An effective tableau system for the linear time mu-calculus

Julian Bradfield†, Javier Esparza‡, Angelika Mader‡

Abstract: We present a tableau system for the model checking problem of the linear time μ-calculus. It improves the system of Stirling and Walker by simplifying the success condition for a tableau. In our system success for a leaf is determined by the path leading to it, whereas Stirling and Walker's method requires the examination of a potentially infinite number of paths extending over the whole tableau.

Keywords: temporal logic, linear-time mu-calculus, local model-checking, tableau systems

1 Introduction.

Tableau techniques have been used for more than twenty years in order to establish validity of modal logics [HC68, Fit83]. A tableau system for a logic has three parts: deduction rules, termination conditions, and success conditions. The rules are goal-directed; they tell, given a sequent formalizing the statement we want to prove true, how to obtain subgoals. The termination conditions tell when to stop the construction of the proof tree, which we then call a tableau. Finally, the success conditions indicate when a tableau succeeds in establishing the truth of the root sequent.

Stirling has advocated the use of tableau techniques for local model-checking problems [Sti87]. Local model-checking asks whether a particular state has a temporal property, rather than, what is the set of states that satisfy it. Since tableau techniques check the properties of a given state by reference to properties of adjacent states, local model-checking may avoid having to compute all the states of the system.

Tableau techniques are particularly suitable for computer-assisted verification. They give very good insight into why a property holds. Also, they allow the verifier to apply her knowledge of the system to select the most promising course of action, by deciding which rule to apply or which branch of a proof tree to explore first.

Stirling and Walker have proposed tableau systems for mu-calculi, a group of fixpoint logics very popular in the formal verification community. A system for the modal mu-calculus (a branching time logic) is presented in [SW91], and a system for the linear-time mu-calculus (the linear-time counterpart of the modal mu-calculus), can be found in [SW90].

While the system of [SW91] is very simple and satisfactory, the one presented in [SW90] has very complicated success conditions. In the case of the modal mu-calculus, deciding whether a leaf of the tableau is successful can be done by examining the path

† LFCS, University of Edinburgh, King's Buildings, Edinburgh, United Kingdom, EH9 3JZ; jcb@dcs.ed.ac.uk

‡ Institut für Informatik, Technische Universität München, Arcisstr. 21, 80333 München, Germany; {esparza,mader}@informatik.tu-muenchen.de

of the tableau leading to it. On the contrary, the success condition of [SW90] requires the examination of a potentially infinite number of so-called extended paths, which are structures that may extend all over the tableau. The decidability of the success condition is difficult to prove, and in fact this point is not addressed in [SW90].

Stirling and Walker were aware of this problem, and they wrote ([SW90], p. 176) that "it may be possible to find a simpler definition of successful termination". This is precisely the contribution of this paper. We provide a simple, alternative tableau system, in which the success condition of a terminal only depends on the path leading to it. Our approach uses some ideas of [Kai95], where Kaivola addresses the satisfiability problem for the linear-time mu-calculus.

The paper is structured as follows. In section 2 we give basic definitions and results about the linear-time mu-calculus. In section 3 we present the tableau system, while section 4 illustrates it on an example. The proofs of soundness and completeness are in section 5, while section 6 discusses complexity issues.

2 The linear time μ-calculus.

We now define linear time μ-calculus syntax and semantics, and some notation for later use. The language is built from propositions, variables, boolean connectives, the minimal and maximal fixpoint operators μ and ν, and two temporal operators, the *strong nexttime* $\bigcirc$ and the *weak nexttime* $\odot$. Intuitively, $\bigcirc\phi$ means 'there is a next moment in time and ϕ is true at that moment', whereas $\odot\phi$ means 'if there is a next moment in time, then ϕ is true at that moment'.

Definition 1. Fix two disjoint countable sets, $\mathcal{Z}_C$, the set of *propositions*, and $\mathcal{Z}_V$, the set of *variables*, and define $\mathcal{Z} = \mathcal{Z}_C \cup \mathcal{Z}_V$. The formulae of νTL are defined by the abstract syntax:

$$\phi \ ::= \ Q \mid Z \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi \mid \odot\phi \mid \mu Z.\phi \mid \nu Z.\phi$$

where Q varies over $\mathcal{Z}_C$ and Z over $\mathcal{Z}_V$. The symbol σ ranges over both ν and μ. An occurrence of a variable Z in ϕ is *bound* iff it is within a subformula $\sigma Z.\phi'$ of ϕ and *free* otherwise. If Z is a variable, $\phi[\phi'/Z]$ is the result of simultaneously substituting ϕ' for all free occurrences of Z in ϕ. $\triangleleft$

Furthermore, in any νTL-formula ϕ, we assume that all the bound variables are distinct, and that all occurrences of bound variables are guarded, i.e. that each occurrence of variable Z in $\sigma Z.\phi$ is in a subformula of the type $\bigcirc\phi'$ or $\odot\phi'$. Any formula can be effectively transformed into an equivalent one fulfilling these restrictions.

Definition 2. A *transition system* is a set $\mathcal{S}$ of states together with a binary relation $\rightarrow$ on states. A *run* of a transition system is a maximal sequence $s_0 \rightarrow s_1 \rightarrow \cdots$ (which may be finite, if a state has no successor, or infinite). Let $\mathcal{R}$ be the set of runs. We shall let σ range over runs, and if $\sigma = s_0 \rightarrow s_1 \rightarrow \cdots$ then we write $\sigma(i)$ for s_i, and σ^i for $s_i \rightarrow s_{i+1} \rightarrow \cdots$. We write $\Sigma(s)$ for the set $\{\, \sigma \mid \sigma(0) = s \,\}$ of runs starting at s.

