TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ-Calculus

Richard Mayr

TUM-I9634
SFB-Bericht Nr.342/15/96 A
Oktober 1996

TUM–INFO–10-96-I34-80/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ-Calculus

Richard Mayr *

Abstract

We present a tableau system for model checking Petri nets with a fragment of the linear time μ-calculus (without the strong nexttime operator). The model checking problem is already known to be decidable for the full linear time μ-calculus [Esp], but the algorithm has non-elementary complexity and gives no insight into the proof process. In contrast to this, tableau systems give a good intuition why a property holds. Although the tableau system is sound and complete and yields a decision procedure, it is not intended to be used as such, but rather as a method for proving and understanding properties expressed in this calculus.

Keywords: temporal logic, linear-time μ-calculus, Petri nets, model checking, tableau systems

1 Introduction

Model checking is a very successful technique for verifying temporal properties of concurrent systems, which is viewed as being essentially algorithmic. The corresponding standard algorithms fall into two classes: the iterative algorithms and the tableaux-based algorithms. The iterative algorithms compute all the states of the system which have the desired property, and usually yield higher efficiency in the worst case. The tableaux-based algorithms are designed to check whether a particular expression has a temporal property. This is called local model checking which avoids the investigation of for the verification irrelevant parts of the process being verified. This is due to the fact that the truth of a property at a state may depend on a small neighborhood of the state, rather than the whole system. Therefore this method is applicable for the verification of systems with infinite state spaces. In local model checking the proof system is developed in a goal directed fashion (top down).

Tableau systems consist of tree parts: deduction rules, termination conditions and success conditions. The rules are goal-directed; they transform a goal into subgoals that must be proved, in order to establish the truth of the root-sequent. The termination conditions tell when to stop the construction of the tableau (the proof tree). This is the case when enough information has been collected to decide the truth/falsity of a branch

*Address: Institut für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München, Germany; e-mail: mayrri@informatik.tu-muenchen.de

in the tableau. Finally, the success conditions tell if the tableau succeeds in establishing the truth of the root-sequent.

Tableau systems are particularly suitable for computer-assisted verification, i.e. theorem provers with human interaction. This is due to the fact that they give a good intuition on why a property holds and allow the user to apply his knowledge on the system to guide the construction of the tableau by choosing the most promising branches of the tableau, which will be explored first. Although some tableau systems (like the one in this paper) are complete in the sense that they yield a decision procedure for the chosen logic and model of systems, they are not intended to be used as such, but rather as a proof method.

In the field of fixpoint logics, μ-calculi have become very popular in the formal verification community. Most important are the modal μ-calculus, which is interpreted over the states of systems, and the linear time μ-calculus, which is interpreted over runs. Complete tableau systems for these calculi and finite state systems have been described in [SW90] and [SW91, BEM96], respectively.

For infinite state systems the situation is different. While model checking with the modal μ-calculus is undecidable even for very simple models of infinite state concurrent systems (for example BPP) [Esp], the tableau systems for model checking Petri nets described in [Bra92] still provide a useful tool for proving properties. In contrast to this, model checking Petri nets with the linear time μ-calculus is decidable [Esp], but no tableau method existed yet. This is the contribution of this paper. We use a restricted version of the linear time μ-calculus without the strong nexttime operator, but including the weak nexttime operator.

Section 2 contains basic definitions about the linear time μ-calculus. Section 3 defines the tableau system, while section 4 contains the proofs of its soundness and completeness. In section 5 we discuss an example and in section 6 we show some possible generalizations of the tableau method. The paper closes with a section about complexity matters and related work.

2 The Linear Time μ-Calculus

We now define the linear time μ-calculus and the restricted fragment that will be used in the next section. The language is built from a set of 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 this moment', whereas $\odot\,\Phi$ means 'if there is a next moment in time, then Φ is true at this moment'.

Definition 2.1 Fix two disjoint countable sets, $\mathcal{Z}_C$, the set of *propositions*, and $\mathcal{Z}_V$, the set of *variables*, and define $\mathcal{Z} = \mathcal{Z}_C \cup \mathcal{Z}_V$. Let Act be a countable set of atomic actions. Atomic actions will be denoted by $a, b, c, \ldots$ and sets of actions by $A, B, \ldots$. The formulae of μTL are defined by the abstract syntax:

$$\Phi ::= Q \mid Z \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \bigcirc_A \Phi \mid \odot_A \Phi \mid \mu Z.\Phi \mid \nu Z.\Phi$$

where Q ranges over $\mathcal{Z}_C$ and Z over $\mathcal{Z}_V$. The symbol σ ranges over $\{\mu, \nu\}$. 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.

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

Definition 2.2 A *labelled transition system* T over a set of actions Act consists of a (possibly infinite) set of states S and a binary relation $\stackrel{a}{\rightarrow} \subseteq S \times S$ for each $a \in Act$. The system is called *rooted* if it has a distinguished initial state $s_0 \in S$. A path of T is either an infinite sequence $s_0 \stackrel{a_0}{\rightarrow} s_1 \stackrel{a_1}{\rightarrow} s_2 \stackrel{a_2}{\rightarrow} s_3 \ldots$ or a finite sequence $s_0 \stackrel{a_0}{\rightarrow} s_1 \stackrel{a_1}{\rightarrow} \ldots \stackrel{a_{n-1}}{\rightarrow} s_n$ such that s_n has no successors. A *run* is a path that starts at the initial state. Let $\mathcal{R}$ be the set of runs. We shall let σ range over runs, and if $\sigma = s_0 \stackrel{a_0}{\rightarrow} s_1 \stackrel{a_1}{\rightarrow} \cdots$ then we write $\sigma(i)$ for s_i and σ^i for the path $s_i \stackrel{a_i}{\rightarrow} s_{i+1} \stackrel{a_{i+1}}{\rightarrow} \cdots$. The i-th action a_i will be denoted by σ_i.

A μ-calculus model is a transition system $\mathcal{T}$ together with a valuation $\mathcal{V} : \mathcal{Z}_V \rightarrow 2^{\mathcal{R}}$ and a valuation $\mathcal{W} : \mathcal{Z}_C \rightarrow 2^S$. The denotation $\| \Phi \|^{\mathcal{T}}_{\mathcal{V},\mathcal{W}}$ of a μ-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):

$$\begin{array}{rcl}
\|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\
\|Q\|_{\mathcal{V}} & = & \{\sigma \mid \sigma(0) \in \mathcal{W}(Q)\} \\
\|\Phi_1 \wedge \Phi_2\|_{\mathcal{V}} & = & \|\Phi_1\|_{\mathcal{V}} \cap \|\Phi_2\|_{\mathcal{V}} \\
\|\Phi_1 \vee \Phi_2\|_{\mathcal{V}} & = & \|\Phi_1\|_{\mathcal{V}} \cup \|\Phi_2\|_{\mathcal{V}} \\
\|\bigcirc_A \Phi\|_{\mathcal{V}} & = & \{\sigma \mid \sigma_0 \in A \wedge \sigma^1 \in \|\Phi\|_{\mathcal{V}}\} \\
\|\odot_A \Phi\|_{\mathcal{V}} & = & \{\sigma \mid \sigma_0 \in A \wedge \sigma^1 \in \|\Phi\|_{\mathcal{V}}\} \cup \{\sigma \mid \sigma_0 \notin A \vee \sigma(0) \not\rightarrow\} \\
\|\nu Z.\Phi\|_{\mathcal{V}} & = & \bigcup\{R \subseteq \mathcal{R} \mid \|\Phi\|_{\mathcal{V}[Z:=R]} \supseteq R\} \\
\|\mu Z.\Phi\|_{\mathcal{V}} & = & \bigcap\{R \subseteq \mathcal{R} \mid \|\Phi\|_{\mathcal{V}[Z:=R]} \subseteq R\}
\end{array}$$

where $\mathcal{V}[Z := R](Z') := \begin{cases} \mathcal{V}(Z), & \text{if } Z \neq Z' \\ R, & \text{if } Z = Z' \end{cases}$.

