s'_{i+1} \vdash_{\Delta'_{i+1}} U_{i+1}$ $\mathcal{C}$-succeeds $s'_i \vdash_{\Delta'_i} U_i$. Suppose i_0 is the largest i with $U_i = U_0$. Then since $\mathcal{C}(\Delta'_0(U_0)) = \emptyset$, by Lemma 3.1(ii), $d(s'_{i_0+1} \vdash_{\Delta'_{i_0+1}} U_{i_0+1}) < d(s'_{i_0} \vdash_{\Delta'_{i_0}} U_0)$ and $\mathcal{C}(\Delta'_{i_0+1}(U_{i_0+1})) \subseteq \{U_0\}$.

Now suppose i_1 is the largest i with $U_i = U_{i_0+1}$. Then again by Lemma 3.1(ii), $d(s'_{i_1+1} \vdash_{\Delta'_{i_1+1}} U_{i_1+1}) < d(s'_{i_1} \vdash_{\Delta'_{i_1}} U_{i_0+1})$ and $\mathcal{C}(\Delta'_{i_1+1}(U_{i_1+1})) \subseteq \{U_0, U_{i_0+1}\}$. By Lemma 3.1(iii), $d(s'_{i_1+1} \vdash_{\Delta'_{i_1+1}} U_{i_1+1}) < d(s'_{i_0+1} \vdash_{\Delta'_{i_0+1}} U_{i_0+1}) < d(s'_0 \vdash_{\Delta'_0} U_0)$.

By repeating this argument sufficiently often we obtain a contradiction since d is a nonnegative integer measure. □

Now we come to the proofs of soundness and completeness.

Theorem 4 $s \vdash^{\mathcal{M}} A$ has a successful tableau if and only if $s \in \| A \|_{V_{\mathcal{M}}}$.

Proof: First some notation and a standard lemma.

If $B = \nu Z.D$ then $B^0 = \texttt{true}$ and $B^{i+1} = D[Z := B^i]$.

If $B = \neg\nu Z.D$ then $B^0 = \texttt{false}$ and $B^{i+1} = \neg D[Z := \neg B^i]$.

Lemma 4.1 ($\mathcal{M}$ finite)

(i) If $B = \nu Z.D$ and $s \notin \| B_{\Delta} \|_V$, then there is $n < \omega$ such that $s \in \| (B^n)_{\Delta} \|_V - \| (B^{n+1})_{\Delta} \|_V$.

(ii) If $C = \neg\nu Z.D$ and $s \in \| C_{\Delta} \|_V$, then there is $n < \omega$ such that $s \in \| (C^{n+1})_{\Delta} \|_V - \| (C^n)_{\Delta} \|_V$. □

We omit the indices $\mathcal{M}$ and $V_{\mathcal{M}}$.

($\Longrightarrow$) Suppose $s \vdash A$ has a successful tableau τ. If all the leaves of τ are true (i.e. if whenever $t \vdash_{\Delta} B$ labels a leaf then $t \in \| B_{\Delta} \|$), then all the nodes of τ are true: for, as we noted earlier, the rules are backwards sound. So it suffices to show that all the leaves of τ are true.

If a leaf is labelled $t \vdash_{\Delta} B$ with $B = Q$, $\neg Q$ or $[a]C$, then it is certainly true. Hence any false leaf must be labelled $t \vdash_{\Delta} U$ with $\Delta(U) = \nu Z.B$. Suppose there is a false leaf. From amongst all false leaves choose one, labelled $t \vdash_{\Sigma} U$ say, such that there is no constant U' introduced before U in τ for which there is a false leaf labelled $t' \vdash_{\Sigma'} U'$ for some t', Σ'. Consider the subtableau τ_1 of τ whose root is the node, labelled $s \vdash_{\Delta} U$ say, at which U is introduced in τ. For each of the false leaves of τ labelled $t \vdash_{\Sigma} U$ for some t, Σ, by Lemma 4.1(i) there is $n < \omega$ such that $t \in \| (\nu Z.B)^{n}_{\Sigma} \| - \| (\nu Z.B)^{n+1}_{\Sigma} \|$ where $\Delta(U) = \nu Z.B$. Choose such a leaf l, labelled $t \vdash_{\Sigma} U$ say, such that the corresponding n is as small as possible. Note that since l is a leaf, there is above l in τ_1 a node k, the *companion node* of l, labelled $t \vdash_{\Sigma'} U$ for some Σ'.