A mu-calculus *model* is a transition system $\mathcal{T} = (\mathcal{S}, \rightarrow)$ together with a valuation $\mathcal{V}\colon \mathcal{Z}_V \rightarrow 2^{\mathcal{R}}$, and a valuation $\mathcal{W}\colon \mathcal{Z}_C \rightarrow 2^{\mathcal{S}}$. The denotation $\|\phi\|^{\mathcal{T}}_{\mathcal{V},\mathcal{W}}$ of a mu-formula ϕ

in the model $(\mathcal{T}, \mathcal{V}, \mathcal{W})$ is given by the following rules (omitting the superscript $\mathcal{T}$ and the subscript $\mathcal{W}$, which do not change):

$$\|Z\|_{\mathcal{V}} = \mathcal{V}(Z)$$

$$\|Q\|_{\mathcal{V}} = \{\, \sigma \mid \sigma(0) \in \mathcal{W}(Q) \,\}$$

$$\|\phi_1 \wedge \phi_2\|_{\mathcal{V}} = \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}}$$

$$\|\phi_1 \vee \phi_2\|_{\mathcal{V}} = \|\phi_1\|_{\mathcal{V}} \cup \|\phi_2\|_{\mathcal{V}}$$

$$\|\bigcirc\phi\|_{\mathcal{V}} = \{\, \sigma \mid \sigma^1 \in \|\phi\|_{\mathcal{V}} \,\}$$

$$\|\odot\phi\|_{\mathcal{V}} = \{\, \sigma \mid \sigma^1 \in \|\phi\|_{\mathcal{V}} \,\} \cup \{\, \sigma \mid \sigma(0) \not\rightarrow \,\}$$

$$\|\nu Z.\phi\|_{\mathcal{V}} = \bigcup \{\, R \subseteq \mathcal{R} \mid \|\phi\|_{\mathcal{V}[Z:=R]} \supseteq R \,\}$$

$$\|\mu Z.\phi\|_{\mathcal{V}} = \bigcap \{\, R \subseteq \mathcal{R} \mid \|\phi\|_{\mathcal{V}[Z:=R]} \subseteq R \,\}$$

where $\mathcal{V}[Z := R]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ save that $\mathcal{V}'(Z) = R$.

We say that a state s satisfies a formula, $s \vDash \phi$, if all runs starting at s satisfy ϕ. $\triangleleft$

Definition 3. For all ordinals $\alpha \in \mathrm{Ord}$, the *fixpoint approximants* $\mu^\alpha Z.\phi$ and $\nu^\alpha Z.\phi$ are defined by: $\mu^0 Z.\phi = \mathrm{ff}$, $\nu^0 Z.\phi = \mathrm{tt}$, $\sigma^{\alpha+1} Z.\phi = \phi[\sigma^\alpha Z.\phi/Z]$, $\mu^\lambda Z.\phi = \bigvee_{\alpha<\lambda} \mu^\alpha Z.\phi$ and $\nu^\lambda Z.\phi = \bigwedge_{\alpha<\lambda} \nu^\alpha Z.\phi$, where λ is a limit ordinal. $\triangleleft$

Proposition 4. (Knaster-Tarski) $\mu Z.\phi = \bigvee_\alpha \mu^\alpha Z.\phi$, $\nu Z.\phi = \bigwedge_\alpha \nu^\alpha Z.\phi$. $\square$

Definition 5. The *μ-signature* $\mu\text{-}sig(\sigma, \phi)$ of a run σ at a formula ϕ (where $\sigma \vDash \phi$) is the least sequence $\zeta_1, \zeta_2, \ldots, \zeta_k$ such that $\sigma \vDash \phi[\mu^{\zeta_i} Z_i.\phi_i/\mu Z_i.\phi_i]$ where $\mu Z_i.\phi_i$ are the μ subformulae of ϕ in order of depth.

Dually, the *ν-signature* of $\sigma \nvDash \phi$ is defined as the least sequence such that $\sigma \nvDash \phi[\nu^{\zeta_i} Z_i.\phi_i/\nu Z_i.\phi_i]$. $\triangleleft$

3 The tableau system.

3.1 The sequents.

In the tableau system for the modal μ-calculus described in [SW91], the sequents have the form $s \vdash \phi$, where s is a state and ϕ is a formula. For the linear-time μ-calculus the sequents have to be a bit more complicated (as was already observed in [SW90]). The reason has to do with disjunction. In the modal case, $s \models A \vee B$ means $s \in \|A\| \cup \|B\|$, and therefore implies either $s \vDash A$ or $s \vDash B$. In consequence, the two rules

$$\frac{s \vdash A \vee B}{s \vdash A} \qquad \frac{s \vdash A \vee B}{s \vdash B}$$

are complete. In the linear-time case, $s \models A \vee B$ means $\Sigma(s) \subseteq \|A\| \cup \|B\|$. Since $\Sigma(s)$ is a set of runs, we can no longer infer $s \models \|A\|$ or $s \models \|B\|$: some runs in $\Sigma(s)$ may satisfy A but not B, and others B but not A. The solution is to allow sets of formulae in the right hand side of a sequent, which are interpreted disjunctively. This way, the rule

$$\frac{s \vdash A \vee B}{s \vdash A, B}$$

is sound and complete.

3.2 The rules.

If we have sets of formulae on the right of a sequent, there would usually be a choice of rules to apply, since several of the formulae may be eligible to have a rule applied. For technical reasons, it is helpful to have a unique rule to apply to a given sequent, so our system makes use of concurrent rule applications: a rule application in the tableau is actually the concurrent application of a maximal number of the basic rules; for example,

$$\frac{s \vdash A \vee B, C \wedge D}{s \vdash A, B, C \qquad s \vdash A, B, D}$$

is the concurrent application of the $\vee$ and $\wedge$ basic rules.

The basic rules are as one expects, namely

$$\wedge \qquad \frac{s \vdash \Gamma, A \wedge B}{s \vdash \Gamma, A \qquad s \vdash \Gamma, B}$$

$$\vee \qquad \frac{s \vdash \Gamma, A \vee B}{s \vdash \Gamma, A, B}$$

$$Q \qquad \frac{s \vdash \Gamma, Q}{s \vdash \Gamma} \quad \text{where } s \text{ fails } Q$$

$$\bigcirc \qquad \frac{s \vdash \bigcirc\Gamma, \odot\Delta}{s_1 \vdash \Gamma, \Delta \qquad \ldots \qquad s_n \vdash \Gamma, \Delta} \qquad \text{where } \{s_1, \ldots, s_n\} = \{\, s' \mid s \rightarrow s' \,\}$$

$$\sigma Z \qquad \frac{s \vdash \Gamma, \sigma Z.A}{s \vdash \Gamma, A[\sigma Z.A/Z]}$$

A rule is then the concurrent application of a maximal set of basic rules. We have the following lemma trivially from the definitions:

Lemma 6. The antecedent of a rule is true if and only if all its consequents are true.

□

Notice that the result of the application of a rule to a sequent is completely determined by the sequent. In other words, the children are completely determined by the parent.