A state s satisfies a formula Φ, $s \models \Phi$, iff all runs starting at s satisfy Φ, i.e. $\{\sigma \mid \sigma(0) = s\} \subseteq \|\Phi\|$. For a run σ we also write $\sigma \models \Phi$ instead of $\sigma \in \|\Phi\|$.

Definition 2.3 For all ordinals $\alpha \in Ord$, the *fixpoint approximants* $\mu^\alpha Z.\Phi$ and $\nu^\alpha Z.\Phi$ are defined by: $\mu^0 Z.\Phi = ff$ and $\nu^0 Z.\Phi = tt$, $\sigma^{\alpha+1} Z.\Phi = \Phi[\sigma^\alpha Z.\Phi/Z]$, $\mu^\lambda Z.\Phi = \bigvee_{\alpha \leq \lambda} \mu^\alpha Z.\Phi$, $\nu^\lambda Z.\Phi = \bigwedge_{\alpha \leq \lambda} \nu^\alpha Z.\Phi$, where λ is a limit ordinal.

Proposition 2.4 (Knaster-Tarski) $\mu Z.\Phi = \bigvee_\alpha \mu^\alpha Z.\Phi$, $\nu Z.\Phi = \bigwedge_\alpha \nu^\alpha Z.\Phi$.

Definition 2.5 The μ-signature $\mu\text{-}sig(\sigma, \Phi)$ of a run σ w.r.t. a formula Φ (where $\sigma \models \Phi$) is the lexicographically least sequence $\zeta_1, \ldots, \zeta_k$ such that $\sigma \models \Phi[\mu^{\zeta_i} Z_i.\Phi_i / \mu Z_i.\Phi_i]$ where $\mu Z_i.\Phi_i$ are the μ-subformulae of Φ in order of depth (i.e. in some (fixed) order such that subformulae appear after any containing subformulae).

Dually, the ν-signature of $\sigma \not\models \Phi$ is the least sequence s.t. $\sigma \not\models \Phi[\nu^{\zeta_i} Z_i.\Phi_i / \nu Z_i.\Phi_i]$.

These preliminary definitions apply to finite or infinite systems. For a thorough treatment of finite systems refer to [BEM96] and [SW91]. Here we are interested in infinite systems described by general Petri nets. Esparza [Esp] has proved that model checking

Petri nets with the linear time μ-calculus is decidable, provided that the propositions satisfy certain restrictions. The algorithm works by reduction to the reachability problem for Petri nets and Büchi-automata. One problem is that this algorithm has non-elementary complexity and is therefore not useful in practice. (The problem is at least as hard as the reachability problem for Petri nets and therefore EXPSPACE-hard). Even more important is that the algorithm yields hardly any insight on why a property holds and is useless as a proof method. The tableau method described in this paper is a generalization of the tableau method of [BEM96] and can be used as a proof technique.

Definition 2.6 A labelled Petri net $N = (S, T, W, L, Act)$ consists of a finite set of places S, a finite set of transitions T, a function $W : S \times T \cup T \times S \to \mathbb{N}$ that assigns weights to the arcs, a set of actions Act and a labelling function $L : T \to Act$ that assigns actions to the transitions. Markings of nets will be denoted by Σ. As a technicality markings will be mappings $S \mapsto (\mathbb{N} \cup \{\omega\})$ instead of $S \mapsto \mathbb{N}$, where ω is the first limit ordinal.

In the rest of the paper we will use a restricted version of the linear time μ-calculus.

Definition 2.7 (Restrictions)

- *The strong nexttime operator $\bigcirc$ will be left out completely. This is done in order to make it impossible to express the state of deadlock with the calculus and so to avoid having to solve the reachability problem for Petri nets in the tableau. As mentioned earlier the model checking problem is decidable for the full linear time μ-calculus [Esp], but so far there exist no tableau methods for solving the reachability problem for Petri nets.*

  So we only use the weak nexttime operator $\odot$.

- *The propositions $Q \in \mathcal{Z}_C$ must satisfy two restrictions.*

  Q1 $\Sigma \in \mathcal{W}(Q) \;\Rightarrow\; \forall \Sigma' \leq \Sigma.\ \Sigma' \in \mathcal{W}(Q)$

  Q2 $(\Sigma + \omega\Sigma') \notin \mathcal{W}(Q) \;\Rightarrow\; \exists k \in \mathbb{N}\ \forall k' \geq k.\ (\Sigma + k'\Sigma') \notin \mathcal{W}(Q)$.

3 The Tableau System

3.1 The Sequents

An important difference between the modal μ-calculus and the linear time μ-calculus is the treatment of disjunction. In a tableau system for the **modal** μ-calculus (see [SW91]) the sequents have the form $\Sigma \vdash \Phi$, which means that the state s satisfies the formula Φ. So $\Sigma \models \Phi \vee \Psi$ means $\Sigma \in \|\Phi\| \cup \|\Psi\|$ and therefore implies either $\Sigma \models \Phi$ or $\Sigma \models \Psi$. Thus the two rules

$$\frac{\Sigma \vdash \Phi \vee \Psi}{\Sigma \vdash \Phi} \qquad \frac{\Sigma \vdash \Phi \vee \Psi}{\Sigma \vdash \Psi}$$

are complete.

For the **linear** time μ-calculus the situation is different, because here $\Sigma \models \Phi \vee \Psi$ means $\{\sigma | \sigma(0) = \Sigma\} \subseteq \|\Phi\| \cup \|\Psi\|$. As $\{\sigma | \sigma(0) = \Sigma\}$ is a set of runs (that can have more than one element) we can no longer infer $\Sigma \models \|\Phi\|$ or $\Sigma \models \|\Psi\|$: some runs starting at Σ

may satisfy Φ but not Ψ, while others may satisfy Ψ and not Φ. The solution is to allow sets of formulae in the right hand side of the sequent that are interpreted disjunctively (see [BEM96]). This way, the rule

$$\frac{\Sigma \vdash \Phi \vee \Psi}{\Sigma \vdash \Phi, \Psi}$$

is sound and complete.