Now transform the tableau τ_1 into a new tableau τ_1^* by replacing each definition list Δ' in a sequent of τ_1 by $\Delta'[(\nu Z.B)^n/U]$. An examination of the rules shows that if the leaves of τ_1^* are true then all the nodes of τ_1^* are true: the only rule which could prevent this, namely

$$\frac{s' \vdash_{\Delta'} \nu Z.B}{s' \vdash_{\Delta''} U} \quad \Delta'' \text{ is } \Delta' \cdot U = (\nu Z.B)^n$$

is not applied in τ_1^* since the root of τ_1^* is labelled $s \vdash_{\Delta[(\nu Z.B)^n/U]} U$. But the image of the successor of the companion node k of l under the transformation is false since it is labelled $t \vdash_{\Sigma'[(\nu Z.B)^n/U]} B[Z := U]$ and $t \notin \| (\nu Z.B)^{n+1}_{\Sigma'} \|$. Therefore some leaf of τ_1^* is false.

Suppose $t' \vdash_{\Delta''} U'$ labels such a false leaf where the corresponding leaf of τ_1 is labelled $t' \vdash_{\Delta'} U'$ so that $\Delta'' = \Delta'[(\nu Z.B)^n/U]$. Then by the choice of n we have that $U' \neq U$. Moreover, U' is not introduced before U in τ, since otherwise, by the observation immediately following Lemma 2, the leaf of τ labelled $t' \vdash_{\Delta'} U'$ would be false, contradicting the choice of U. Hence U' is introduced after U in τ.

But now we may apply the entire argument above to the tableau τ_1^*. And so on. But this contradicts Theorem 3, that every tableau is finite.

($\Longleftarrow$) We build a *pseudo-tableau* with root $s \vdash A$. The rules for pseudo-tableaux differ from those for tableaux in just one case: the rule for constants defined as minimal fixpoints. The pseudo-tableau rule is

$$\frac{t \vdash_{\Delta} U}{t \vdash_{\Delta'} \neg B[Z := \neg U]} \quad C, \text{ and } \Delta(U) = \neg \nu Z.B \text{ or } (\neg \nu Z.B)^n$$

where $\Delta' = \Delta[(\neg \nu Z.B)^k/U]$ with k such that $s \in \| (\neg \nu Z.B)^{k+1}_{\Delta} \| - \| (\neg \nu Z.B)^{k}_{\Delta} \|$. Note that by Lemma 4.1(ii), if $t \in \| U_{\Delta} \|$ then this rule is applicable (provided C holds), and in such a case, if $\Delta(U) = (\neg \nu Z.B)^n$ and $\Delta'(U) = (\neg \nu Z.B)^k$, then $k < n$. We assume the same termination conditions for pseudo-tableaux as for tableaux. Moreover, defining the degree function as in the proof of Theorem 3 with $d(\Delta(U)) = d(\neg \nu Z.B)$ when $\Delta(U) = (\neg \nu Z.B)^n$, then by an argument similar to that in the proof of Theorem 3 we have that every pseudo-tableau for $s \vdash A$ is finite (provided $\mathcal{M}$ is finite). Finally we define the notion of a *successful* pseudo-tableau as for tableaux with the requirement that no leaf is labelled $t \vdash_{\Delta} U$ where $\Delta(U) = (\neg \nu Z.B)^n$.

A successful pseudo-tableau can be transformed into a successful tableau simply by updating the definition lists, changing $\Delta(U)$ from $(\neg \nu Z.B)^n$ to $\neg \nu Z.B$ as necessary. Hence it suffices to show that there is a successful pseudo-tableau for $s \vdash A$. Such a pseudo-tableau may be constructed as follows.

Its root is labelled $s \vdash A$ and is true. Suppose $t \vdash_{\Delta} B$ labels a leaf of the partial pseudo-tableau and $t \in \| B_{\Delta} \|$. We define the successors of this node in the pseudo-tableau as follows depending on the structure of B.

(1) $B = Q$ or $\neg Q$: the node has no successors.

(2) $B = \neg\neg C$: the node has single true successor labelled $t \vdash_{\Delta} C$.