3.3 Paths, internal paths, and terminals.

A *proof tree* is a tree of sequents constructed by the iterated application of these rules, starting with a root $s_0 \vdash \phi_0$. Given two nodes $\mathbf{n} = s \vdash \Gamma$ and $\mathbf{n}' = s' \vdash \Gamma'$, $\mathbf{n} \equiv \mathbf{n}'$ means that they are the same node, while $\mathbf{n} \simeq \mathbf{n}'$ means that $s = s'$ and $\Gamma = \Gamma'$. We extend these notations to sequences of nodes in the obvious way. (We shall use $\mathbf{n}, \mathbf{m}$ to range over nodes in a proof tree.)

Associated with a path π of the proof tree is a sequence $s \to s_1 \to \cdots \to s_m$ of transitions arising from the applications of the $\bigcirc$-rule on π. We call this sequence $trans(\pi)$.

The price to pay for allowing sets of formulae in the right hand side of a sequent is that a path of a proof tree is an object with a rather complicated internal structure: a set of *internal paths* describing the dependencies between formulae at different nodes. The path

$$\cfrac{\mathbf{n}_1 \colon s \vdash (\bigcirc A \wedge B) \vee \bigcirc B}{\cfrac{\mathbf{n}_2 \colon s \vdash \bigcirc A \wedge B, \bigcirc B}{\cfrac{\mathbf{n}_3 \colon s \vdash \bigcirc A, \bigcirc B}{\mathbf{n}_4 \colon s' \vdash A, B}}}$$

has the following internal paths:

$$\begin{array}{ccc} \mathbf{n}_1, (\bigcirc A \wedge B) \vee \bigcirc B & \qquad & \mathbf{n}_1, (\bigcirc A \wedge B) \vee \bigcirc B \\ \downarrow & & \downarrow \\ \mathbf{n}_2, \bigcirc A \wedge B & & \mathbf{n}_2, \bigcirc B \\ \downarrow & & \downarrow \\ \mathbf{n}_3, \bigcirc A & & \mathbf{n}_3, \bigcirc B \\ \downarrow & & \downarrow \\ \mathbf{n}_4, A & & \mathbf{n}_4, B \end{array}$$

It is intuitively clear that the truth of a sequent depends on the structure of the internal paths starting at it: in particular, it is important which μ or ν-variables are unfolded in those paths. Since the terminals of our tableau system will use these notions, we define them formally:

Definition 7. Let $\pi \equiv \mathbf{n}_1\mathbf{n}_2 \ldots$ be a path of a proof tree. An *internal path* of π is a finite or infinite sequence of pairs $(\mathbf{n}_1, \phi_1)(\mathbf{n}_2, \phi_2)\ldots$ such that ϕ_1 appears in $\mathbf{n}_1$, and for any two consecutive pairs $(\mathbf{n}_i, \phi_i)(\mathbf{n}_{i+1}, \phi_{i+1})$, one of the following cases holds:

- $\mathbf{n}_{i+1}$ is a child of $\mathbf{n}_i$, no basic rule is applied to ϕ_i, and $\phi_{i+1} = \phi_i$, or
- $\mathbf{n}_{i+1}$ is a child of $\mathbf{n}_i$, some basic rule different from Q is applied to ϕ_i, and ϕ_{i+1} is the formula given by the basic rule application.

An *internal circuit* of a finite path $\pi \equiv \mathbf{n}_1\mathbf{n}_2 \ldots \mathbf{n}_k$ such that $\mathbf{n}_1 \simeq \mathbf{n}_k$ is a finite sequence of internal paths of π

$$((\mathbf{n}_1, \phi_1)\ldots(\mathbf{n}_k, \phi_k)) \quad ((\mathbf{n}_1, \phi_{k+1})\ldots(\mathbf{n}_k, \phi_{2k}))\ldots$$

$$\ldots((\mathbf{n}_1, \phi_{jk+1})\ldots(\mathbf{n}_k, \phi_{(j+1)k})) \qquad \text{for } j \in \mathbb{N}$$

such that $\phi_{ik+1} = \phi_{ik}$ and $\phi_1 = \phi_{(j+1)k}$ and there are no two identical pairs in the circuit.

The *characteristic* of a finite internal path is either the highest variable that is unfolded in it, or the symbol $\perp$ if no variable is unfolded; the characteristic of an infinite internal path is the highest variable that is unfolded infinitely often. If the characteristic of an internal path is a μ-variable (ν-variable), then we also say that the path has μ-characteristic (ν-characteristic).

Let $\mathbf{n} \ldots \mathbf{n}'$ be a path. The relation $Int(\mathbf{n}, \mathbf{n}')$ is defined as the set of triples (ϕ, ϕ', Z) such that there exists an internal path $(n, \phi) \ldots (n', \phi')$ with characteristic Z. Sometimes we denote the triple (ϕ, ϕ', Z) as $(n, \phi) \xrightarrow{Z} (n', \phi')$. $\triangleleft$

It is easy to see that if the formula at the root of a proof tree is guarded, then the characteristic of any internal circuit is always different from $\perp$.

We can now define the terminal nodes, which are rather similar to those given by Kaivola in [Kai95] for the satisfiability problem.

Definition 8. A node $\mathbf{n} = s \vdash \Gamma$ is a *terminal* if

(i) $\Gamma = \emptyset$; or
(ii) $\Gamma = \bigcirc\Gamma', \odot\Gamma''$ (where Γ'' is non-empty) and s has no successor; or
(iii) $\Gamma = \bigcirc\Gamma'$ and s has no successor; or
(iv) $Q \in \Gamma$ and s satisfies Q; or
(v) $\mathbf{n}$ has a predecessor $\mathbf{n}' \simeq \mathbf{n}$ and every internal circuit of the path $\mathbf{n}' \ldots \mathbf{n}$ has μ-characteristic; we call $\mathbf{n}'$ the *companion* of $\mathbf{n}$; or
(vi) $\mathbf{n}$ is not a terminal of type (v), and it has two predecessors $\mathbf{n}'' \simeq \mathbf{n}' \simeq \mathbf{n}$, with $\mathbf{n}''$ above $\mathbf{n}'$, such that $Int(\mathbf{n}'', \mathbf{n}') = Int(\mathbf{n}'', \mathbf{n})$; the nodes $\mathbf{n}''$ and $\mathbf{n}'$ are called the *second* and *first companions* of $\mathbf{n}$.