3.2 The Rules

The rules for the tableau can be divided into two groups: the basic rules and the special rules. While the basic rules are sufficient for finite state systems the special rules are needed for the treatment of Petri nets. We will define and discuss the basic rules first as they are more intuitive, and make the necessary adjustments and extensions by the special rules later. For convenience of notation we define that $\Gamma, \Delta, \ldots$ denote sequences of formulae (i.e. $\Gamma = \Phi_1, \Phi_2, \ldots, \Phi_n$ and $\odot_A \Gamma = \odot_A \Phi_1, \ldots, \odot_A \Phi_n$).

$$\wedge \quad \frac{\Sigma \vdash \Gamma, \Phi \wedge \Psi}{\Sigma \vdash \Gamma, \Phi \quad \Sigma \vdash \Gamma, \Psi}$$

$$\vee \quad \frac{\Sigma \vdash \Gamma, \Phi \vee \Psi}{\Sigma \vdash \Gamma, \Phi, \Psi}$$

$$Q \quad \frac{\Sigma \vdash \Gamma, Q}{\Sigma \vdash \Gamma} \quad \text{where } \Sigma \notin \mathcal{W}(Q)$$

$$\odot \quad \frac{\Sigma \vdash \odot_{A_1} \Gamma_1, \ldots, \odot_{A_n} \Gamma_n}{\Sigma_1 \vdash \Delta \ \ldots \ \Sigma_k \vdash \Delta} \quad \text{where } \Delta = \Gamma_1, \ldots, \Gamma_k \text{ and}$$

$$\{\Sigma_1, \ldots, \Sigma_k\} = \{\Sigma' \mid \exists a \in \bigcap_{i=1}^{n} A_i.\ \Sigma \xrightarrow{a} \Sigma'\}$$

$$\sigma Z \quad \frac{\Sigma \vdash \Gamma, \sigma Z.\Phi}{\Sigma \vdash \Gamma, \Phi[\sigma Z.\Phi / Z]}$$

Additionally use the following "cleanup-procedure" after each rule application: If a formula Φ occurs more than once in a sequence Γ, then delete all occurrences but the first.

Lemma 3.1 *The antecedent of a rule is true if and only if all its consequents are true.*

Proof Trivially from the definitions. □

For these basic rules 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. We'll see later that this is not the case for the special rules. There several ancestors (in the path from the root to the sequent) must be taken into account.

3.3 Paths and Internal Paths

A *proof tree* is a tree of sequents constructed by the iterated application of rules, starting with a root $\Sigma_0 \vdash \Phi_0$. Associated with a path π in a proof tree is a sequence $\sigma = t_1, t_2, \ldots, t_n$ of transitions arising from the applications of the $\odot$-rule in π. We denote this by $\sigma = \mathit{trans}(\pi)$.

In a tableau each node is assigned a unique label n_i. Let $n_i : \Sigma_i \vdash \Phi_i$ and $n_j : \Sigma_j \vdash \Phi_j$ be two nodes in the tableau. By $n_i \simeq n_j$ we mean that $\Sigma_i = \Sigma_j$ and $\Phi_i = \Phi_j$. We write $n_i \gg n_j$ if n_j occurs earlier on the path from the root to n_i. It follows that $\gg$ is a partial order on the set of nodes in a tableau.

The price we pay for allowing sets of formulae in the right hand side of a sequent is that a path in the proof tree has a more complex internal structure: a set of *internal paths* describing the dependencies between formulae at different nodes. The path

$$\dfrac{\dfrac{\dfrac{n_1 : \Sigma \vdash (\odot_{\{a,b\}}\Phi \wedge \Psi) \vee \odot_{\{a,c\}}\Psi}{n_2 : \Sigma \vdash \odot_{\{a,b\}}\Phi \wedge \Psi, \odot_{\{a,c\}}\Psi}}{n_3 : \Sigma' \vdash \Phi \wedge \Psi, \Psi}}{n_4 : \Sigma' \vdash \Phi, \Psi}$$

has the following internal paths:

$$\begin{array}{ccc} n_1 : (\odot_{\{a,b\}}\Phi \wedge \Psi) \vee \odot_{\{a,c\}}\Psi & \qquad & n_1 : (\odot_{\{a,b\}}\Phi \wedge \Psi) \vee \odot_{\{a,c\}}\Psi \\ \downarrow & & \downarrow \\ n_2 : \odot_{\{a,b\}}\Phi \wedge \Psi & & n_2 : \odot_{\{a,c\}}\Psi \\ \downarrow & & \downarrow \\ n_3 : \Phi \wedge \Psi & & n_3 : \Psi \\ \downarrow & & \downarrow \\ n_4 : \Phi & & n_4 : \Psi \end{array}$$

Intuitively the truth of a sequent depends on the structure of the internal paths starting at it, particularly on which μ or ν-variables are unfolded in those paths.

Definition 3.2 (Internal paths, Internal circuits) *Let π be a path of the proof tree. An* internal path *of π is a finite or infinite sequence of tripels $(n_1, \Sigma_1, \Phi_1)(n_2, \Sigma_2, \Phi_2), \ldots$ s.t. Φ_1 appears in n_1, and for any two consecutive pairs $(n_1, \Sigma_i, \Phi_i), (n_{i+1}, \Sigma_{i+1}, \Phi_{i+1})$, one of the following cases holds:*

- *n_{i+1} is a child of n_i, no rule is applied to Φ_i and $\Phi_{i+1} = \Phi_i$, or*
- *n_{i+1} is a child of n_i, some rule different from Q is applied to Φ_i, and Φ_{i+1}, Σ_{i+1} are the formula/marking given by the rule application.*

An internal circuit *of a finite path $\pi = n_1 n_2 \ldots n_k$ such that $n_1 : \Sigma_1 \vdash \Gamma$, $n_k : \Sigma_k \vdash \Gamma$ and $\Sigma_k \geq \Sigma_1$ is a finite sequence of internal paths of π*

$$\begin{array}{l} ((n_1, \Sigma_1, \Phi_1) \ldots (n_k, \Sigma_k, \Phi_k)) \quad ((n_1, \Sigma_1, \Phi_{k+1}) \ldots (n_k, \Sigma_k, \Phi_{2k})) \ldots \\ \ldots ((n_1, \Sigma_1, \Phi_{jk+1}) \ldots (n_k, \Phi_{(j+1)k})) \quad \textit{for } j \in \mathbb{N} \end{array}$$

such that $\Phi_{ik+1} = \Phi_{ik}$, $\Phi_1 = \Phi_{(j+1)k}$ and $\Sigma_1 \leq \Sigma_k$ and $\Phi_1 \in \Gamma$.

The characteristic *of a finite internal path is the highest variable that is unfolded (by the σZ-rule) 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 ν-variable (μ-variable), then we say that the path has ν-characteristic (μ-characteristic). For a path $n \ldots n'$ the set $Int(n, n')$ is defined as the set of tripels (Φ, Φ', Z) such that there exists an internal path $(n, \Phi) \ldots (n', \Phi')$ with characteristic Z.*

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

Before defining the special rules we must make some additions. We assign each node a label consisting of a finite set of pairs $\in \mathbb{N}^l \times N$, where l is the number of places of the Petri net and N the set of nodes in the tableau. The label of the root node is the empty set. For a node n with state Σ, label D and sequence of formulae Γ we write $n(D) : \Sigma \vdash \Gamma$. If the label is of no concern we just write "?" for it.

Child-nodes do not inherit these labels; they are only introduced by special rules.

3.4 The Special Rules

Now that we have defined these notions we can define the special rules. Here $n \longrightarrow n'$ means that node n is replaced by n'.