(3) $B = C \wedge D$ or $\neg(C \wedge D)$: if $B = C \wedge D$ then the node has two successors, one labelled $t \vdash_{\Delta} C$, the other $t \vdash_{\Delta} D$. Since $t \in \| B_{\Delta} \|$, the successors are true. If $B = \neg(B \wedge C)$ there is one true successor labelled $t \vdash_{\Delta} \neg C$ or $t \vdash_{\Delta} \neg D$.

(4) $B = [a]C$ or $\neg[a]C$: similar to (2) with the extra possibility that $\{t' \mid t \xrightarrow{a} t'\} = \emptyset$ in which case the node has no successors.

(5) $B = \nu Z.C$ or $\neg\nu Z.C$: if $B = \nu Z.C$ then since $t \in \| B_{\Delta} \|$, $t \in \| U_{\Delta'} \|$ where Δ' is $\Delta \cdot U = \nu Z.C$. Similarly for $\neg\nu Z.C$.

(6) $B = U$: if $\mathcal{C}$ holds and $\Delta(U) = \neg\nu Z.C$ or $(\neg\nu Z.C)^n$ then by Lemma 4.1 there is k with $t \in \| (\neg\nu Z.C)^{k+1}_{\Delta} \| - \| (\neg\nu Z.C)^{k}_{\Delta} \|$, when $t \in \| \neg C[Z := \neg U]_{\Delta'} \|$ where $\Delta' = \Delta[(\neg\nu Z.C)^k/U]$. The case $\Delta(U) = \nu Z.C$ is simpler.

By the remarks above we thus obtain a pseudo-tableau in which all the nodes are true. The only possible impediment to its success could be that $t \vdash_{\Delta} U$ labels a leaf where $\Delta(U) = (\neg\nu Z.B)^k$. But by the choices of k in the construction this is impossible. □

Acknowledgments

We would like to thank Rance Cleaveland, Kim Larsen and Bernhard Steffen for comments and discussions about model checking. The second author was supported by a grant from the Venture Research Unit of BP.

References

[Cl] R. Cleaveland, *Tableau-Based Model Checking in the Propositional Mu-Calculus*, Technical Report, University of Sussex 1988.

[CPS] R. Cleaveland, J. Parrow and B. Steffen, *The Concurrency Workbench*, to appear in Proc. IFIP 1989.

[EL] E. Emerson and C. Lei, *Efficient model checking in fragments of the propositional mu-calculus*, Proc. Symposium on Logic in Computer Science, Cambridge, Mass., 267–278, 1986.

[GS] S. Graf and J. Sifakis, *A modal characterization of observational congruence of finite terms of CCS*, Information and Control 68, 125–145, 1986.

[HM] M. Hennessy and R. Milner, *Algebraic laws for nondeterminism and concurrency*, JACM 32, 137–161, 1985.

[Ho] S. Holmström, *Hennessy-Milner Logic with Recursion as a Specification Language, and a Refinement Calculus based on it*, Report 44 Programming Methodology Group, University of Göteborg, 1988.

[Kn] D. Knuth, *Additional Comments on a Problem in Concurrent Programming Control*, Comm. ACM 9/5, 1966.

[Ko] D. Kozen, *Results on the propositional mu-calculus*, Theoretical Computer Science 27, 333-354, 1983.

[La] K. Larsen, *Proof systems for Hennessy-Milner logic with recursion*, Proc. CAAP 1988.

[Mi] R. Milner, Communication and Concurrency, Prentice-Hall 1989.

[Pn] A. Pnueli, *Specification and development of reactive systems*, Information Processing 86, North-Holland, 854–858, 1986.

[Pr] V. Pratt, *A decidable μ-calculus*, Proc. 22nd. FOCS, 421-27, 1981.

[St] B. Steffen, *Characteristic formulae*, to appear in Proc. ICALP 1989.

[Sti1] C. Stirling, *Modal Logics for Communicating Systems*, Theoretical Computer Science 49, 311–347, 1987.

[Sti2] C. Stirling, *Temporal Logics for CCS*, to appear in Proc. of REX Workshop, 1988.

[Wa] D. Walker, *Automated Analysis of Mutual Exclusion Algorithms Using CCS*, submitted for publication, 1988.

[Wi] G. Winskel, *Model Checking the Modal Nu-Calculus*, to appear in Proc. ICALP 1989.