Terminals of types (i), (iii) and (v) are *unsuccessful*, and terminals of types (ii), (iv) and (vi) are *successful*.

A *tableau* is a finite proof tree whose leaves (and no other node) are terminals. A tableau is *successful* if all its terminals are successful. $\triangleleft$

We have the following result

Proposition 9. There is a unique tableau with a given root.
Proof. The children of a nonterminal node are determined solely by the node, and the termination conditions are deterministic. $\square$

We briefly explain the intuition behind the definition of the terminals. Each path of a tableau can be seen as an attempt to construct a *false* run of the system, i.e., a run which does not satisfy the formula at the root. The terminals identify the points at which we have gathered enough information, either to construct such a run (unsuccessful terminal), or to give up searching the continuations of the path (successful terminal). Let π be a path of the tableau ending in a terminal $\mathbf{n}$, and let $\sigma = trans(\pi)$.

- If $\mathbf{n}$ is of type (i), then its parent is of the form $s \vdash Q$, and no run starting at s satisfies Q. So every run of the form $\sigma\sigma'$ is false.
- if $\mathbf{n}$ is of type (ii), then σ is already a false run.
- if $\mathbf{n}$ is of type (iii), then σ is a true run, and there are no continuations of σ.

- If $\mathbf{n}$ is of type (iv), then all runs of the form $\sigma\sigma'$ are true; therefore, we can give up the search.
- If $\mathbf{n}$ is of type (v), then the run $\sigma_1(\sigma_2)^\omega$ is false, where $\sigma_2 = \mathit{trans}(n' \ldots n)$ and $\sigma_1\sigma_2 = \sigma$; loosely speaking, the reason is that in any chain of dependencies corresponding to this run some μ-variable is unfolded infinitely often.
- If n is of type (vi), then we cannot say that all runs of the form $\sigma\sigma'$ are true. However, if such a run is false, then there exists also a false run of the form $\sigma''\sigma'$, where $\sigma'' = \mathit{trans}(\mathbf{n}_0 \ldots \mathbf{n}'' \ldots \mathbf{n}')$. Moreover, the path showing the falsity of $\sigma''\sigma'$ will be shorter than the path for $\sigma\sigma'$ (loosely speaking, it will not contain the part $\mathbf{n}' \ldots \mathbf{n}$). So we can give up the search.

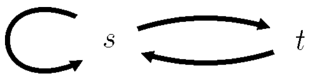
This concludes the presentation of the tableau system. A comparison with the systems of [SW90] and [Kai95] can be found in the conclusions.

4 Example.

For an example, we shall check the formula

$$\nu Z.\mu X.\bigcirc((Z \wedge P) \vee X)$$

'(on every path) infinitely often P' on the state s of the following system, where P holds at the state t.



This formula fails at s, so the constructed tableau will contain an unsuccessful terminal. We shall only show the first branch containing an unsuccessful terminal, since that is sufficient to fail.

For readability, we use the variables Z, X to represent the associated fix-point formulae; and ϕ abbreviates $((Z \wedge P) \vee X)$. The terminals marked with # are the unsuccessful terminals (all of type (v)); both of them fail due to an internal path of the form

$$s, \bigcirc\phi \rightarrow s, (Z \wedge P) \vee X \rightarrow s, X \rightarrow s, \bigcirc\phi$$

which is the only internal path from the companion $s \vdash \bigcirc\phi$ to the terminal $s \vdash \bigcirc\phi$.

Tableaux can be quite large—we have only given one unsuccessful branch for this one. If the property being checked is true, one must construct the entire tableau, but as noted above, in the case of failure it suffices to find one unsuccessful branch. It is this feature which allows local model-checking: the user may construct the tableau depth-first, and decide at every node with several children which one to examine first.

The second tableau is the tableau for the same formula on the system with the self-loop from s to s deleted. In this case, the formula is true, and accordingly the tableau is successful. The successful terminals (marked with ✓) are some of type (iv) and some

$$\dfrac{s \vdash \nu Z.\mu X.\bigcirc\phi}{\dfrac{s \vdash \mu X.\bigcirc\phi}{\dfrac{s \vdash \bigcirc\phi}{\dfrac{s \vdash (Z \wedge P) \vee X}{\dfrac{s \vdash Z \wedge P, X}{\dfrac{s \vdash P, \bigcirc\phi}{\#\ s \vdash \bigcirc\phi} \qquad \dfrac{s \vdash Z, \bigcirc\phi}{\ldots}}} \qquad \dfrac{t \vdash (Z \wedge P) \vee X}{\ldots}}}}$$

∞

$$\dfrac{s \vdash \nu Z.\mu X.\bigcirc\phi}{\dfrac{s \vdash \mu X.\bigcirc\phi}{\dfrac{\mathbf{n}'':\ s \vdash \bigcirc\phi}{\dfrac{t \vdash (Z \wedge P) \vee X}{\dfrac{t \vdash Z \wedge P, X}{\checkmark t \vdash P, \bigcirc\phi \qquad \dfrac{t \vdash Z, \bigcirc\phi}{\dfrac{t \vdash X, \bigcirc\phi}{\dfrac{t \vdash \bigcirc\phi}{\dfrac{s \vdash (Z \wedge P) \vee X}{\dfrac{s \vdash Z \wedge P, X}{\dfrac{s \vdash P, \bigcirc\phi}{\dfrac{\mathbf{n}':\ s \vdash \bigcirc\phi}{\dfrac{t \vdash (Z \wedge P) \vee X}{\dfrac{t \vdash Z \wedge P, X}{\dfrac{t \vdash Z, \bigcirc\phi}{\dfrac{t \vdash X, \bigcirc\phi}{\dfrac{t \vdash \bigcirc\phi}{\dfrac{s \vdash (Z \wedge P) \vee X}{\dfrac{s \vdash Z \wedge P, X}{\dfrac{s \vdash P, \bigcirc\phi}{\checkmark s \vdash \bigcirc\phi} \qquad \dfrac{s \vdash Z, \bigcirc\phi}{\dfrac{s \vdash X, \bigcirc\phi}{\checkmark\ \mathbf{n}:\ s \vdash \bigcirc\phi}}}}}}} \qquad \checkmark t \vdash P, \bigcirc\phi}}}} \qquad \dfrac{s \vdash Z, \bigcirc\phi}{\dfrac{s \vdash X, \bigcirc\phi}{\dfrac{\mathbf{m}':\ s \vdash \bigcirc\phi}{\dfrac{t \vdash (Z \wedge P) \vee X}{\dfrac{t \vdash Z \wedge P, X}{\dfrac{t \vdash Z, \bigcirc\phi}{\dfrac{t \vdash X, \bigcirc\phi}{\dfrac{t \vdash \bigcirc\phi}{\dfrac{s \vdash (Z \wedge P) \vee X}{\dfrac{s \vdash Z \wedge P, X}{\dfrac{s \vdash P, \bigcirc\phi}{\checkmark s \vdash \bigcirc\phi} \qquad \dfrac{s \vdash Z, \bigcirc\phi}{\dfrac{s \vdash X, \bigcirc\phi}{\checkmark s \vdash \bigcirc\phi}}}}}}} \qquad \checkmark t \vdash P, \bigcirc\phi}}}}}}}}}}}}}}}$$