ω $\quad n_2(D) : \Sigma_2 \vdash \Gamma \longrightarrow n_3(D) : \Sigma_2 + \omega(\Sigma_2 - \Sigma_1) \vdash \Gamma$
if there is a previous node $n_1(?) : \Sigma_1 \vdash \Gamma$ s.t. $n_2 \gg n_1$, $\Sigma_2 \geq \Sigma_1$
and there is a place s s.t. $\Sigma_1(s) < \Sigma_2(s) \neq \omega$

M $\quad n(D) : \Sigma \vdash \Gamma \longrightarrow n(D \cup \{(\delta, n'')\}) : \Sigma \vdash \Gamma$
if there are two ancestors $n'(?) : \Sigma \vdash \Gamma$ and $n''(?) : \Sigma \vdash \Gamma$ s.t. $n \gg n' \gg n''$
$Int(n'', n') = Int(n'', n)$ and δ is the Parikh-vector of the sequence of transitions fired between n' and n

Special rules take precedence over basic rules and rule ω takes precedence over rule M.

The intuition for the special rules is as follows:

ω Let Σ be a marking and $\Sigma' \geq \Sigma$. If there is an unsuccessful run starting at Σ, then the same run can also start at Σ'. So the chance to find an unsuccessful run is better if the start-marking is larger. The new marking with ω represents the infinitely many reachable markings with arbitrarily high numbers of tokens in these places. Note that the ω does not mean that there are infinitely many tokens on this place, but only that there are reachable markings with arbitrarily high numbers of tokens.

M If the conditions for the M-rule are satisfied then the path from node n' to node n gives us no new information for the construction of an unsuccessful run. This is because of the condition $Int(n'', n') = Int(n'', n)$. The only thing worth remembering are the changes δ in the marking of the net. By adding the vector to the label of the node we remember that we could insert this piece of the branch as often as we want, and change the marking by δ. This is necessary, because later in the tableau it might turn out that we should have inserted this piece of the branch between n' and n a certain number of times in order to be able to construct an infinite unsuccessful run. However, at the point where the M-rule is applied we don't know yet how often to insert this part of the branch.

Now we can define the terminal nodes.

Definition 3.3 A node $n(?) : \Sigma \vdash \Gamma$ is a *terminal* if any of the following conditions is satisfied:

1. $\Gamma = Q$ and $\Sigma \notin \mathcal{W}(Q)$
2. $\Gamma = \odot_{A_1}\Gamma_1, \ldots, \odot_{A_n}\Gamma_n$ and $\nexists a \in \bigcap_{i=1}^{n} A_i, \Sigma'.\ \Sigma \xrightarrow{a} \Sigma'$
3. $Q \in \Gamma$ and $\Sigma \in \mathcal{W}(Q)$
4. n has an ancestor $n' \simeq n$ s.t. $n' \ll n$ and
   - every internal circuit of the path $n' \ldots n$ has μ-characteristic, and
   - Let δ_0 be the Parikh-vector of the sequence of fired transitions between n' and n. Let $\{\delta_1, \ldots, \delta_k\} := \{\delta \mid \exists \tilde{n}.\ n' \ll \tilde{n}(D) \ll n \ \wedge\ \exists(\delta, \hat{n}) \in D.\hat{n} \gg n'\}$. There are $x_1, \ldots, x_k \in \mathbb{N}$ s.t. $\delta_0 + x_1\delta_1 + \ldots + x_k\delta_k \geq \vec{0}$.
5. There are nodes $n''(?) : \Sigma \vdash \Gamma$ and $n'(D) : \Sigma \vdash \Gamma$ s.t. $n'' \ll n' \ll n$. Let π_1 be the path between n'' and n' and π_2 the path between n' and n. Let δ be the Parikh-vector of the sequence of transitions fired between in π_1. $\pi_1 = \pi_2$ and $\exists \tilde{n}.\ (\delta, \tilde{n}) \in D$.

Terminals of type 1 and 4 are *unsuccessful*, and terminals of type 2,3 and 5 are *successful*.

A *tableau* is a finite proof tree whose leaves (and no other nodes) are terminals. A tableau is *successful* iff all its terminals are successful.

Lemma 3.4 *There is a unique tableau with a given root.*

Proof For the basic rules the children of a nonterminal node are determined only by the node and the rules are deterministic. For the special rules the changes of a node depend only on the node and its predecessors. The precedence conditions ensure that these changes are deterministic too. Finally the termination conditions are deterministic. □

The intuition behind the definition of the special rules and the terminals is the following: Each path of the tableau can be seen as an attempt to construct a *false* run of the system, i.e. a run that 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 n, and let $\sigma = trans(\pi)$.

- If n is of type 1, then it is of the form $\Sigma \vdash Q$, and no nun starting at Σ satisfies Q. Therefore any run of the form $\sigma\sigma'$ is false.
- If n is of type 2 then any run of the form $\sigma\sigma'$ is a true run. This is due to the definition of $\odot_A$ as σ has no continuations σ' starting with an action in $\bigcap_{i=1}^{n} A_i$.
- If n is of type 3 then any run of the form $\sigma\sigma'$ is true.
- If n is of type 4, then an infinite unsuccessful run can be constructed. Basically this is because that in any chain of dependencies corresponding to this run some μ-variable is unfolded infinitely often. The details will be explained in section 4.
- If n is of type 5, then nothing new has happened between n' and n. This is because the same path has already occurred earlier in the tableau between n'' and n'. Even the Parikh-vector of transitions fired between n' and n has already been recorded in the label of n'. Basically this means that if any unsuccessful run can be found, then it can be found elsewhere in the tableau in an easier (shorter) way.

4 Soundness and Completeness

First recall a general lemma that is very useful for decidability problems about Petri nets.

Lemma 4.1 (Dickson's lemma) *Given an infinite sequence of vectors $M_1, M_2, M_3, \ldots$ in $\mathbb{N}^k$ there are $i < j$ s.t. $M_i \leq M_j$ ($\leq$ taken componentwise).*

Lemma 4.2 *The tableau for a given root is finite.*

Proof Let τ be the tableau with root $\Sigma_0 \vdash \Phi_0$ and m the number of symbols in Φ_0. It is easy to see that the size of the closure (i.e. the subformulae, modulo unfolding) of Φ_0 is bounded by m. Therefore at most 2^m different sequents Γ can occur in nodes of the tableau and there are at most 2^{m^3} different *Int* relations. Let t be the number of transitions in the Petri net. Then each node has at most $max\{2, t\}$ children.

Assume that there is an infinite path in the tableau. Because of the special rule ω and Dickson's lemma (4.1) the number of different markings Σ occurring in nodes of the tableau is finite. Thus there are only finitely many different paths between different nodes with the same marking Σ and the same sequence of formulae Γ.

Because of the special rule M all the Parikh-vectors of these paths will eventually be stored in the labels of the nodes. So the path will end by termination condition 5.

Thus every path in the tableau has finite length. As each node has only finitely many children the tableau is finite. □

Lemma 4.3 *If $\Sigma_0 \models \Phi_0$ then there is a successful tableau.*

Proof Starting with the root-node $n_0(\{\}) : \Sigma_0 \vdash \Phi_0$, apply the rules until the tableau is constructed. The construction terminates by Lemma 4.2.

We will assume that there exists an unsuccessful terminal $n(D) : \Sigma \vdash \Gamma$ and derive a contradiction. There are two cases:

1. n is of type 1. Then n is of the form $n(D) : \Sigma \vdash Q$ and Σ doesn't satisfy Q. Therefore n is a false node. By condition Q1 and Q2 from Def. 2.7 it follows that we could construct another tableau without using the ω-rule that has a path leading to a node $n_2(?) : \Sigma' \vdash Q$ s.t. $\Sigma' \leq \Sigma$ and $\forall s \in S.\ \Sigma'(s) \neq \Sigma(s) \ \Rightarrow\ \Sigma(s) = \omega$ and Σ' fails Q. This is a contradiction, because by Lemma 3.1 the node n_2 should be true.

2. If n is of type 4 then because of condition Q1 and Q2 for any $k \in \mathbb{N}$ it is possible to construct another tableau without using the M- and ω-rules s.t. this tableau contains two nodes $n_1(?) : \Sigma_1 \vdash \Gamma \ \ll\ n_2(?) : \Sigma_2 \vdash \Gamma$, $\Sigma_2 \geq \Sigma_1$, $\forall s \in S.\Sigma(s) = \omega \ \Rightarrow\ \Sigma_1(s) \geq k$ and every internal circuit of the path $n_1 \ldots n_2$ has μ-characteristic.

   Let $\sigma = trans(n_1 \ldots n_2)$. By our assumption the run σ^ω starting at Σ_1 satisfies some formula of Γ. Let $\{\Phi_1, \ldots, \Phi_l\}$ be the satisfied formulae. Let $\vec{\sigma}$ be the Parikh-vector of σ. We know that $\vec{\sigma} \geq \vec{0}$. An internal path starting with Φ_1 of the form $\Sigma_1 \vdash \Phi_1 \ldots \Sigma_2 = \Sigma_1 + \vec{\sigma} \vdash \Phi_x \ldots \Sigma_1 + i\vec{\sigma} \vdash \Phi_y \ldots$ which is constructed by certain rules must be periodic. Especially some formula Φ_i must occur infinitely often. Now construct this periodic internal path $\pi = \Sigma_1 \vdash \Phi_i \ldots \Sigma_1 + 1 * m\vec{\sigma} \vdash \Phi_i \ldots \Sigma_1 + j * m\vec{\sigma} \vdash \Phi_i \ldots$. The construction is guided by inductively associating to each pair $\Sigma \vdash \Phi$ a suffix ρ_i of σ^ω s.t. $\rho_i(0) = \Sigma$ and ρ_i satisfies Φ. For the initial pair $\Sigma_1 \vdash \Phi_i$ this is σ^ω itself. Now we define how to select the $(x+1)$-th element $\Sigma' \vdash \Phi'$ and ρ_{x+1}, given the x-th element $\Sigma \vdash \Phi$ and ρ_x.

- If $\Phi = Q$ and the Q-rule is applied, then $\Sigma \vdash \Phi$ is the last node of π. In the original tableau there is a corresponding marking Σ_ω s.t. $\Sigma \leq \Sigma_\omega$ and $\forall s \in S.\Sigma_\omega(s) \neq \Sigma(s) \Rightarrow \Sigma_\omega(s) = \omega \ \wedge \ \Sigma(s) \geq k'$. $k' \in \mathbb{N}$ is finite, but we can choose it arbitrarily high, because we can choose k arbitrarily high and $\vec{\sigma} \geq \vec{0}$. As $\Sigma_\omega \not\models Q$ it follows that $\Sigma \not\models Q$, because of condition Q2 defined in Def. 2.7.
- If $\Phi = \Psi \wedge \Upsilon$ and $\wedge$ is applied to Φ, then $\Sigma' = \Sigma$, Φ' is either Ψ or Υ, according to the choice in the path from n_1 to n_2, and $\rho_{i+1} = \rho_i$.
- If $\Phi = \Psi \vee \Upsilon$ and $\vee$ is applied to Φ, then $\Sigma' = \Sigma$, $\rho_{i+1} = \rho_i$ and

$$\Phi' = \begin{cases} \Psi, & \text{if } \mu - sig(\rho_i, \Psi) \leq \mu - sig(\rho_i, \Upsilon) \\ \Upsilon, & \text{otherwise} \end{cases}$$

- If $\Phi = \odot_A \Psi$ then $\odot$ is applied and $\Phi' = \Psi$, $\rho_{i+1} = \rho_i^{(1)}$ and Σ' is the state corresponding to $\rho_{i+1}(0)$.
- If $\Phi = \sigma Z.\Psi$ and σZ is applied then $\Sigma' = \Sigma$, $\Phi' = \Psi[\sigma Z.\Psi/Z]$ and $\rho_{i+1} = \rho_i$.

There are two possible sub-cases:

(a) π is finite.
Then the last node must be of the form $\Sigma \vdash Q$, and the Q-rule is applied. Therefore no run starting at Σ satisfies Q. This is a contradiction, as the node $\Sigma \vdash Q$ should be true.

(b) π is infinite.
Let Z be the characteristic of π. Then Z is also the characteristic of some internal circuit of $n_1 \ldots n_2$, and therefore a μ-variable. Assign to each element $\Sigma \vdash \Phi$ of π (with corresponding run ρ) a truncated prefix of $\mu - sig(\rho, \Phi)$ by removing all ordinals corresponding to μ-variables lower than Z. 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 where the $\vee$-rule was applied. As we have shown before an infinite number of sequents of the form $\Sigma_1 + j * m\vec{\sigma} \vdash \Phi_i$ for $i = 1, 2, \ldots$ occur in π, s.t. each has the associated run σ^ω. So the associated truncated μ-signatures are the same. This is a contradiction, because the truncated μ-signature should decrease as the variable Z is unfolded between two occurrences of this sequent.

□

Lemma 4.4 *If there is a successful tableau, then* $\Sigma_0 \models \Phi_0$.

Proof Assume that there is a successful tableau τ for $\Sigma_0 \vdash \Phi_0$, but $\Sigma_0 \not\models \Phi_0$. We will derive a contradiction.

Assuming that $\Sigma_0 \not\models \Phi_0$ there must be a run σ_0 starting at Σ_0 s.t. $\sigma_0 \notin \|\Phi_0\|$. We shall use this run to show the existence of an unsuccessful terminal, contradicting the success of τ.

To do this, we first use σ_0 to construct a (possibly infinite) path π' in a tableau τ' constructed without using the special rules (i.e. by the basic rules only). Using this

path we shall then prove the existence of a finite unsuccessful path π in the tableau τ constructed by all rules.