of type (vi). One of the type (vi) leaves is labelled $\mathbf{n}$, and its companions are labelled $\mathbf{n}''$ and $\mathbf{n}'$; the others are all similar. The relations $Int(\mathbf{n}'', \mathbf{n}')$ and $Int(\mathbf{n}'', \mathbf{n})$ are both

$$\{(\bigcirc\phi, \bigcirc\phi, Z), (\bigcirc\phi, \bigcirc\phi, X)\}$$

This example also shows that tableaux can be very redundant. For instance, the two subtableaux rooted at $\mathbf{n}'$ and the similar node $\mathbf{m}'$ to its right (labelled $s \vdash \bigcirc\phi$). This redundancy is typical of tableau methods, and there exist several techniques that palliate it, which are out of the scope of this paper (see [Mad92, Mad95]).

5 Soundness and completeness.

In this section we prove soundness and completeness of the tableau system. The completeness proof is rather straightforward, while the soundness proof is quite delicate.

Theorem 10. The (unique) tableau for a given root is finite.
Proof. Let τ be the tableau with root $s_0 \vdash \phi_0$, let n be the number of states of the transition system, and let m be the number of symbols of ϕ_0. It is easy to see that the size of the closure of ϕ_0 is bounded by m. Therefore, τ contains at most $n \cdot 2^m$ different sequents, and there are at most 2^{m^3} different Int relations. By the definition of the terminals of type (vi), every path of τ has at most length $O(n \cdot 2^{m^3})$. Since a node has finitely many children, a tableau contains finitely many nodes. □

Theorem 11. If $s_0 \models \phi_0$, then there exists a successful tableau.
Proof.

Starting with root $s_0 \vdash \phi_0$, apply the rules until a tableau is constructed. The construction terminates by Theorem 10. Assume that the tableau so constructed contains an unsuccessful terminal $\mathbf{n} = s \vdash \Gamma$. We prove that $\mathbf{n}$ is a false node, which contradicts Lemma 6. There are three cases:

(1) $\mathbf{n}$ is of type (i).
Then the parent of $\mathbf{n}$ is of the form $s \vdash Q$, and no run starting at s satisfies Q. So the parent of $\mathbf{n}$ is false.

(2) $\mathbf{n}$ is of type (iii).
Then $\mathbf{n}$ is false by definition.

(3) $\mathbf{n}$ is of type (v).
Let $\sigma = trans(\mathbf{n}' \ldots \mathbf{n})$. Then σ^ω is a run of the transition system with s as initial state. We shall prove that σ^ω does not satisfy any formula of Γ.

Assume that σ^ω satisfies some formula ϕ of Γ. We construct an internal circuit $\pi \equiv (\mathbf{n}_1, \phi_1)(\mathbf{n}_2, \phi_2) \ldots$ of $\mathbf{n}' \ldots \mathbf{n}$ such that $\phi_1 = \phi$. To serve as guide during the construction, we inductively associate to each pair $(\mathbf{n}_i, \phi_i)$ a suffix ρ_i of σ^ω satisfying the following two properties: the state of $\mathbf{n}_i$ is $\rho_i(0)$, and ρ_i satisfies ϕ_i.

The suffix associated with the initial pair $(\mathbf{n}_1, \phi_1)$ is σ^ω itself. We explain how to select the $(i+1)$-th element of the internal path and its suffix, given the i-th element and its suffix. With the convention that the children of a type (v) terminal node are the children of its companion, we have

- If no basic rule is applied to ϕ_i, then $\mathbf{n}_{i+1}$ is an arbitrary child of $\mathbf{n}_i$, $\phi_{i+1} = \phi_i$ and $\rho_{i+1} = \rho_i$.
- If $\phi_i = Q$ and Q is applied to ϕ_i, then $(\mathbf{n}_i, \phi_i)$ is the last node of π.
- If $\phi_i = A \wedge B$ and $\wedge$ is applied to ϕ_i, then $\mathbf{n}_{i+1}$ is an arbitrary child of $\mathbf{n}_i$, ϕ_{i+1} is either A or B, according to our choice of $\mathbf{n}_{i+1}$, and $\rho_{i+1} = \rho_i$.
- If $\phi_i = A \vee B$ and $\vee$ is applied to ϕ_i, then $\mathbf{n}_{i+1}$ is the only child of $\mathbf{n}_i$, $\rho_{i+1} = \rho_i$, and $\phi_{i+1} = A$ if $\mu\text{-}sig(\rho_i, A) \leq \mu\text{-}sig(\rho_i, B)$ or $\phi_{i+1} = B$ otherwise.
- if $\phi_i = \bigcirc A$ or $\phi = \odot A$, so $\bigcirc$ is applied to ϕ_i, then $\phi_{i+1} = A$, $\rho_{i+1} = \rho_i^{(1)}$, and $\mathbf{n}_{i+1}$ is the child of $\mathbf{n}_i$ having $\rho_{i+1}(0)$ as state.
- if $\phi_i = \sigma Z.A$ and σZ is applied to ϕ_i, then $\mathbf{n}_{i+1}$ is the only child of $\mathbf{n}_i$, $\phi_{i+1} = A[\sigma Z.A/Z]$, and $\rho_{i+1} = \rho_i$.

There are now two possible sub-cases:

(3.1) π is finite.

Then its last node must be of the form $(\mathbf{n}_i, Q)$, and the Q-rule is applied to Q. Therefore, no run starting at the state of $\mathbf{n}_i$ satisfies Q. This contradicts the fact that ρ_i satisfies $\phi_i = Q$.

(3.2) π is infinite.

Let Z be the characteristic of π. Then Z is also the characteristic of some internal circuit of the path $\mathbf{n}' \ldots \mathbf{n}$, and therefore a μ-variable. Assign to each element $(\mathbf{n}_i, \phi_i)$ of π a *truncated signature* obtained from $\mu\text{-}sig(\rho_i, \phi_i)$ by removing all ordinals corresponding to μ-variables lower than Z (the truncated signature is therefore a prefix of the signature). Let T_π be the sequence of truncated signatures associated with π.

The sequence T_π is non-increasing, because no variable higher than Z is ever unfolded, and because of the way we defined the internal path at the $\vee$-nodes. Moreover, since Z is unfolded infinitely often in π, T_π contains an infinite, strictly decreasing subsequence, which contradicts the well-foundedness of the lexicographic order. □

Theorem 12. If there exists a successful tableau, then $s_0 \vDash \phi_0$.

Proof. Let τ be the tableau for $s_0 \vdash \phi_0$. Suppose that τ is successful, but that $s_0 \nvDash \phi_0$; we shall derive a contradiction.

An extended path of τ is a finite or infinite sequence of nodes $\mathbf{n}_1\mathbf{n}_2 \ldots$ such that for every two consecutive nodes $\mathbf{n}_i$, $\mathbf{n}_{i+1}$ either $\mathbf{n}_i$ is a σ-terminal and $\mathbf{n}_{i+1}$ a child of its (first) companion, or $\mathbf{n}_i$ is a nonterminal and $\mathbf{n}_{i+1}$ is one of its children. The internal paths of an extended path are defined as those of normal paths.

Assuming that $s_0 \nvDash \phi_0$ there must be a run σ_0 which does not satisfy ϕ_0. We shall use this run to show the existence of an unsuccessful terminal, contradicting the success of τ.

Following σ_0, we define inductively an extended path $\pi \equiv \mathbf{n}_0\mathbf{n}_1\mathbf{n}_2 \ldots$ of τ. To serve as guide during the construction, we inductively associate to each node $\mathbf{n}_i$ a suffix ρ_i of σ_0 such that the state of $\mathbf{n}_i$ is $\rho_i(0)$ and ρ_i fails every formula of $\mathbf{n}_i$. The suffix associated with the root $\mathbf{n}_0$ is σ_0. If $\mathbf{n}_i$ is a terminal of type (i) – (iv), then $(\rho_i, \mathbf{n}_i)$ is the last element of π. Otherwise, its successor $\mathbf{n}_{i+1}$ and the associated suffix ρ_{i+1} are chosen as follows, where as usual we consider the children of a type (v) or (vi) terminal to be the children of its (first) companion:

- if the $\bigcirc$-rule is applied to $\mathbf{n}_i$, then $\rho_{i+1} = \rho_i^{(1)}$, and $\mathbf{n}_{i+1}$ is the child of $\mathbf{n}_i$ having $\rho_{i+1}(0)$ as state;
- otherwise, the rule applied to $\mathbf{n}_i$ is some combination of the basic rules Q, $\vee$, σZ and $\wedge$, and $\rho_{i+1} = \rho_i$ with $\mathbf{n}_{i+1}$ that child of $\mathbf{n}_i$ given by choosing the branches in the $\wedge$ rules to preserve the ν-signature of ρ_i (if ρ fails $A \wedge B$ with ν-signature ξ, then ρ fails either A or B with ν-signature ξ).

In particular, it follows from this definition that every node of π is false.

We consider two cases:

(a) π is finite.
Let $\mathbf{n}$ be the last node of π. By construction, $\mathbf{n}$ is a terminal of type (i)-(iv). It cannot be a terminal of type (iii) or (iv), because $\mathbf{n}$ is false. So $\mathbf{n}$ is an unsuccessful terminal, and we are done.

(b) π is infinite.
Since τ contains only finitely many sequents, we can extract from π an infinite subsequence $\mathbf{n}_{m_1}, \mathbf{n}_{m_2}, \dots$ such that $\mathbf{n}_{m_i} \equiv \mathbf{n} = s \vdash \bigcirc\Gamma$ for some terminal $\mathbf{n}$.

Let $\phi_1, \dots, \phi_n$ be an arbitrary enumeration of the set Γ. We assign to each node $\mathbf{n}_{m_i}$ a vector of signatures

$$\overline{x}_i = (\ \nu\text{-}sig(\bigcirc\phi_1, \rho_{m_i}), \dots, \nu\text{-}sig(\bigcirc\phi_n, \rho_{m_i})\ )$$

By Dickson's Lemma, there exist two indices i, j such that $\overline{x}_i \leq \overline{x}_j$, where $\leq$ is the pointwise order on vectors.

We prove that every internal circuit of $\mathbf{n}_{m_i} \dots \mathbf{n}_{m_j}$ has a μ-characteristic.

Let γ be an arbitrary internal circuit of $\mathbf{n}_{m_i} \dots \mathbf{n}_{m_j}$ with a ν-characteristic Z. Assign to each element $(\mathbf{n}_i, \phi_i)$ of γ a *truncated signature* obtained by removing from $\nu\text{-}sig(\phi_i, \rho_i)$ all ordinals corresponding to ν-variables lower than Z (the truncated signature is therefore a prefix of the signature). Let T_γ be the sequence of truncated signatures corresponding to γ. (Note here the sequence of signatures has the form

$$(\nu\text{-}sig(\phi_l, \rho_{m_i}), \dots, \nu\text{-}sig(\phi_{l'}, \rho_{m_j})\nu\text{-}sig(\phi_{l'}, \rho_{m_i+1}), \dots;$$

that is, we are following a particular formula many times round a prefix of ρ_{m_i}, *not* following it all the way along ρ_{m_i}. The fact that $\overline{x}_i \leq \overline{x}_j$ allows us to make use of this apparently odd construction.)

We claim that T_γ is non-increasing. Let $(\mathbf{n}_k, \phi_k)$ and $(\mathbf{n}_{k+1}, \phi_{k+1})$ be two consecutive nodes of γ. If $\mathbf{n}_k \equiv \mathbf{n}_{m_j}$, then $\mathbf{n}_{k+1} \equiv \mathbf{n}_{m_i+1}$, and since $\overline{x}_i \leq \overline{x}_j$ we are done; if $\mathbf{n}_k \not\equiv \mathbf{n}_{m_j}$ then $\mathbf{n}_{k+1}$ is a successor of $\mathbf{n}_k$, and the truncated signature cannot increase when moving from $\mathbf{n}_k$ to $\mathbf{n}_{k+1}$ because no variable higher than Z is ever unfolded.

Since Z is unfolded somewhere in γ, the last element of T_γ is lexicographically less than the first. This implies that the last full signature is also lexicographically less than the first, which contradicts the assumption $\overline{x}_i \leq \overline{x}_j$. So Z is a μ-variable.

Now consider the extended path $\mathbf{n}_0 \dots \mathbf{n}_{m_i} \dots \mathbf{n}_{m_j}$. We have $\mathbf{n}_{m_i} \equiv \mathbf{n}_{m_j}$. Therefore, τ contains an extended path $\pi_0 \equiv \mathbf{n}_0 \dots \mathbf{n}_{m_i}(\mathbf{n}_{m_i+1} \dots \mathbf{n}_{m_j})^\omega$. We have:

(1) the path π_0, seen as a sequence of sequents, is ultimately periodic;

(2) every internal circuit of the path $\mathbf{n}_{m_i} \dots \mathbf{n}_{m_j}$ has a μ-characteristic.

The extended paths satisfying these two properties are called in the sequel *μ-periodic paths.*

Since τ is successful, there are no type (v) terminals in π_0. We now show that τ contains another μ-periodic path $\pi_1 \simeq \mathbf{n}_0 \ldots \mathbf{n}_p(\mathbf{n}_{p+1} \ldots \mathbf{n}_l)^\omega$ satisfying in addition to (1) and (2) an additional property:

(3) the first terminal of τ contained in π_1 is at or after $\mathbf{n}_l$

It follows from (3) that $\mathbf{n}_0 \ldots \mathbf{n}_l$ is a path of τ, and that $\mathbf{n}_p$ is the first companion of $\mathbf{n}_l$; then, by (2), $\mathbf{n}_l$ is a type (v) terminal, which is impossible.

Note that we wrote $\pi_1 \simeq \ldots$, rather than $\pi_1 \equiv \ldots$. The rough idea of the following algorithm is this: we start with π_0, and then repeatedly 'cut out' segments between two similar (i.e. $\simeq$) nodes. The result of 'cutting out' a segment is not actually an extended path itself, but it does define an extended path similar to the cut path. This works because the tableau rules are deterministic (this is where we use the maximal concurrent combination of basic rules), so similar nodes have similar successors. For example, if we cut the bracketted portion out of

$$\mathbf{n}_0 \ldots \mathbf{n}_i[\mathbf{n}_{i+1} \ldots \mathbf{n}_j]\mathbf{n}_{j+1} \ldots$$

where $\mathbf{n}_j \simeq \mathbf{n}_i$, then since $\mathbf{n}_i$ has a successor similar to $\mathbf{n}_{j+1}$, and so on, we define an extended path which is similar to

$$\mathbf{n}_0 \ldots \mathbf{n}_i\mathbf{n}_{j+1} \ldots$$

Accordingly, the precise statement of condition (3) is $\pi_1 \equiv \mathbf{m}_0 \ldots \mathbf{m}_p\mathbf{m}_{p+1} \ldots$, such that $\mathbf{m}_0 \equiv \mathbf{n}_0$, and $\pi_1 \simeq \mathbf{n}_0 \ldots \mathbf{n}_p(\mathbf{n}_{p+1} \ldots \mathbf{n}_l)^\omega$, and the first $\mathbf{m}_i$ that is a terminal has index $i \geq l$.

The algorithm is thus, where we write $\pi :\simeq \ldots$ to mean 'set π to the extended path generated by $\ldots$':

input: $\pi_0 \equiv \mathbf{n}_0 \ldots \mathbf{n}_{m_i}(\mathbf{n}_{m_i+1} \ldots \mathbf{n}_{m_j})^\omega$

begin
$\pi_1 := \pi_0$;
$k :=$ index of the first node of π_1 which is a terminal of τ;
$p := m_i$ % index of the last node before π_1's period;
$l := m_j$ % index of the last node of π_1's period;

while $k < l$ **do**
  $k' :=$ index of $\mathbf{n}_k$'s first companion
  $k'' :=$ index of $\mathbf{n}_k$'s second companion;
  % terminal before the period
  if $k \leq p$ **then**
    $\pi_1 :\simeq \mathbf{n}_0 \ldots \mathbf{n}_{k'-1}\mathbf{n}_k \ldots (\mathbf{n}_{p+1} \ldots \mathbf{n}_l)^\omega$
  % terminal in the period, first companion before
  elseif $k' \leq p$ **then**
    $\pi_1 :\simeq \mathbf{n}_0 \ldots \mathbf{n}_{k'-1}(\mathbf{n}_k \ldots \mathbf{n}_l\mathbf{n}_{p+1} \ldots \mathbf{n}_{k-1})^\omega$;
  % terminal and first companion in the period, second companion before

elseif $k'' \leq p$ **then**

$\pi_1 :\simeq \mathbf{n}_0 \ldots \mathbf{n}_{k''-1}(\mathbf{n}_k \ldots \mathbf{n}_l \mathbf{n}_{p+1} \ldots \mathbf{n}_{k-1})^\omega$

% terminal and both companions in the period

else

$\pi_1 :\simeq \mathbf{n}_0 \ldots (\mathbf{n}_{p+1} \ldots \mathbf{n}_{k'-1} \mathbf{n}_k \ldots \mathbf{n}_l)^\omega$

endif

$p :=$ index of the first node of π_1's period;

$l :=$ index of the last node of π_1's period

$k :=$ index of first node of π_1 that is a terminal

endwhile

end

We prove that the assertion 'π_1 is a μ-periodic path' is an invariant of the algorithm for the four branches of the **if** command:

- First branch: easy.
- Second and third branches: the internal circuits of $\mathbf{n}_k \ldots \mathbf{n}_l \mathbf{n}_{p+1} \ldots \mathbf{n}_{k-1}$ are the same as those of $\mathbf{n}_{p+1} \ldots \mathbf{n}_{k-1} \mathbf{n}_k \ldots \mathbf{n}_l$, but just shifted, so the characteristics are the same.
- Last branch: the internal circuits of $\mathbf{n}_{p+1} \ldots \mathbf{n}_{k''} \ldots \mathbf{n}_{k'-1} \mathbf{n}_k \ldots \mathbf{n}_l$ do not coincide with those of $\mathbf{n}_p \ldots \mathbf{n}_{k''\ldots} \mathbf{n}_{k'} \ldots \mathbf{n}_k \ldots \mathbf{n}_l$. However, since we have $\mathit{Int}(\mathbf{n}_{k''}, \mathbf{n}_k) = \mathit{Int}(\mathbf{n}_{k''}, \mathbf{n}_{k'})$, we can guarantee that all of them have μ-characteristic.

Moreover, the algorithm terminates, because each iteration of the loop decreases l. After termination, there is no terminal before $\mathbf{n}_l$, and by the μ-periodicity property, $\mathbf{n}_l$ satisfies the conditions to be an unsuccessful terminal, which contradicts the success of τ. □

6 Complexity issues.

Let us first consider the problem of deciding if, given a state s and a formula ϕ, some path starting at s does *not* satisfy ϕ (this is the complement of the model-checking problem as defined in this paper).

We may nondeterministically guess a path of the tableau with root $s \vdash \phi$ leading to an unsuccessful terminal. The length of a path is $O(n \cdot 2^{m^3})$, where n is the number of states of the transition system and m the number of symbols of the formula. So we have a nondeterministic algorithm with linear space complexity in the size of the system and exponential in the length of the formula.

It is possible to do better if we observe that we do not have to store all the path. We may nondeterministically guess its length k, and the position k' of the companion. Then, it suffices to store the current node n_i, and, from the moment we reach the $k+1$-th node on, the relation $\mathit{Int}(\mathbf{n}_k, \mathbf{n}_i)$, because the characteristics of the internal paths can be obtained from there. An element of the *Int* relation can be stored in $O(m)$ space, and there are at most $O(m^3)$ of them. So we need $O(m^4 + \log n)$ space (including the space to store k and k'). It is still possible to improve things a bit by observing that we do not really need to store every element of $\mathit{Int}(\mathbf{n}_k, \mathbf{n}_i)$, because we are only interested in the characteristics of the internal circuits. In consequence, for every pair of formulae (ϕ, ψ), we only need to store the element $(\phi, \psi, Z) \in \mathit{Int}(\mathbf{n}_k, \mathbf{n}_i)$ having the *highest* Z. This reduces the space to $O(m^3 + \log n)$. Some other tricks (see [Kai95]) reduce it further to $O(m^2 \log m + \log n)$. Using standard results of complexity theory, it is then possible to obtain deterministic algorithms with $2^{O(m^2 \log m + \log n)}$ space and time, or $O(f)$ space and $O(2^f)$ time, where $f \in O((m^2 \log m + \log n)^2)$. These deterministic results also apply, of course, to the model-checking problem.

7 Conclusions.

We have presented a tableau system for the linear-time mu-calculus with simpler success conditions than those of Stirling and Walker's system [SW90]. The success condition of [SW90] uses the notion of *extended path*. Loosely speaking, an extended path is a sequence of nodes in which the successor of an element is either one of its children or, in case the element is a fixpoint terminal, its companion (in [SW90] fixpoint terminals only have one companion). The success condition for a terminal $\mathbf{n}$ requires to examine the internal circuits of *all* the extended paths leading from the companion $\mathbf{n}'$ of $\mathbf{n}$ to $\mathbf{n}$. Such extended paths may visit many different terminals of the tableau, because the subtree with root $\mathbf{n}'$ may have many different leaves. This makes the condition very difficult to verify. In fact, it is not easy to prove that the condition is decidable (this point is not addressed in [SW90]). Our success condition requires to examine only the internal circuits of the *unique* path leading from the companion to the terminal, and makes the complexity analysis very simple.

The price to pay for the simpler success condition is a larger tableau. In [SW90], the construction of a branch of the tableau terminates whenever we hit a sequent we have seen before (in the branch). In our system, we may have to continue further. So, in some sense, our tableau can be seen as an unfolding of the tableau of [SW90], in which the success conditions are more transparent and easier to check.

Kaivola also sketches a tableau system for the satisfiability problem at the end of [Kai95]. The system checks if some run satisfies a formula. A tableau consists of one path and a *recurrence point*, a node that must be guessed by the user. These points increase very much the number of possible tableaux for a given root.Our system improves Kaivola's by getting rid of recurrence points. In this way, we obtain a 'conventional' tableau system, closer to the system for the modal mu-calculus of [SW91].

The complexity of our system (discussed in the previous section) coincides with the complexity of model-checking techniques based on automata theory (see, for instance, [Var88]). Our aim is not to compete with these techniques, which are probably better for automatic verification, but to provide a logical system which may be used to solve small examples by hand, explains why a formula holds, and allows us to use knowledge about the system to speed up the verification process.

8 References.

[Fit83] M. Fitting, *Proof Methods for Modal and Intuitionistic Logics* (Reidel, 1983).

[HC68] G.E. Hughes and M.J. Creswell, *An Introduction to Modal Logic*, (Methuen and Co., 1968)

[Kai95] R. Kaivola, A Simple Decision Method for the Linear Time Mu-calculus, *Proc. Int. Workshop on Structures in Concurrency Theory* (J. Desel, ed.) (1995)

[Mad92] A. Mader, Tableau Recycling, *Proc. CAV '92*, LNCS **663** (1992).

[Mad95] A. Mader, Modal μ-Calculus, Model Checking and Gauß Elimination, *Proc. TACAS'95*, to appear in LNCS. (1995)

[Sti87] C. P. Stirling, Modal logics for communicating systems. *Theoret. Comput. Sci.* **49** 311–347 (1987).

[SW90] C. Stirling and D. Walker, CCS, liveness, and local model checking in the linear time mu-calculus, *Proc. First International Workshop on Automatic Verification Methods for Finite State Systems*, LNCS **407** 166–178. (1990).

[SW91] C. Stirling and D. Walker, Local Model Checking in the Modal Mu-calculus, *Theor. Comput. Sci.* **89**, 161–177. (1991).

[Var88] M. Vardi, A Temporal Fixpoint Calculus, *Proc. 15th PoPL*, 250–259. (1988)