To serve as guide during the construction of the path $\pi' = n_0 n_1 n_2 \ldots$, we inductively associate to each node n_i a suffix ρ_i of σ_0 s.t. the state of n_i is $\rho_i(0)$ and ρ_i fails every formula of n_i. The suffix associated with the root n_0 is σ_0. If n_i is a terminal of type 1, 2, 3 or 4 then (ρ_i, n_i) is the last element of π. Otherwise its successor n_{i+1} and associated suffix ρ_{i+1} are chosen as follows:

- If the $\odot$-rule is applied to n_i, then $\rho_{i+1} = \rho_i^{(1)}$, and n_{i+1} is the child of n_i having $\rho_{i+1}(0)$ as state;
- If the $\wedge$-rule is applied then to n_i, then $\rho_{i+1} = \rho_i$ and n_{i+1} is a child of n_i s.t. the ν-signature of ρ_i is preserved (if ρ fails $\Phi \wedge \Psi$ with ν-signature ξ, then ρ fails either Φ or Ψ with ν-signature ξ).
- If one of the rules $\vee$, Q or σZ is applied then $\rho_{i+1} = \rho_i$ and n_{i+1} is the only child of n_i.

As no special rules are used the labels of all nodes are empty. It follows from Lemma 3.1 that every node of π' is false. There are two cases:

1. π' is infinite.
   As there are only finitely many subformulae of Φ_0, there are only finitely many different sequents Γ in the tableau τ'. So π' must contain an infinite subsequence $n_{m_1}, n_{m_2}, \ldots$ s.t. $n_{m_i}(\{\}) : \Sigma_{m_i} \vdash \Gamma$. Let $\Gamma = \Phi_1, \ldots, \Phi_n$. We assign to each node n_{m_i} a vector of signatures $\bar{x_i} = (\nu - sig(\Phi_1, \rho_{m_i}), \ldots, \nu - sig(\Phi_n, \rho_{m_i}))$. By Dickson's Lemma there are two indices $i \leq j$ s.t. $\bar{x_i} \leq \bar{x_j}$ and $\Sigma_{m_i} \leq \Sigma_{m_j}$. Note that the relation between $\bar{x_i}$ and $\bar{x_j}$ is the pointwise order on vectors, while the order on their components is the lexicographic order.

   Now we prove that every internal circuit of $n_{m_i} \ldots n_{m_j}$ has μ-characteristic.

   Assume there is an internal circuit γ of $n_{m_i} \ldots n_{m_j}$ with ν-characteristic Z. Assign to each element (n_k, Φ_k) of γ a prefix of $\nu - sig(\Phi_k, \rho_k)$ obtained by removing all ordinals corresponding to ν-variables lower than Z. Let T_γ be the sequence of truncated signatures corresponding to γ. We claim that T_γ is non-increasing. Let n_k and n_{k+1} be two consecutive nodes of γ. If $n_k \neq n_{m_j}$ then n_{k+1} is a successor of n_k and the truncated signature cannot increase when moving from n_k to n_{k+1}, because no variable higher than Z is ever unfolded and because of the way the branch is chosen at the $\wedge$-nodes. If $n_k = n_{m_j}$, then $n_{k+1} = n_{m_i+1}$ and since $\bar{x_i} \leq \bar{x_j}$ the truncated signature cannot increase as well.

   Since Z is unfolded somewhere in γ, the last element of T_γ is lexicographically less than the first. This contradicts the assumption that $\bar{x_i} \leq \bar{x_j}$. Therefore Z must be a μ-variable.

   Using these properties we will construct a tableau τ'' by using the basic rules and the ω-rule, but omitting the M-rule. The condition Q1 from Def. 2.7 ensures that τ'' contains two nodes $n_1(\{\}) : \Sigma \vdash \Gamma \ll n_2(\{\}) : \Sigma \vdash \Gamma$, s.t. every internal circuit between them has μ-characteristic. (Σ will possibly contain ωs). There are two cases:

(a) Either the ω-rule has been applied before n_{m_i} and n_1 corresponds to n_{m_i} and n_2 to n_{m_j}.

(b) Or the ω-rule is applied at n_{m_j}. Let π_ω be the path from n_{m_i} to n_{m_j}. Then n_1 corresponds to the modified n_{m_j} and n_2 corresponds to the node that is reached from n_1 via π_ω.

Note that now both n_1 and n_2 have the same marking Σ (which can contain ωs).

Let δ be the Parikh-vector of the sequence of transitions fired between n_1 and n_2. As $\Sigma_{m_i} \leq \Sigma_{m_j}$ we know that $\delta \geq \vec{0}$.

Now we construct the tableau τ''' using all rules and show that it must contain a terminal of type 4 that occurs at the same place as n_2, or even before. Note that n_2 would be a candidate for such a terminal, were it not for the M-rule and termination condition 5. We will start with the tableau τ'' and successively cut out segments of the path leading from the root to n_2, thus obtaining a shorter path leading to a type 4 terminal. We repeat this until termination condition 5 is not satisfied anywhere in this path. Thus we obtain a tableau that could have been constructed from scratch by using all rules. Let n'', n' and n be the nodes that satisfy the conditions of the M-rule and π_1 be the path from the root to n_1 in τ''. Note that no application of the ω-rule takes place between n'' and n, as well as between n_1 and n_2. There are four cases:

(a) $n \ll n_1$
Let α be the path from n' to n. In the path from n to n_1 we can cut out all the subpaths equal to α, thus obtaining a shorter path. n_2 still satisfies the conditions to be a type 4 terminal.

(b) $n' \ll n_1 \ll n$
Let α be the path from n to n_2 and β the path from n_1 to n. Let n_3 be the node that can be reached from n' with the path $\alpha\beta$. It follows that $n_3 \ll n_2$ and n_3 is a type 4 terminal.

(c) $n'' \ll n_1 \ll n'$
Let α be the path from n to n_2 and β the path from n_1 to n. Let n_3 be the node that can be reached from n'' with the path $\alpha\beta$. It follows that $n_3 \ll n_2$ and n_3 is a type 4 terminal.

(d) $n_1 \ll n''$
Let α be the path from n' to n. In the path from n to n_2 we can now cut out all subpaths equal to α. All internal circuits of the path $n_1 \ldots n_2$ still have μ-characteristic, because $Int(n'', n') = Int(n'', n)$. Let δ' be the Parikh-vector of the sequence of transitions fired in α. n_2 is still a type 4 terminal, because n now carries the additional label (δ', n'') and $n'' \gg n_1$.

So τ''' must contain an unsuccessful terminal. Since the tableau for a given root is unique it follows that $\tau''' = \tau$. This is a contradiction, as τ is successful.

2. π' is finite.
Let n be the last node of π'. n cannot be a terminal of type 5, because all labels are empty in τ'. n cannot be a terminal of type 2 or 3, because n is false. So n must be a terminal of type 1 or 4.

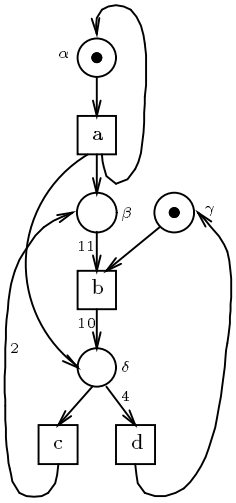
(a) If n is of type 1 then it must have the form $n(\{\}) : \Sigma \vdash Q$ s.t. $\Sigma \notin \mathcal{W}(Q)$. There is a path π in τ corresponding to a subsequence σ_0' of σ_0 s.t. πs last node is $n'(?) : \Sigma' \vdash Q$ and $\exists \Sigma''.\ \Sigma' = \Sigma + \omega\Sigma''$. It follows from condition Q1 in Def. 2.7 that $\Sigma' \notin \mathcal{W}(Q)$. So n' is an unsuccessful node in τ, a contradiction.

(b) If n is of type 4, then we have the same situation as in case 1.

□

5 Example

Figure 1: Does this system satisfy the formula $\nu x.\ \odot_a\ (x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y)))$?



No. As the tableau is quite large we only describe one unsuccessful branch. Σ is the initial marking. $\Sigma = (1,0,1,0)$, $\Sigma_2 = (1,1,1,1)$, $\Sigma_3 = (1,\omega,1,\omega)$, $\Sigma_4 = (1,\omega,0,\omega)$.

$$
\begin{aligned}
\Sigma \vdash \nu x.\ \odot_a\ (x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y))) &\xrightarrow{\sigma Z} \Sigma \vdash \odot_a (x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y)) \\
&\xrightarrow{\odot} \Sigma_2 \vdash x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y)) \\
&\xrightarrow{\wedge+\omega} \Sigma_3 \vdash \nu x.\ \odot_a\ (x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y))) \\
\xrightarrow{\sigma Z} \Sigma_3 \vdash \odot_a (x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y))) &\xrightarrow{\odot} \Sigma_3 \vdash x \wedge \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y)) \\
&\xrightarrow{\wedge} n_1 : \Sigma_3 \vdash \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y)) \xrightarrow{\sigma Z} \Sigma_3 \vdash \odot_b (\nu z.\ \odot_c\ (z \wedge \odot_d y)) \\
&\xrightarrow{\odot} n_2 : \Sigma_4 \vdash \nu z.\ \odot_c\ (z \wedge \odot_d y) \xrightarrow{\sigma Z} \Sigma_4 \vdash \odot_c (z \wedge \odot_d y) \xrightarrow{\odot} \Sigma_4 \vdash z \wedge \odot_d y \\
&\xrightarrow{\wedge} \Sigma_4 \vdash \nu z.\ \odot_c\ (z \wedge \odot_d y) \xrightarrow{\sigma Z} \Sigma_4 \vdash \odot_c (z \wedge \odot_d y) \xrightarrow{\odot} \Sigma_4 \vdash z \wedge \odot_d y \\
&\xrightarrow{\wedge+M} n_3(\{((0,2,0,-1), n_2)\}) : \Sigma_4 \vdash \nu z.\ \odot_c\ (z \wedge \odot_d y) \\
&\xrightarrow{\sigma Z} n_4(\{\}) : \Sigma_4 \vdash \odot_c (z \wedge \odot_d y) \xrightarrow{\odot} n_5(\{\}) : \Sigma_4 \vdash z \wedge \odot_d y \xrightarrow{\wedge} \Sigma_4 \vdash \odot_d y \\
&\xrightarrow{\odot} \Sigma_3 \vdash \mu y.\ \odot_b\ (\nu z.\ \odot_c\ (z \wedge \odot_d y))
\end{aligned}
$$

The last node in this path already occurred earlier at n_1, and all internal paths between them (only one in this case) have μ-characteristic, because the highest unfolded variable y

is a μ-variable. The only problem is that the Parikh-vector of the sequence of transitions fired between them is not positive. It is $b + 3 * c + d = (0, -5, 0, 3)$. Fortunately we have a label at n_3 with an entry marked with n_2, which is below n_1 (and thus we may use it). The question is now if there is a $k \in \mathbb{N}$ s.t. $(0, -5, 0, 3) + k * (0, 2, 0, -1) \geq \vec{0}$? We see that there is one (in this case only 1) solution, $k = 3$. Thus termination-condition 4 is satisfied and the branch is unsuccessful.

It can be verified that even a slight change of the system makes it satisfy the formula. If the arc from β to b is labelled by 13 instead of 11, then no infinite sequence of the form $(bc^n d)^\omega$ is possible, although such sequences of arbitrary length are possible if enough a's are done first. Thus the above construction is impossible. Any infinite path must contain infinitely many a's and in the tableau each a-action is accompanied by an unfolding of the outermost ν-variable x. Therefore the system satisfies the formula.

6 Extensions

The restrictions on the elementary predicates in Def. 2.7 basically amount to the fact that every predicate has the form $P := \{x_1 \leq k_1, x_2 \leq k_2, \ldots, x_n \leq k_n\}$, where $x_1, \ldots, x_n$ are the places in the net and $k_1, \ldots, k_n \in \mathbb{N} \cup \{\omega\}$. A marking Σ satisfies the predicate P iff $\forall i \in \{1, \ldots, n\}.\Sigma(x_i) \leq k_i$.

A possible generalization is to allow conditions of the form $s_i = k_i$ instead of $s_i \leq k_i$. The only problem with this is the application of the ω-rule. We can no longer assume that if a marking Σ fails a predicate P, then every marking $\Sigma' \geq \Sigma$ also fails P. So we can no longer assume that whenever the ω-rule is applicable, an infinite path could be created leading to markings with arbitrary high numbers of tokens in some places. This is because now it might be possible to terminate this path by termination condition 3. The solution is to modify the ω-rule appropriately.

$n_2(D) : \Sigma_2 \vdash \Gamma \longrightarrow n_3(D) : \Sigma_2 + \omega(\Sigma_2 - \Sigma_1) \vdash \Gamma$ if there is a previous node $n_1(?) : \Sigma_1 \vdash \Gamma$ s.t. $n_2 \gg n_1$ and

- $\Sigma_2 \geq \Sigma_1$ and there is a place s s.t. $\Sigma_1(s) < \Sigma_2(s) \neq \omega$ and
- Let $\Sigma'_1, \ldots, \Sigma'_m$ be the states of nodes in the tableau between n_1 and n_2 where the Q-rule is applied and $P_1, \ldots, P_m$ the corresponding predicates. $\forall i \in \{1, \ldots, m\}.\forall k \in \mathbb{N}.\ \Sigma'_i + k(\Sigma_2 - \Sigma_1) \notin \mathcal{W}(P_i)$.

Note that this condition is still decidable for these predicates.

Now it can be seen that the tableau method can be generalized for predicates satisfying the following (weaker) conditions:

P1 $\Sigma \notin \mathcal{W}(Q) \ \Rightarrow\ \Sigma + \omega\Sigma' \notin \mathcal{W}(Q)$

P2 It is decidable for a marking Σ and a vector $\delta \geq \vec{0}$ if there is an $i \in \mathbb{N}$ s.t. $\Sigma + i\delta \in \mathcal{W}(Q)$.

Q2 $\Sigma + \omega\Sigma' \notin \mathcal{W}(Q) \ \Rightarrow\ \exists k \in \mathbb{N}\ \forall k' \geq k.\ (\Sigma + k'\Sigma') \notin \mathcal{W}(Q)$.

7 Conclusion

We have presented a tableau system for model checking Petri nets with a (fairly expressive) fragment of the linear time μ-calculus. It uses the technique of examining internal paths that was first used for finite state systems in [BEM96]. In spite of the restrictions defined in Definition 2.7 this tableau system is still a true generalization of the one for finite systems in [BEM96]. This is due to the fact that the strong nexttime-operator $\bigcirc$ can be expressed by the weak nexttime-operator $\odot$ and a predicate P s.t. $\Sigma \in \mathcal{W}(P) \Leftrightarrow \exists \Sigma \xrightarrow{a} \Sigma'$. As we can model finite systems with Petri nets by assigning a place to each state and a transition to each arc the different states will be incomparable and thus P will satisfy the conditions $Q1$ and $Q2$ from Definition 2.7. Of course this is not the case for infinite state systems.

Even for finite state systems the complexity of the algorithm is exponential [BEM96]. For general Petri nets the method relies on Dickson's lemma for termination and is therefore not primitive recursive.

Although the tableau is sound and complete it is not intended to be used as a decision procedure, but rather as a proof method that could be implemented in the framework of a theorem prover with human interaction.

Acknowledgement I would like to thank Javier Esparza and Angelika Mader for many helpful discussions.

References

[BEM96] J. Bradfield, J. Esparza, and A. Mader. An effective tableau system for the linear time μ-calculus. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of ICALP'96*, number 1099 in LNCS. Springer Verlag, 1996.

[Bra92] J. Bradfield. *Verifying Temporal Properties of Systems.* Birkhauser, 1992.

[Esp] Javier Esparza. Decidability of model checking for infinite-state concurrent systems. To appear in Acta Informatica.

[SW90] C. Stirling and D. Walker. Ccs, liveness, and local model checking in the linear time μ-calculus. In *Proceedings of the First International Workshop on Automatic Verification Methods for Finite State Systems*, number 407 in LNCS, pages 166–178. Springer Verlag, 1990.

[SW91] C. Stirling and D. Walker. Local model checking in the modal μ-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990

342/2/90 A Reinhard Fößmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990

342/3/90 A Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990

342/4/90 A Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode

342/5/90 A Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover

342/6/90 A Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover

342/7/90 A Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual

342/8/90 A Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover

342/9/90 A Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)

342/10/90 A Walter Vogler: Bisimulation and Action Refinement

342/11/90 A Jörg Desel, Javier Esparza: Reachability in Reversible Free- Choice Systems

342/12/90 A Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement

342/13/90 A Rob van Glabbeek: The Linear Time - Branching Time Spectrum

342/14/90 A Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen

342/15/90 A Peter Rossmanith: The Owner Concept for PRAMs

342/16/90 A G. Böckle, S. Trosch: A Simulator for VLIW-Architectures

342/17/90 A P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen

342/18/90 A Christoph Zenger: SPARSE GRIDS

Reihe A

342/19/90 A Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems

342/20/90 A Michael Griebel: A Parallelizable and Vectorizable Multi- Level-Algorithm on Sparse Grids

342/21/90 A V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results

342/22/90 A Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions

342/23/90 A Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement

342/24/90 A Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm

342/25/90 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)

342/26/90 A Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)

342/27/90 A Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems

342/28/90 A Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras

342/29/90 A Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics

342/30/90 A Michael Griebel: Parallel Multigrid Methods on Sparse Grids

342/31/90 A Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits

342/32/90 A Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs

342/33/90 A Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes

342/1/91 A Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?

342/2/91 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets

342/3/91 A Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems

342/4/91 A Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity

Reihe A

342/5/91 A Robert Gold: Dataflow semantics for Petri nets

342/6/91 A A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

342/7/91 A Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions

342/8/91 A Walter Vogler: Generalized OM-Bisimulation

342/9/91 A Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen

342/10/91 A Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System

342/11/91 A Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen

342/12/91 A Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken

342/13/91 A Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage

342/14/91 A Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines

342/15/91 A Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?

342/16/91 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS

342/17/91 A Ulrich Furbach: Answers for disjunctive logic programs

342/18/91 A Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs

342/19/91 A Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme

342/20/91 A M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover

342/21/91 A Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids

342/22/91 A Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems

Reihe A

342/23/91 A Astrid Kiehn: Local and Global Causes

342/24/91 A Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine

342/25/91 A Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition

342/26/91 A Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch

342/27/91 A Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C

342/28/91 A Claus Dendorfer: Funktionale Modellierung eines Postsystems

342/29/91 A Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems

342/30/91 A W. Reisig: Parallel Composition of Liveness

342/31/91 A Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments

342/32/91 A Frank Leßke: On constructive specifications of abstract data types using temporal logic

342/1/92 A L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI

342/2/92 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS

342/2-2/92 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)

342/3/92 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/4/92 A Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS

342/5/92 A Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstüt- zung paralleler und verteilter Programmierung

342/6/92 A Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications

342/7/92 A Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study

342/8/92 A Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement

Reihe A

342/9/92 A Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp

342/10/92 A H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems

342/11/92 A M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks

342/12/92 A Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions

342/13/92 A Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme

342/14/92 A Michael Griebel: Grid– and point–oriented multilevel algorithms

342/15/92 A M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems

342/16/92 A J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme

342/17/92 A Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen

342/18/92 A Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS

342/19/92 A Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries

342/20/92 A Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets

342/21/92 A Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids

342/22/92 A Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken

342/23/92 A Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency

342/24/92 A T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity

342/25/92 A Ekkart Kindler: Invariants, Compositionality and Substitution

342/26/92 A Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs

342/1/93 A M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique

342/2/93 A Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents

Reihe A

342/3/93 A Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments

342/4/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation

342/5/93 A Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals

342/6/93 A Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms

342/7/93 A Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

342/8/93 A Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving

342/9/93 A Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z = f(x,y)): The Analysis and the Parallelization of Algorithms

342/10/93 A Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods

342/11/93 A Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods

342/12/93 A Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination

342/13/93 A Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids

342/14/93 A Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation

342/15/93 A Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms

342/16/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation

342/17/93 A Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib

342/18/93 A Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component

342/19/93 A Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems

Reihe A

342/20/93 A Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism

342/01/94 A Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation

342/02/94 A Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency

342/03/94 A Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router

342/04/94 A Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2

342/05/94 A Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations

342/06/94 A Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

342/07/94 A Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach

342/08/94 A Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls

342/09/94 A Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits

342/10/94 A Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style

342/11/94 A Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus

342/12/94 A Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids

342/13/94 A Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/14/94 A Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware

342/15/94 A M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems

342/16/94 A Gheorghe Ştefănescu: Algebra of Flownomials

342/17/94 A Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

342/18/94 A Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization

Reihe A

342/19/94 A Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method

342/20/94 A Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids

342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law

342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space

342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes

342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems

342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik

342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations

342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications

Reihe A

342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C

342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control

342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes

342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation

342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems

342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries

342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs

342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses

342/05/96 A Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification

342/06/96 A Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components

342/07/96 A Richard Mayr: Some Results on Basic Parallel Processes

342/08/96 A Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht

342/09/96 A P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme

342/10/96 A Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFSLib – A File System for Parallel Programming Environments

342/11/96 A Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions

342/12/96 A Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete

342/13/96 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks

Reihe A

342/14/96 A Giannis Bozas, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project

342/15/96 A Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ-Calculus

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications

342/2/90 B Jörg Desel: On Abstraction of Nets

342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems

342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme

342/1/91 B Barbara Paech1: Concurrency as a Modality

342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox -Anwenderbeschreibung

342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen

342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods

342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis

342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme

342/7/91 B W. Reisig: Concurrent Temporal Logic

342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support

Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support

342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen

342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung

342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick

342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS