TUM

INSTITUT FÜR INFORMATIK

Some Results on Basic Parallel Processes

Richard Mayr

TUM-I9616
März 96

TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM-INFO-03-96-I9616-200/1.-FI

Druck: Fakultät für Mathematik und
Institut für Informatik der
Technischen Universität München

Some Results on Basic Parallel Processes

Richard Mayr*
TU München†

Abstract

Basic Parallel Processes (BPP) are a very natural subclass of the class of CCS processes. They are a simple model for the description of infinite state concurrent systems. BPPs are closely related to communication-free Petri nets, a special class of labeled Petri nets, where every transition has exactly one place in its preset. Unlike for general Petri nets, it is decidable if a BPP and a finite state labeled transition system are weakly bisimulation equivalent. This is the first time that weak bisimulation equivalence to a finite state labeled transition system is proved decidable for a non-trivial model of infinite state concurrent systems.

While language equivalence is undecidable for communication-free nets and so far only non-primitive recursive algorithms exist for deciding strong bisimulation equivalence, in the subcase of one-to-one labeled communication-free nets the strong bisimulation/language equivalence problem is decidable in polynomial time.

For communication-free nets a model checking problem for a weak branching time temporal logic is shown to be PSPACE-complete. Moreover, subproblems of this model checking problem for communication-free nets are complete for the n-th order of the polynomial time hierarchy.

Keywords: Basic Parallel Processes, communication-free Petri nets, bisimulation, model checking.

1 Introduction

Finite state concurrent systems have been extensively studied within the theory of process calculi, often by making use of results from standard formal language theory. For these systems all standard behavioral equivalences are decidable, so many automated tools have been designed for their analysis.

Recently, questions regarding the decidability of process equivalences on several classes of infinite state systems have been studied. As a well-known theorem from formal language theory states that language equivalence is undecidable even for context-free processes [5], stronger notions of equivalence are regarded. These stronger equivalences are designed to take into account the notions of deadlock, livelock and causality. As all the

*Research supported by “Graduiertenkolleg Kooperation und Ressourcenmanagement in verteilten Systemen”, TU München

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

standard behavioral equivalences except strong bisimulation equivalence are undecidable over context-free processes [4], while strong bisimulation equivalence is in fact decidable for context-free processes [9], bisimulation equivalence has become a central notion in concurrency theory. Therefore it is an important question for which classes of processes it is decidable whether two processes are bisimulation equivalent and if there is a decision procedure which is efficient.

One of the most common languages for describing parallel processes is the Calculus of Communicating Systems (CCS) introduced by Milner [8]. In CCS processes are built from a set of atomic actions by the operations of action prefix, sequential composition, parallel composition, nondeterministic choice and process synchronization. There are some very natural subsets of CCS: the Basic Parallel Processes (BBP) that are built by the operations of nondeterministic choice and parallel composition and the Basic Process Algebra (BPA) whose operations are nondeterministic choice and sequential composition. The elements of the BPA are called context free processes, because they correspond naturally to context free grammars. Basic Parallel Processes correspond to communication-free nets, which are the Petri nets where every transition has exactly one place in its preset and the arc from this place to the transition is labeled by 1. A special case of BPP/BPA are the normed BPP/BPA, which are exactly the BPP/BPA-processes that in every reachable state have at least one terminating computation. In the process algebra PA, which is a superset of BPA and BPP, process terms are built by nondeterministic choice, sequential composition and parallel composition.

While bisimulation equivalence is known to be undecidable for CCS [8], it is decidable for BPP and BPA [12]. It is an open problem whether it is decidable for PA. For the deterministic case bisimulation equivalence coincides with language equivalence. In the nondeterministic case language equivalence is undecidable for all these process algebras [12].

There are other important equivalences which respect the intuition that actions of internal communication should not be externally observable. These are weak bisimulation equivalence and weak bisimulation congruence [8]. All these equivalences define a semantics for the processes. Although BPPs are a model for infinite state concurrent systems it is possible that a BPP is semantically equivalent to a finite state system. Section 3 contains a decidability result for this problem for the case of weak bisimulation equivalence.

In section 2 process algebras, labeled transition systems and Basic Parallel Processes are defined and the notion of strong and weak bisimulation equivalence is motivated. Section 3 describes an algorithm that decides if a given BPP and a finite state labeled transition system are weakly bisimulation equivalent. In section 4 a polynomial time algorithm is described that decides if two one-to-one labeled communication-free nets are strongly bisimulation equivalent. In section 5 it is shown that model checking communication-free nets with a weak branching time temporal logic is a PSPACE-complete problem. The paper closes with some remarks on open problems and on related work.

2 Preliminaries

2.1 Strong and weak bisimulation

The following notions of labeled transition system (for short: LTS), strong and weak bisimulation are important for the definition of the semantics of Petri nets and process algebras.

Definition 2.1 A *labeled transition system* T over a set of Actions Act consists of a (possibly infinite) set of states S and a binary relation $\xrightarrow{a} \subseteq S \times S$ 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 $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \ldots$ or a finite sequence $s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$ such that s_n has no successors. A *run* is a path that starts at the initial state. The language of a rooted LTS T is the set of all finite and infinite sequences of actions obtained from the runs of T.

Definition 2.2 A binary relation R over the states of an LTS is a *strong bisimulation* (often simply called bisimulation) iff

$$\begin{aligned} \forall s_1, s_2 \in S \text{ s.t.} s_1 R s_2\ \forall a \in Act.\quad & (s_1 \xrightarrow{a} s_1' \ \Rightarrow \exists s_2 \xrightarrow{a} s_2'.\ s_1' R s_2') \ \wedge \\ & (s_2 \xrightarrow{a} s_2' \ \Rightarrow \exists s_1 \xrightarrow{a} s_1'.\ s_1' R s_2') \end{aligned}$$

Two states s_1 and s_2 are *strongly bisimulation equivalent* if there is a strong bisimulation R such that $s_1 R s_2$. This definition can be extended to states in different transition systems by putting them 'side by side' and considering them as a single transition system. It is not difficult to see that there always exists a largest strong bisimulation which is an equivalence relation and is called **strong bisimulation equivalence**. It is denoted by $\sim$.

There is another elegant characterization: Let $\theta_a(s) := \{s' \mid s \xrightarrow{a} s'\}$. An equivalence relation R on S is a strong bisimulation iff

$$\forall s \in S\ \forall a \in Act.\ \theta_a([s]_R) \subset [\theta_a(s)]_R$$

In process algebras processes are described by process terms. There are syntactical definitions of how these terms are built from a set of atomic actions and some operators. The dynamical behavior of the processes is described by transition rules of the form $t \xrightarrow{a} t'$, where t and t' are processes and a is an atomic action. Basically this means that process t can perform the action a and become t'. So a process expression determines a labeled transition system (LTS) whose states are labeled with process expressions and whose arcs are labeled with atomic actions. The semantics of the processes is given by defining an equivalence relation over the term algebra. The equivalence classes then represent the intended processes. It follows that the dynamical rules describe unambiguously the dynamics of the quotient algebra, only if the chosen equivalence on the term algebra is a bisimulation. This is a main reason why bisimulation equivalence is the preferred choice for process equivalence.

However strong bisimulation equivalence is sometimes to strict. Processes can contain silent internal actions (labeled by τ) which should not be externally visible. Therefore another equivalence called weak bisimulation equivalence is defined that treats these τ-actions accordingly.

Definition 2.3 Let $\overset{a}{\Rightarrow} := (\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^*$ for every $a \in Act$ and

$$\overset{\hat{a}}{\Rightarrow} := \begin{cases} \overset{a}{\Rightarrow}, & \text{if } a \neq \tau \\ (\xrightarrow{\tau})^*, & \text{if } a = \tau \end{cases}$$

A binary relation R over the states of an LTS is a *weak bisimulation* if

$$\begin{aligned} \forall s_1, s_2 \in S \text{ s.t.} s_1 R s_2 \; \forall a \in Act. \;\; & (s_1 \xrightarrow{a} s_1' \;\Rightarrow \exists s_2 \overset{\hat{a}}{\Rightarrow} s_2'.\ s_1' R s_2') \wedge \\ & (s_2 \xrightarrow{a} s_2' \;\Rightarrow \exists s_1 \overset{\hat{a}}{\Rightarrow} s_1'.\ s_1' R s_2') \end{aligned}$$

Two states s_1 and s_2 are *weakly bisimulation equivalent* if there is a weak bisimulation R such that $s_1 R s_2$. Again this can be extended to states in different transition systems. There always exists a largest weak bisimulation which is an equivalence relation and is called **weak bisimulation equivalence**. It is denoted by $\approx$. It is clear that $\sim \subseteq \approx$ for every LTS.

2.2 Bisimulation games

Another approach to bisimulation is via bisimulation games. A game consists of a triple $(\mathcal{T}, s_1, s_2)$, where $\mathcal{T}$ is an LTS and s_1, s_2 are states, and two players A (Attacker) and D (Defender). It is the aim of player A to prove that s_1 and s_2 are not bisimulation equivalent, while player D attempts to frustrate this. A turn in the game goes like this:

1. Player A chooses an $i \in \{1, 2\}$, an action a and a move $s_i \xrightarrow{a} s_i'$. If it is impossible to make any move in s_1 or s_2 then player D wins.

2. Then player D takes the $j \in \{1, 2\}$ s.t. $i \neq j$ and tries to imitate the move, i.e. find a s_j' s.t. $s_j \xrightarrow{a} s_j'$. If this is impossible then player A wins.

After this turn the resulting state is $(\mathcal{T}, s_1', s_2')$. Player D is the winner of any infinite game.

s_1 and s_2 are *strongly bisimulation equivalent up to n* if there is a strategy for player D to defend (i.e. prevent player A from winning) for at least n turns. This is denoted as $s_1 \sim_n s_2$. It is easy to see that $\sim_n$ is an equivalence relation for every $n \in \mathbb{N}$. If the graph of $\mathcal{T}$ is finitely branching, then $s_1 \sim s_2 \Leftrightarrow \forall n \in \mathbb{N}. s_1 \sim_n s_2$, because in this case $\sim = \bigcap_{n=1}^{\infty} \sim_n$.

Weak bisimulation up to n (denoted by $\approx_n$) is defined in the same way, except that player D makes moves $s_j \overset{\hat{a}}{\Rightarrow} s_j'$ instead of $s_j \xrightarrow{a} s_j'$. It follows immediately that $\sim_n \subset \approx_n$. Unlike for strong bisimulation the relation $\approx_n$ is not an equivalence relation in general. Also $\approx \neq \bigcap_{n=1}^{\infty} \approx_n$, because $\bigcap_{n=1}^{\infty} \approx_n \not\subset \approx$. The other inclusion $\approx \subset \bigcap_{n=1}^{\infty} \approx_n$ holds however.

2.3 Basic Parallel Processes

The Basic Parallel Processes (BPP) were introduced by Christensen in his Ph.D. dissertation [1] as a very natural subclass of the class of CCS processes [8]. They are a simple model of infinite state concurrent systems. Assume a countably infinite set of atomic actions $Act = \{a, b, c, \ldots\}$ and a countably infinite set of process variables

$Var = \{X, Y, Z, \ldots\}$. The class of BPP expressions is defined by the following abstract syntax [1, 10]:

$$\begin{array}{rcll} E & ::= & 0 & \text{inaction} \\ & & X & \text{process variable} \\ & & aE & \text{action prefix} \\ & & E + E & \text{choice} \\ & & E \| E & \text{merge (parallel composition)} \end{array}$$

$\alpha, \beta, \gamma, \ldots$ denote merges of process variables. A BPP is defined by a family of recursive equations

$$\{X_i := E_i \mid 1 \leq i \leq n\}$$

where the X_i are distinct and the E_i are BPP expressions at most containing the variables $\{X_1, \ldots, X_n\}$. It will be assumed that every variable occurrence in the E_i is *guarded*, i.e. appears within the scope of an action prefix. The variable X_1 is singled out as the *leading variable* and and $X_1 = E_1$ is called the *leading equation.* Any finite family of BPP equations determines a labeled transition system. For every $a \in Act$ the transition relation $\xrightarrow{a}$ is the least relation satisfying the following inference rules:

$$aE \xrightarrow{a} E \quad \frac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'} \quad \frac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F} \quad \frac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'}(X := E) \quad \frac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'} \quad \frac{F \xrightarrow{a} F'}{E\|F \xrightarrow{a} E\|F'}$$

Remark 2.4 *BPP processes generate finitely branching transition graphs, i.e.* $\{F \mid E \xrightarrow{a} F\}$ *is finite for each* E *and* a*. This would not be true if unguarded expressions were allowed. For example, the process* $X := a + a\|X$ *generates an infinitely branching transition graph.*

A BPP is in *normal form*, if every expression E_i at the right hand side of an equation is of the form $a_1\alpha_1 + \ldots + a_n\alpha_n$. It is shown in [1] that every BPP is strongly bisimulation equivalent to a BPP in normal form (i.e. the leading variables are strongly bisimulation equivalent).

Definition 2.5 A *Basic Parallel Process Algebra* is described by a pair $\langle \widehat{Var}, \Delta \rangle$, where

$$\widehat{Var} := \{X_1^{m_1} \cdots X_n^{m_n} \mid m_1, \ldots, m_n \in \mathbb{N}\}$$

is the commutative algebra generated by Var, whose elements are finite multisets of process variables. Such a multiset represents a BPP in normal form and the multiset union corresponds to the merge operator. Δ is a set of transition rules of the form $X \xrightarrow{a} \alpha$, where $\alpha \in \widehat{Var}$. These transition rules are constructed from the defining equations. For every definition $X := a_1\alpha_1 + \ldots + a_n\alpha_n$ transition rules $X \xrightarrow{a} \alpha_1, \ldots, X \xrightarrow{a} \alpha_n$ are introduced. These rules determine a more general *transition relation* by

$$X\beta \xrightarrow{a} \alpha\beta$$

if $X \xrightarrow{a} \alpha$ is a transition rule.

So a BPP in normal form can be translated into a labeled Petri net by introducing a place for each process variable and a transition for each transition rule. For a rule $X \xrightarrow{a} Y_1^{m_1} \cdots Y_n^{m_n}$ introduce a transition t labeled by a, an arc labeled with 1 leading from place X to t and arcs labeled by n_i leading from t to places Y_i. It is important to note that in these nets every transition has exactly one input place with an arc labeled by 1. Such Petri nets will be called *communication-free nets*. Communication-free nets will be denoted by N and their markings by Σ. Translating a communication-free net N back into a BPP-algebra is analogous. Any marking Σ of N can then be translated into a BPP in this algebra. So there is a one-to-one correspondence between BPP-algebras and communication-free nets, as well as between elements of the algebra and markings of the net. Therefore they can be used interchangeably.

For labeled Petri nets there is a labeling function $L : T \to Act$ that assigns actions to the transitions. The labeling function L is extended to sequences of transitions in the standard way. If a sequence of transitions σ is firable at a marking Σ and leads to a new marking Σ' this is denoted by $\Sigma \xrightarrow{\sigma} \Sigma'$. A sequence σ' is called a *subsequence* of σ if its Parikh-vector is smaller (componentwise) than the one of σ. In this case σ is called a *supersequence* of σ'.

Definition 2.6 For a Petri net $N = (S, T, W)$ with $W : T \times S \cup S \times T \to \mathbb{N}$ the subnet generated by a subset of places $S' \subseteq S$ is defined as the net

$$N' := (S', {}^{\cdot}S' \cup S'^{\cdot}, W_{|({}^{\cdot}S' \times S' \cup S' \times S'^{\cdot})})$$

The definition of a subnet generated by a subset of transitions $T' \subseteq T$ is analogous.

3 Weak bisimulation equivalence between a BPP and a finite state labeled transition system

Jančar [7] proved that strong bisimulation equivalence is undecidable for general Petri nets, while Milner [8] showed that it is undecidable for CCS. It is however decidable, if a Petri net is strongly bisimulation equivalent to a finite state LTS [6, 7]. For BPP the decidability of strong bisimulation equivalence was proved by Hirshfeld, Christensen and Moller [10]. The subclass of normed BPP consists of those BPP which in every reachable state have a terminating computation. For normed context free processes strong bisimulation equivalence is decidable in polynomial time [12]. Also for normed BPP strong bisimulation equivalence is decidable in polynomial time [11]. On the other hand language equivalence is undecidable for BPP [12].

All these equivalence relations define a semantics for the processes. A process with an infinite state space can still be semantically equivalent to a finite state process. The question is now for which models of infinite state concurrent systems and which semantic equivalences it is decidable if an infinite state process and a finite state system are equivalent. It is clear that if strong bisimulation equivalence is undecidable for a system, then weak bisimulation equivalence is undecidable as well. Although it is decidable if a general Petri net is strongly bisimulation equivalent to a finite state LTS [6], the same problem is undecidable for weak bisimulation equivalence [7]. It will be shown now that it is decidable if a BPP and a finite state LTS are weakly bisimulation equivalent.

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

Lemma 3.1 (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).*

Unlike for CCS, weak bisimulation equivalence is a congruence for BPP.

Lemma 3.2 *Weak bisimulation equivalence is a congruence for BPP, i.e.* $\alpha \approx \beta \Rightarrow \alpha\gamma \approx \beta\gamma$.

Lemma 3.3 *Let* R *be an LTS with* m *states. Let* $r_1, r_2 \in R$ *be states of* R. *Then*

$$r_1 \approx r_2 \;\Leftrightarrow\; r_1 \approx_{m^2-m} r_2$$

Proof As R has a finite number of states the relation $\overset{\hat{a}}{\Rightarrow}$ is finitely branching for any $a \in Act$. Therefore $r_1 \approx r_2 \Leftrightarrow \forall n \in \mathbb{N}. r_1 \approx_n r_2$. So if $r_1 \not\approx r_2$ then there must be an $i \in \mathbb{N}$ and a strategy for player A, such that he can win every bisimulation game in $\leq i$ steps. A bisimulation game is described by a sequence of pairs $(r, r') \in R \times R$. There are at most $m^2 - m$ different such pairs (r, r') with $r \neq r'$. So there must be such an i with $i \leq m^2 - m$, and therefore $r_1 \not\approx r_2 \Rightarrow r_1 \not\approx_{m^2-m} r_2$. The other direction is trivial. □

Definition 3.4 The *size* of a communication-free net $N = (S, T, W)$ is the space needed to describe it. This means that if n is the size of N then N has $\leq n$ places ($|S| \leq n$), $\leq n$ transitions ($|T| \leq n$) and $\forall t \in T\ \forall s \in S. W(t, s) \leq 2^n$, where W is the weight function that assigns weights to the arcs in the net. The size of a marking of the net is defined as the number of tokens. The size of a BPP-algebra is the size of the corresponding communication-free net. The size of a BPP is the size of the corresponding marking of the communication-free net.

Definition 3.5 In a communication-free net tokens can move freely through the net, because every transition has only one place in its preset and the arc leading from this place to the transition is labeled by 1. When a transition fires it takes a token form the place in its preset and puts some tokens on the places in its postset. One can freely choose any of these tokens and call it the *continuation* of the original token. All the others will be called *spin-offs*. Assume a firing sequence σ. A subsequence σ' of σ that does nothing but move a token back to its original place, generating some spin-offs on the way is called a *cycle*. When a cycle is possible it can be repeated an arbitrary number of times, because the resulting marking is bigger than the original one. A cycle does not change a marking, except that it generates some new tokens (the spin-offs).

Lemma 3.6 *Let* (N, Σ) *be a communication-free net with marking* Σ. *Let* n *be the size of the net* N *and* x *the number of tokens in* Σ. *Let* σ *be a firing sequence starting in* Σ *and not containing any cycle* σ' *as subsequence. Then* $length(\sigma) \leq x\frac{2^{n^2-n}-1}{2^n-1}$.

Proof Any non-cyclic path in N has a length of at most $n - 1$. As there are no cycles a token can move at most $n - 1$ steps. The firing of a transition increases the number of tokens in the net by at most $2^n - 1$. So the sequence σ has a maximal length of

$$\sum_{i=0}^{n-2} x * (2^n)^i \;=\; x\frac{1-(2^n)^{n-1}}{1-2^n} = x\frac{2^{n^2-n}-1}{2^n-1}$$

□

Definition 3.7 Let N be a Petri net and S the set of its places. For every $x \in \mathbb{N}$ the relation $\leq_x$ on the set of markings of N is defined by

$$\begin{array}{lll} \Sigma \leq_x \Sigma' & :\Leftrightarrow & \Sigma \leq \Sigma' \qquad\qquad\qquad\qquad\quad \wedge \\ & & \forall s \in S.\ \Sigma(s) \neq \Sigma'(s) \Rightarrow \Sigma(s) \geq x \end{array}$$

For every x the relation $\leq_x$ is a partial order.

Lemma 3.8 *Let N be a communication-free net with n places and two markings Σ_1 and Σ_2 such that $\Sigma_1 \leq_x \Sigma_2$ for an $x \geq n$. Then for any sequence $\Sigma_2 \xrightarrow{\sigma} \Sigma_2'$ s.t. $L(\sigma) \in \tau^* a \tau^*$ there is a subsequence σ' s.t. $L(\sigma') \in \tau^* a \tau^*$, $\Sigma_1 \xrightarrow{\sigma'} \Sigma_1'$ and $\Sigma_1' \leq_{[x/n]} \Sigma_2'$.*

Proof The only difference between σ and the subsequence σ' are the moves of tokens starting in places s s.t. $\Sigma_1(s) \neq \Sigma_2(s)$. For such places $\Sigma_1(s) \geq x$. By a sequence of τ-moves in σ a token starting in s is either moved to a place s' (possibly doing cycles on the way, and generating spin-offs) or finally vanishes (by a transition that has no places in its postset). In σ' some, but not necessarily all (at most $[x/n]$) of these moves are imitated. As $\Sigma_1(s) \geq x$ there are enough tokens to move up to $[x/n]$ tokens to any place s' reachable from s, if necessary. If $\Sigma_2'(s) \geq [x/n]$ then keep $[x/n]$ tokens on s, otherwise move further tokens away, s.t. $\Sigma_2'(s) = \Sigma_1'(s)$. All the cycles in σ are imitated in σ'. This is no problem, since cycles just generate new tokens. The single a-move is also imitated. So one gets a Σ_1' that on any place either has the same number of tokens as Σ_2', or at least $[x/n]$. Thus $\Sigma_1' \leq_{[x/n]} \Sigma_2'$. □

Lemma 3.9 *Let N be a communication-free net with n places and two markings Σ_1 and Σ_2 such that $\Sigma_1 \leq_x \Sigma_2$ for an $x \geq n$. Then for any sequence $\Sigma_1 \xrightarrow{\sigma} \Sigma_1'$ s.t. $L(\sigma) \in \tau^* a \tau^*$ there is a supersequence σ' of σ s.t. $L(\sigma') \in \tau^* a \tau^*$, $\Sigma_2 \xrightarrow{\sigma'} \Sigma_2'$ and $\Sigma_1' \leq_{[x/n]} \Sigma_2'$.*

Proof Again the only difference between σ and σ' are the moves of tokens starting in places s s.t. $\Sigma_1(s) \neq \Sigma_2(s)$. For such places $\Sigma_1(s) \geq x$. By a sequence of τ-moves in σ a token starting in s is either moved to a place s' (possibly doing cycles on the way, and generating spin-offs) or finally vanishes (by a transition which has no places in its postset). As $\Sigma_1(s) \geq x$ at least one of the following conditions must be satisfied:

1. There is a path in the net going only through τ-transitions that leads from s to some place s' such that in σ tokens are moved from s to s' s.t. $\Sigma_1'(s') \geq [x/n]$. (Note that it is possible that $s = s'$).

2. There is a path in the net going only through τ-transitions that have only one place in their postset leading from s to a τ-transition t with $t\cdot = \{\}$. (Tokens on s can vanish by a sequence of τ-moves).

In σ' first imitate all the moves of σ. If $\Sigma_1'(s) < [x/n]$ then do the following, depending on which of the conditions is satisfied.

1. If the first condition is satisfied then track the moves of a token that moves from s to s' in σ. Imitate these moves in σ' for the all extra tokens on place s in Σ_2 until the same number of tokens on s is reached as in Σ_1'. The result may be that $\Sigma_2'(s') \gg \Sigma_1'(s')$, but still $\Sigma_1'(s') \geq [x/n]$.

2. In this case let the extra tokens on s in the marking Σ_2 vanish by sequences of τ-moves as described in the second condition, and get $\Sigma_2'(s) = \Sigma_1'(s)$.

Adding these extra τ-moves to σ yields a σ' s.t. σ' is a supersequence of σ, $\Sigma_2 \xrightarrow{\sigma'} \Sigma_2'$ and $\Sigma_2' \geq_{[x/n]} \Sigma_1'$. □

Remark: Lemma 3.8 and Lemma 3.9 do not hold for general Petri nets, not even for Petri nets where all arcs are labeled by 1.

The relation $\bigcup_{i=0}^{n}(\xrightarrow{\tau})^i$ is also denoted by $\xrightarrow{\tau\leq n}$. Let

$$\Lambda(\Sigma, a, l) := \begin{cases} \{\tilde{\Sigma} \mid \Sigma \xrightarrow{\tau\leq l}\xrightarrow{a}\xrightarrow{\tau\leq l} \tilde{\Sigma}\}, & \text{if } a \neq \tau \\ \{\tilde{\Sigma} \mid \Sigma \xrightarrow{\tau\leq 2l+1} \tilde{\Sigma}\}, & \text{if } a = \tau \end{cases}$$

Lemma 3.10 *Let N be a communication-free net and Σ a marking of N. Let R be a finite state system and $r \in R$ a state of R. Let n be the size of the net N, x the number of tokens in Σ and m the number of states in R. There is a function $l : \mathbb{N}^3 \rightarrow \mathbb{N}$ such that*

$$\Sigma \approx r \ \wedge \ r \xrightarrow{a} r' \ \Rightarrow \ \exists \tilde{\Sigma} \in \Lambda(\Sigma, a, l(n, m, x)) \ s.t. \ \tilde{\Sigma} \approx r'$$

Proof From $r \xrightarrow{a} r'$ and $\Sigma \approx r$ it follows that there exists a Σ' s.t. $\Sigma \overset{\hat{a}}{\Rightarrow} \Sigma'$ and $\Sigma' \approx r'$. Let σ be the sequence leading form Σ to Σ'. It is possible to construct a subsequence $\tilde{\sigma}$ that is exactly the same as σ, except that it possibly contains fewer cycles. So one obtains a new marking $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$ such that $\tilde{\Sigma} \leq_{n^{m^2-m}} \Sigma'$ (see Def. 3.7).

What is the maximal number of cycles in σ' that are needed to reach such a $\tilde{\Sigma}$? Cycles just generate new tokens, and at most n^{m^2-m} new tokens need be produced per place in N. So at most $n * n^{m^2-m} = n^{m^2-m+1}$ cycles are needed in σ'. In every cycle $\leq n$ transitions are fired, so $\leq (2^n - 1) * n^{m^2-m+1}$ new tokens are produced. So at most $x + (2^n - 1) * n^{m^2-m+1}$ tokens are in the net for moves without cycles. By Lemma 3.6 at most

$$(x + (2^n - 1) * n^{m^2-m+1}) * \frac{2^{n^2-n} - 1}{2^n - 1}$$

non-cyclic moves can be made. This does not necessarily mean that the cycles are done first, and the non-cyclic moves afterwards. Moves belonging to cycles and non-cyclic moves can occur in any order.

So altogether at most

$$l(n, m, x) := n^{m^2-m+2} + (x + (2^n - 1) * n^{m^2-m+1}) * \frac{2^{n^2-n} - 1}{2^n - 1}$$

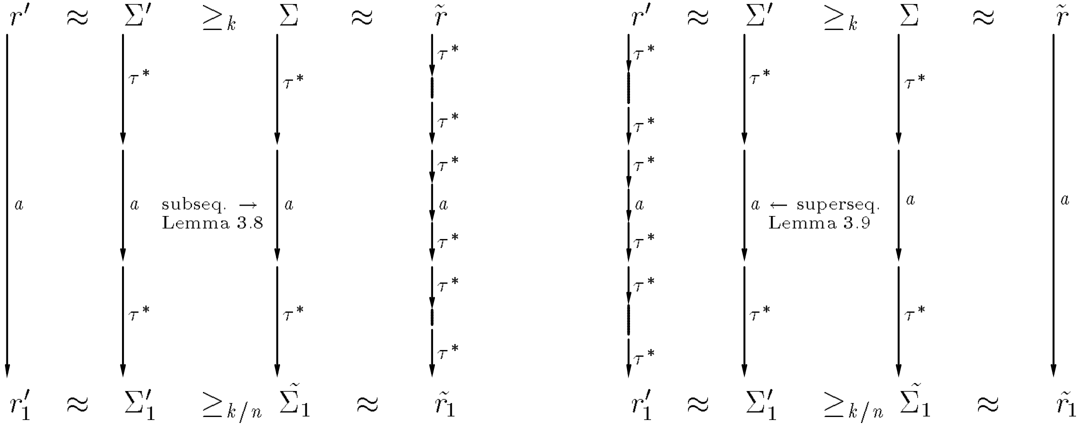
moves need to be made in a $\tilde{\sigma}$ to reach a $\tilde{\Sigma}$ with $\tilde{\Sigma} \leq_{n^{m^2-m}} \Sigma'$.

It remains to prove that $\tilde{\Sigma} \approx r'$. As $\Sigma \approx r$ and $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$ there must be a $\tilde{r} \in R$ s.t. $\tilde{\Sigma} \approx \tilde{r}$. Because of Lemma 3.3 it now suffices to prove that $r' \approx_{m^2-m} \tilde{r}$. So it suffices to prove that in the bisimulation game player D can defend against any attack for at least $m^2 - m$ steps. By Lemma 3.8 and Lemma 3.9 Figure 1 shows a strategy for player D. □

Definition 3.11 Let A be a finite set of pairs of the form (r, α), where r is a state in a finite state system R with m states and α an element of a BPP-algebra of size n. Let $l : \mathbb{N}^3 \rightarrow \mathbb{N}$ be the function from Lemma 3.10.

A set A' is a *weak expansion* of A if:

Figure 1: **Strategy for player D (Defender) to defend for at least $m^2 - m$ steps**


$r' \approx \Sigma' \geq_k \tilde{\Sigma} \approx \tilde{r}$
τ^*
a
subseq. →
Lemma 3.8
$r'_1 \approx \Sigma'_1 \geq_{k/n} \tilde{\Sigma}_1 \approx \tilde{r}_1$
$r' \approx \Sigma' \geq_k \tilde{\Sigma} \approx \tilde{r}$
a ← superseq.
Lemma 3.9
$r'_1 \approx \Sigma'_1 \geq_{k/n} \tilde{\Sigma}_1 \approx \tilde{r}_1$

- For every pair $(r, \alpha) \in A$ and every step $r \xrightarrow{a} r'$ there is a step $\alpha \overset{\hat{a}}{\Rightarrow} \alpha'$ with $(r', \alpha') \in A'$.

- For every pair $(r, \alpha) \in A$ and every step $\alpha \xrightarrow{a} \alpha'$ there is a step $r \overset{\hat{a}}{\Rightarrow} r'$ with $(r', \alpha') \in A'$.

- A' is minimal; no proper subset of A' satisfies these two properties.

A' is a *bounded weak expansion* of A if:

- For every pair $(r, \alpha) \in A$ and every step $r \xrightarrow{a} r'$ there is an $\alpha' \in \Lambda(\alpha, a, l(n, m, x))$ with $(r', \alpha') \in A'$, where x is the size of α.

- For every pair $(r, \alpha) \in A$ and every step $\alpha \xrightarrow{a} \alpha'$ there is an $r' \in \Lambda(r, a, m-1)$ with $(r', \alpha') \in A'$. Note that unlike in the previous case this is no restriction, because R has only m states.

- A' is minimal; no proper subset of A' satisfies these two properties.

The bounded weak expansions of A are denoted by $bwexp(A)$. Note that $bwexp(A)$ is finite if A is finite.

Lemma 3.12 *Let A be a finite set of pairs of the form (r, α), where r is a state in a finite state system and α is a BPP and $r \approx \alpha$. Then there is a bounded weak expansion $A' \in bwexp(A)$ s.t. $\forall (r', \alpha') \in A'. r' \approx \alpha'$.*

Proof directly from Definition 3.11 and Lemma 3.10, because of the one-to-one correspondence between BPPs and communication-free nets. □

Theorem 3.13 *It is decidable if a BPP X and a finite state transition system R with initial state r_1 are weakly bisimulation equivalent.*

Proof The general outline of the proof has some similarities to Hirshfelds proof of the decidability of strong bisimulation equivalence for BPP [10].

There is a BPP algebra $A_1 := \langle Var_1, \Delta_1 \rangle$ s.t. X is the leading variable. Let m be the number of states in R and m' the number of arcs. Construct a BPP algebra $A_2 := \langle Var_2, \Delta_2 \rangle$ for R. Var_2 is a set of m new variables $Y_1, \ldots, Y_m$ s.t. $Var_1 \cap Var_2 = \{\}$.

Each of these variables Y_i represents a state r_i in R. For every arc $r_i \xrightarrow{a} r_j$ in R there is a rule $Y_i \xrightarrow{a} Y_j$ in Δ_2. Y_1 is the leading variable, corresponding to the initial state r_1 of R. Define a new BPP algebra $A_3 = \langle Var_3, \Delta_3 \rangle := \langle Var_1 \cup Var_2, \Delta_1 \cup \Delta_2 \rangle$ with the leading variable X. Let n be the size of the BPP algebra A_3.

Now construct a modified bounded weak bisimulation tree, which will be finite. The nodes in the tree are labeled by finite sets of pairs of the form (Y_i, α) s.t. $Y_i \in Var_2$ and α is a BPP in A_3, i.e. a merge of process variables from Var_3. The root node is labeled by the singleton set $\{(Y_1, X)\}$. The sons of a node are constructed as follows:

For every pair (Y_i, α) in the node there is a finite number of steps, which will be called *questions*. Questions can be of two forms:

- $\alpha \xrightarrow{a} \alpha'$. There are $\leq n$ different questions of this form. This is because the communication-free net corresponding to A_3 has at most n transitions.

- $Y_i \xrightarrow{a} Y_j$. There are $\leq m'$ different ones.

An *answer* to a question is a step $\overset{\hat{a}}{\Rightarrow}$ in the other component of the pair. An answer $Y_i \overset{\hat{a}}{\Rightarrow} Y_j$ to a question $\alpha \xrightarrow{a} \alpha'$ is called *correct answer* if $Y_i \approx \alpha$ and $Y_j \approx \alpha'$. For questions $Y_i \xrightarrow{a} Y_j$ it is analogous.

- For questions of the first type there are $\leq m$ possible different answers $Y_i \overset{\hat{a}}{\Rightarrow} Y_j$. If there is a correct answer then it must be among them.

- For questions of the second type there may be an infinite number of different answers, but Lemma 3.10 yields that if any correct answer exists, then there must be a correct answer $\alpha \overset{\hat{a}}{\Rightarrow} \alpha'$ s.t. $\alpha' \in \Lambda(\alpha, a, l(n, m, x))$, where x is the size of α. So only regard the finite number of answers in $\Lambda(\alpha, a, l(n, m, x))$.

The sons of a node are labeled by the different bounded weak expansions of the set of pairs at the node. A bounded weak expansion represents a possible set of answers to all questions. Lemma 3.10 yields that if a set of correct answers to all questions exists then it is among the bounded weak expansions. As there are only finitely many of them the tree is finitely branching.

Now the newly constructed son-nodes are modified by the following procedure: One can assume that all ancestor-nodes have already been modified. Let (Y_i, α) be a pair in the son-node.

1. If α is a merge of process variables from Var_2 then α represents a finite state system. It follows that it can be decided if $Y_i \approx \alpha$.
   - If $Y_i \approx \alpha$ then remove the pair (Y_i, α) from the son-node.
   - If $Y_i \not\approx \alpha$ then this branch of the modified bounded weak bisimulation tree has failed and is not developed further.

2. Otherwise α contains at least one process variable from Var_1. If the pair (Y_i, α) dominates some pair in an ancestor-node, i.e. $\alpha = \alpha_1\alpha_2$ and (Y_i, α_1) occurs in an ancestor-node, then replace the pair (Y_i, α) by $(Y_i, Y_i\alpha_2)$. If this pair still dominates a pair in an ancestor-node then repeat this step. As the ancestor-node has already been modified and still contains the pair (Y_i, α_1), α_1 must contain at

least one variable from Var_1. So the number of variables form Var_1 in the second component decreases with every step. Finally either a pair is reached that does not dominate a pair occurring in an ancestor-node or no variable from Var_1 is left in the second component and case 1 applies.

So the constructed tree is finitely branching and no pair in a node dominates a pair that occurs in an ancestor-node. By Dickson's Lemma (3.1) the tree is finite.

A successful branch is one that ends with a leaf that is labeled by the empty set. Now X is weakly bisimulation equivalent to Y_1 iff there is a successful branch in the modified bounded weak bisimulation tree whose root is labeled with the singleton set $\{(X, Y_1)\}$.

⇒ If $X \approx Y_1$ then by Lemma 3.12 there is a path $\{(X, Y_1)\} = A_1, A_2, \ldots$ in the tree s.t. $\forall i. A_{i+1} = mod(B_{i+1}) \wedge B_{i+1} \in bwexp(A_i)$, where the function mod describes the modifications of newly constructed nodes that were described earlier and $\forall i \forall (Y, \alpha) \in A_i . Y \approx \alpha$. As by Lemma 3.2 the relation $\approx$ is a congruence for BPP the modifications by mod do not change this property. (If $Y_i \approx \alpha_1\alpha_2$ and $Y_i \approx \alpha_1$ then $Y_i\alpha_2 \approx \alpha_1\alpha_2 \approx Y_i$.) So by the finiteness of the tree the sequence must be finite and end with the empty set. Thus there is a successful branch.

⇐ If there is a successful branch, then the smallest congruence containing all the pairs from the nodes of the path from the root to the successful leaf is a weak bisimulation. Again this is due to the fact that $\approx$ is a congruence for BPP and thus $Y_i \approx \alpha_1 \wedge Y_i \approx Y_i\alpha_2 \Rightarrow \alpha_1\alpha_2 \approx Y_i\alpha_2 \approx Y_i$. This congruence contains (X, Y_1) and therefore $X \approx Y_1$.

As the modified weak bisimulation tree is finite and can be effectively constructed it can be decided if it has a successful branch. □

Remark: The proof of termination of the algorithm relies on Dickson's lemma, so the algorithm is not primitive recursive.

4 Deciding strong bisimulation equivalence for one-to-one labeled communication-free nets

One-to-one labeled Petri nets are nets where the labeling function that assigns actions to the transitions is injective. This means that every transition is uniquely determined by its label. For one-to-one labeled Petri nets strong bisimulation equivalence coincides with language equivalence. Jančar [6, 7] showed that for general one-to-one labeled Petri nets this problem is decidable but has the same complexity as the reachability problem, which means that it is EXPSPACE-hard. For one-to-one labeled communication-free nets the situation is different. While for these nets the reachability problem is NP-complete ([3] or as a corollary from Theorem 5.6), strong bisimulation equivalence can be decided in quadratic time.

Definition 4.1 Let N be a communication-free net and S the set of places of N. For every $s \in S$ let

$$Reach(s) := \bigcap \{M \mid M \subseteq S \wedge s \in M \wedge \forall s' \in M \forall t \in s \cdot .t \cdot \subseteq M\}$$

For a marking Σ let

$$Reach(\Sigma) := \bigcup_{\Sigma(s)>0} Reach(s)$$

This means that for every place s'

$$\Sigma \models \Diamond(s' > 0) \ \Leftrightarrow \ s' \in Reach(\Sigma)$$

It is clear that $Reach(\Sigma)$ can be computed in quadratic time.

A place s in a Petri net N with initial marking Σ is *dead* if it can never become marked, i.e. if $\Sigma \models \Box(s = 0)$. A transition t is dead if it can never fire. This means that in a communication-free net a transition is dead iff the one place in its preset is dead. For every place s in a communication-free net N with marking Σ it follows from Def. 4.1 that

$$s \text{ is dead } \Leftrightarrow \ s \notin Reach(\Sigma)$$

Definition 4.2 A place is called *useless* if it is dead or has no transitions in its postset. A transition is useless if it is dead.

Proposition 4.3 *All useless places and transitions and the corresponding arcs can be removed from a communication-free net N with initial marking Σ in quadratic time.*

Proposition 4.4 *Let N_1, N_2 be one-to-one labeled communication-free nets and Σ_1, Σ_2 their initial markings. Let N_1', N_2' be the nets that result if the useless places and transitions are removed. Then*

$$(N_1, \Sigma_1) \sim (N_2, \Sigma_2) \ \Leftrightarrow \ (N_1', \Sigma_1) \sim (N_2', \Sigma_2)$$

Definition 4.5 Let N_1, N_2 be two one-to-one labeled communication-free nets without useless places and transitions with initial markings Σ_1, Σ_2. Let S_1, S_2 be the sets of places of N_1, N_2 and T_1, T_2 the sets of transitions. Let $L_i : T_i \rightarrow Act$ be the labeling functions. The presets (postsets) of a place or transition x in the net N_i are denoted by $pre_i(x)$ ($post_i(x)$).

A *bitrap* $BT = (M_1, M_2) \subseteq S_1 \times S_2$ is defined by

- $\forall s \in M_1.\{s\}$ is a trap in N_1
- $\forall s \in M_2.\{s\}$ is a trap in N_2
- $\forall s \in M_1 \forall t \in post_1(s) \exists s' \in M_2 \exists t' \in post_2(s').\ L_1(t) = L_2(t')$
- $\forall s \in M_2 \forall t \in post_2(s) \exists s' \in M_1 \exists t' \in post_1(s').\ L_2(t) = L_1(t')$
- BT is minimal, i.e. no proper subset of BT satisfies these properties.

Proposition 4.6 *Let N_1, N_2 be two one-to-one labeled communication-free nets without useless places and transitions with initial markings Σ_1, Σ_2. The bitraps of this system can be computed in quadratic time.*

Proposition 4.7 *For every place $s \in S_i$ there is at most one bitrap (M_1, M_2) s.t. $s \in M_i$.*

Lemma 4.8 *Let N_1, N_2 be two one-to-one labeled communication-free nets without useless places and transitions with initial markings Σ_1, Σ_2. $L : T_1 \cup T_2 \rightarrow Act$ is the labeling function that assigns actions to the transitions in T_1 and T_2. $(N_1, \Sigma_1) \sim (N_2, \Sigma_2)$ iff the following conditions hold:*

A *For every place s in N_i, $i = 1, 2$ s.t. $\{s\}$ is not a trap in N_i there is a place s' in N_j s.t. $i \neq j$ and*

- $\Sigma_i(s) = \Sigma_j(s')$
- *The subnets of N_i, N_j generated by the places s, s' are equal.*

B *For every place s in N_i, $i = 1, 2$ s.t. $\{s\}$ is a trap in N_i there is a bitrap (M_1, M_2) s.t. $s \in M_i$ and*

- *Either*
  - Σ_1 *marks all places in* M_1 *and*
  - Σ_2 *marks all places in* M_2
- *Or*
  - *No place in M_1, M_2 is marked by Σ_1, Σ_2 and*
  - $\forall i \in \{1, 2\}. \forall s', s'' \in M_i . pre_i(s') = pre_i(s'')$ *and*
  - $L(pre_1(M_1)) = L(pre_2(M_2))$

Proof

$\Rightarrow$ **A** If $\{s\}$ is not a trap in N_i then there is a transition $t \in post_i(s)$ s.t. $s \notin post_i(t)$. There must be a transition t' in N_j with $L(t) = L(t')$, because there are no dead transitions. Let s' be the one place in the preset of t'. As the nets are one-to-one labeled it follows that s and s' must always contain the same number of tokens. Therefore $\Sigma_i(s) = \Sigma_j(s')$. Also the subnets generated by s and s' must be equal, because there are no useless places and the nets are one-to-one labeled.

B Now $\{s\}$ is a trap in N_i. Assume that s is not contained in any bitrap BT. Then one of the following conditions must hold:

1. $L(T_1) \neq L(T_2)$. As there are no useless transitions in N_1, N_2 it follows that $(N_1, \Sigma_1) \not\sim (N_2, \Sigma_2)$, a contradiction.
2. There are transitions $t \in T_1$ and $t' \in T_2$ s.t. $L(t) = L(t')$ and $pre_1(t)$ is a trap and $pre_2(t)$ is no trap or vice versa. As there are no useless places it follows that $(N_1, \Sigma_1) \not\sim (N_2, \Sigma_2)$, a contradiction.

So s must be contained in a bitrap (M_1, M_2). At every state in a bisimulation game either all places in $M_1 \cup M_2$ must be marked or all must be empty. If they are all marked at the beginning they will always be marked. Now consider the case where no place in M_1, M_2 is marked by Σ_1, Σ_2. As there are no useless places they can all eventually become marked. So it must be made sure that they all become marked at the same time (i.e. the same step in the bisimulation game). As there are no useless transitions the second and third condition follows.

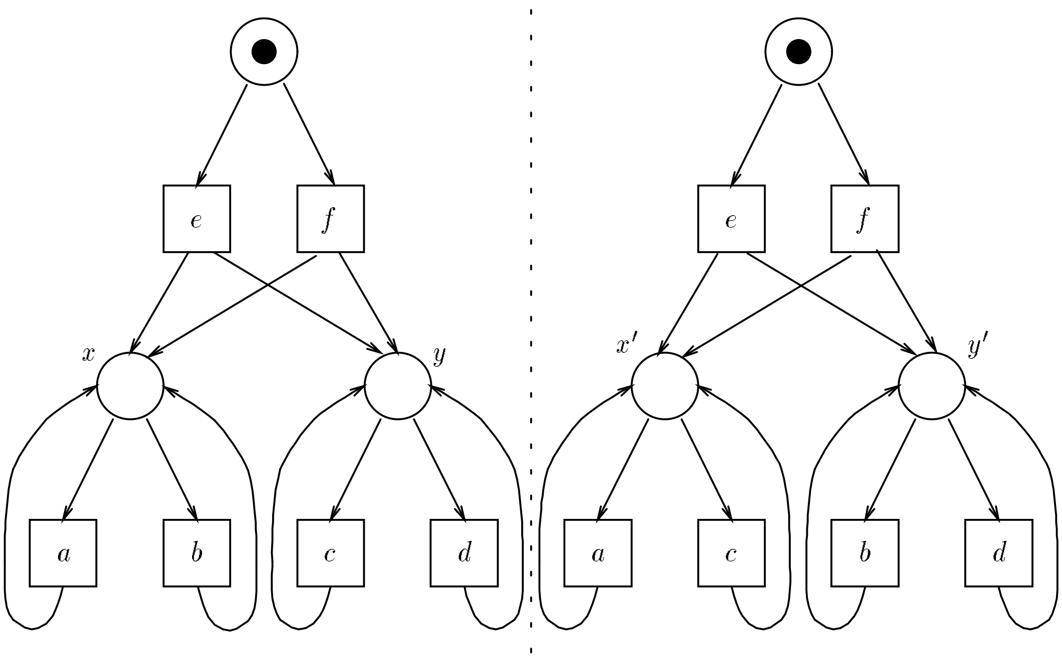
$\Leftarrow$ Assume that a transition t is enabled in (N_i, Σ_i). If $pre_i(t)$ is not a trap in N_i then a transition t' is enabled in (N_j, Σ_j) s.t. $L(t) = L(t')$, because of condition A. If $pre_i(t)$ is a trap in N_i then the same holds because of condition B. Firing both transitions yields two new systems (N_i, Σ_i'), (N_j, Σ_j') that still satisfy the conditions A and B. Therefore $(N_i, \Sigma_i) \sim (N_j, \Sigma_j')$.

□

Theorem 4.9 *Let N_1, N_2 be two one-to-one labeled communication-free nets with initial markings Σ_1, Σ_2. It can be decided in quadratic time if $(N_1, \Sigma_1) \sim (N_2, \Sigma_2)$.*

Proof First remove the useless places and transitions from both nets. By Proposition 4.3 this takes quadratic time, and by Proposition 4.4 it does not affect the result. By Proposition 4.6 computing all bitraps of the system takes quadratic time. As checking the conditions A and B can be done in quadratic time as well the result follows from Lemma 4.8. □

Figure 2: $(\{x, y\}, \{x', y'\})$ is a bitrap.



5 A model checking problem for communication-free nets

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. 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). A property holds iff there is a proof tree with a successful leaf which witnesses this truth.

The algorithm for the following problem is essentially a tableaux-based algorithm which decides the truth of a formula for an infinite state concurrent system by examining only finitely many states.

A weak branching time temporal logic is used to describe properties of a Petri-net N. The syntax of the calculus is as follows:

$$\Phi ::= s \geq k \;\mid\; s \leq k \;\mid\; \neg\Phi \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \Diamond\Phi$$

where s ranges over the places of N and $k \in \mathbb{N}$. For convenience disjunction and another quantifier $\Box$ can be added by defining $\Box := \neg\Diamond\neg$. As a convention assume that for a net of size n every constant k occurring in a subterm of the form $s \geq k/s \leq k$ of a formula satisfies $k \leq O(2^{n^2})$.

Let $\mathcal{F}$ be the set of all formulas. Let Ω be the set of all markings of N. The denotation $\|\Phi\|$ of a formula Φ is the set of markings of N inductively defined by the following rules:

$$\begin{array}{lcl}
\|s \geq k\| & = & \{\Sigma \mid \Sigma(s) \geq k\} \\
\|s \leq k\| & = & \{\Sigma \mid \Sigma(s) \leq k\} \\
\|\neg\Phi\| & = & \Omega - \|\Phi\| \\
\|\Phi_1 \wedge \Phi_2\| & = & \|\Phi_1\| \cap \|\Phi_2\| \\
\|\Diamond\Phi\| & = & \{\Sigma \mid \exists\Sigma \xrightarrow{\sigma} \Sigma'.\Sigma' \in \|\Phi\|\}
\end{array}$$

The property $\Sigma \in \|\Phi\|$ is also denoted by $\Sigma \models \Phi$. An instance of the model checking problem is a net N with a marking Σ and a formula Φ. The question is if $\Sigma \models \Phi$.

While this problem is undecidable for general Petri nets [2] it is in fact decidable for communication-free nets [2]. However, the exact complexity of the problem was unknown so far. It will be shown now that this model checking problem for communication-free nets is PSPACE-complete.

Definition 5.1 $\mathcal{F}_d \subset \mathcal{F}$ is defined as the set of all formulas with a nesting depth of quantifiers $\Diamond$ of at most d. (It follows that formulas in $\mathcal{F}_0$ contain no quantifiers.)

Lemma 5.2 *Let N be a communication-free net of size n and Σ_1 and Σ_2 two markings of N. Let $\Phi \in \mathcal{F}_d$ and $\hat{k}$ be the maximal k occurring in a subterm of the form $s \geq k/s \leq k$ of Φ. If $\Sigma_1 \leq_{(\hat{k}+1)n^d} \Sigma_2$ then*

$$\Sigma_1 \models \Phi \;\Leftrightarrow\; \Sigma_2 \models \Phi$$

Proof by induction on d.

1. If $d = 0$ then Φ doesn't contain any quantifiers and $\Sigma_1 \leq_{(\hat{k}+1)} \Sigma_2$. Let $k \leq \hat{k}$.
   - If $\Sigma_1(s) \geq k$ then $\Sigma_2(s) \geq k$ as $\Sigma_2 \geq \Sigma_1$.
   - If $\Sigma_1(s) \leq k$ then $\Sigma_2(s) = \Sigma_1(s)$, because $k \leq \hat{k}$, and therefore $\Sigma_2(s) \leq k$.
   - If $\Sigma_2(s) \geq k$ then $\Sigma_1(s) = \Sigma_2(s) \geq k$, because $k \leq \hat{k}$.

- If $\Sigma_2(s) \leq k$ then $\Sigma_1(s) \leq k$, because $\Sigma_1 \leq \Sigma_2$.

It follows that for all places s $\Sigma_1(s) \geq k \Leftrightarrow \Sigma_2(s) \geq k$ and $\Sigma_1(s) \leq k \Leftrightarrow \Sigma_2(s) \leq k$ for any $k \leq \hat{k}$. By induction on the structure of terms the result follows.

2. Now $d > 0$. For any subterm ψ of Φ s.t. $\psi \in \mathcal{F}_{d-1}$ the induction hypothesis yields $\Sigma_1 \models \psi \Leftrightarrow \Sigma_2 \models \psi$. Now it only remains to prove that $\Sigma_1 \models \psi \Leftrightarrow \Sigma_2 \models \psi$ for the minimal subterms ψ of Φ s.t. $\psi \in \mathcal{F}_d - \mathcal{F}_{d-1}$. These subterms are of the form $\psi = \Diamond\varphi$ for a $\varphi \in \mathcal{F}_{d-1}$.

   $\Rightarrow$ If $\Sigma_1 \models \Diamond\varphi$ then there is a sequence σ s.t. $\Sigma_1 \xrightarrow{\sigma} \Sigma'_1$ and $\Sigma'_1 \models \varphi$. By Lemma 3.9 there is a supersequence σ' s.t. $\Sigma_2 \xrightarrow{\sigma'} \Sigma'_2$ and $\Sigma'_1 \leq_{(\hat{k}+1)n^{(d-1)}} \Sigma'_2$. By induction hypothesis $\Sigma'_2 \models \varphi$ and therefore $\Sigma_2 \models \Diamond\varphi$.

   $\Leftarrow$ If $\Sigma_2 \models \Diamond\varphi$ then there is a sequence σ s.t. $\Sigma_2 \xrightarrow{\sigma} \Sigma'_2$ and $\Sigma'_2 \models \varphi$. By Lemma 3.8 there is a subsequence σ' s.t. $\Sigma_1 \xrightarrow{\sigma'} \Sigma'_1$ and $\Sigma'_1 \leq_{(\hat{k}+1)n^{(d-1)}} \Sigma'_2$. By induction hypothesis $\Sigma'_1 \models \varphi$ and therefore $\Sigma_1 \models \Diamond\varphi$.

□

Lemma 5.3 *Let N be a communication-free net of size n and Σ a marking of size x. Let $\Phi \in \mathcal{F}_d$ and $\hat{k}$ be the maximal k occurring in a subterm of the form $s \geq k/s \leq k$ of Φ. Then*

$$\Sigma \models \Diamond\Phi \Leftrightarrow \exists \Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}.\ \tilde{\Sigma} \models \Phi \wedge \mathit{length}(\tilde{\sigma}) \leq O((x+\hat{k})2^{n^2})$$

Proof There must be a sequence σ s.t. $\Sigma \xrightarrow{\sigma} \Sigma'$ and $\Sigma' \models \Phi$. Now let $\tilde{\sigma}$ be a subsequence of σ, possibly containing fewer cycles. How many cycles are at most needed in $\tilde{\sigma}$ in order to reach a $\tilde{\Sigma}$ s.t. $\tilde{\Sigma} \leq_{(\hat{k}+1)n^d} \Sigma'$? At most $n * (\hat{k}+1)n^d = (\hat{k}+1)n^{(d+1)}$ new tokens need to be generated. Therefore at most $(\hat{k}+1)n^{(d+1)}$ cycles are needed. So at most $(\hat{k}+1)n^{(d+1)} * n * (2^n - 1)$ new tokens are produced in the cycles. So at most $x + (\hat{k}+1)n^{(d+2)} * (2^n - 1)$ tokens are in the net for moves without cycles. By Lemma 3.6 at most

$$(x + (\hat{k}+1)n^{(d+2)} * (2^n - 1)) * \frac{2^{n^2-n} - 1}{2^n - 1}$$

non-cyclic moves can be made. This does not necessarily mean that the cycles are done first, and the non-cyclic moves afterwards. Moves belonging to cycles and non-cyclic moves can occur in any order. So altogether at most

$$(\hat{k}+1)n^{(d+2)} + (x + (\hat{k}+1)n^{(d+2)} * (2^n - 1)) * \frac{2^{n^2-n} - 1}{2^n - 1}$$

transitions need to be fired in $\tilde{\sigma}$. So $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$, $\mathit{length}(\tilde{\sigma}) \leq O((x+\hat{k})2^{n^2})$ and by Lemma 5.2 $\tilde{\Sigma} \models \Phi$. The other direction is trivial. □

The model checking problem for formulas with a nesting depth of quantifiers bounded by d is complete for the d-th order of the polynomial time hierarchy.

Lemma 5.4 *Let N be a communication-free net, Σ a marking of N s.t. $x := size(\Sigma) \leq O(2^{n^2})$ and $\Phi \in \mathcal{F}_d$. The problem $\Sigma \models \Diamond\Phi$ can be solved in Σ^p_{d+1}.*

Proof by induction on d.

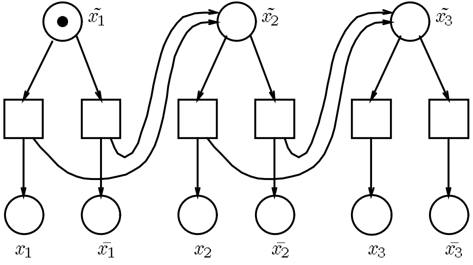
1. If $d = 0$ then Φ doesn't contain any quantifiers. By Lemma 5.3 it suffices to look for a $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$ s.t. $length(\tilde{\sigma}) \leq O((x + \hat{k})2^{n^2})$ and $\tilde{\Sigma} \models \Phi$. As $\hat{k} \leq O(2^{n^2})$ and $x \leq O(2^{n^2})$ the Parikh-vector of $\tilde{\sigma}$ can be written in polynomial space. Esparza [3] showed that for communication-free nets it is decidable in polynomial time if there is a firable sequence of transitions with a given Parikh-vector. (Let P be the Parikh-vector and M the matrix describing the net. There is a firable sequence σ with vector P iff $\Sigma + M \cdot P \geq \vec{0}$ and every nonempty siphon of the subnet generated by the transitions occurring in P is marked by Σ.) Now guess a Parikh-vector and check in polynomial time if there is a $\tilde{\sigma}$ with this Parikh-vector s.t. $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$ and $\tilde{\Sigma} \models \Phi$. It only takes polynomial time to compute the resulting marking $\tilde{\Sigma}$ and to check if $\tilde{\Sigma} \models \Phi$. So the problem can be solved in $NP = \Sigma^p_1$.

2. Now $d > 0$. Again guess a Parikh-vector of polynomial size, check in polynomial time if there is a firable sequence $\tilde{\sigma}$ with this Parikh-vector s.t. $\Sigma \xrightarrow{\tilde{\sigma}} \tilde{\Sigma}$ and compute $\tilde{\Sigma}$ in polynomial time. As $size(\Sigma) = x$ and $length(\tilde{\sigma}) \leq O((x + \hat{k})2^{n^2})$ one can assume that $size(\tilde{\Sigma}) \leq O(x + 2^n((x+\hat{k})2^{n^2})) = O((x+\hat{k})2^{n^2})$. As $size(\Sigma) \leq O(2^{n^2})$ and $\hat{k} \leq O(2^{n^2})$ it follows that $size(\tilde{\Sigma}) \leq O(2^{n^2})$. Therefore it is possible to apply the induction hypothesis and check if $\tilde{\Sigma} \models \Phi$ in polynomial time with the help of a Σ^p_d-oracle. Therefore the problem can be solved in $NP^{\Sigma^p_d} = \Sigma^p_{d+1}$.

□

Lemma 5.5 *Let N be a communication-free net, Σ a marking of N and $\Phi \in \mathcal{F}_d$. The problem $\Sigma \models \Diamond\Phi$ is Σ^p_{d+1}-hard.*

Proof The problem of the bounded quantified boolean formulae (BQBF) can be reduced to this model checking problem. This is best illustrated by an example: For the formula $\exists x_1 \forall x_2 \exists x_3.(x_1 \land \neg x_2 \land \neg x_3) \lor (x_2 \land x_3)$ the following communication-free net is constructed.



It is easy to see that

$$\exists x_1 \forall x_2 \exists x_3.(x_1 \land \neg x_2 \land \neg x_3) \lor (x_2 \land x_3) \Leftrightarrow$$
$$\Diamond(\tilde{x_2} > 0 \land \Box(\tilde{x_3} = 0 \lor \Diamond((x_1 > 0 \land \bar{x_2} > 0 \land \bar{x_3} > 0) \lor (x_2 > 0 \land x_3 > 0))))$$

□

Note that this hardness result even holds for communication-free nets with a finite state space. The lemma remains true if the logic is restricted to statements of the form $s > 0$ instead of $s \geq k / s \leq k$. Esparza [2] proved PSPACE-hardness for a simpler logic and BPPs.

Theorem 5.6 *Let N be a communication-free net, Σ a marking of N and $\Phi \in \mathcal{F}_d$. The problem $\Sigma \models \Diamond\Phi$ is Σ^p_{d+1}-complete.*

Proof directly from Lemma 5.4 and Lemma 5.5. □

Corollary 5.7 *Let N be a communication-free net, Σ a marking of N and $\Phi \in \mathcal{F}$. The problem $\Sigma \models \Phi$ is PSPACE-complete.*

This solves an open problem stated in [2] about the computational complexity of this model checking problem.

While model checking for communication free nets/BPP is decidable for this weak branching time temporal logic it is undecidable for the more expressive modal μ-calculus and several other logics [2]. On the other hand model checking context free processes with the modal μ-calculus has been proved decidable by reduction to the monadic second order theory of n successors. However there remains the problem of finding a tableau method.

6 Conclusion

Basic Parallel Processes are a weak model of computation. It can be argued that any decent model of concurrent computation should be at least as powerful as BPP. What makes them interesting is that they are a model for infinite state concurrent systems that seems to lie just on the "border of decidability". This means that some properties that are undecidable for more powerful notions of concurrency are still decidable for BPP. These include strong bisimulation equivalence [10], weak bisimulation equivalence to a finite state LTS (see section 3) and model checking with the weak branching time temporal logic of section 5. On the other hand BPP are powerful enough to make some properties undecidable. Those properties include model checking with the modal μ-calculus [2] and language equivalence [12].

Some problems for BPP are still open. We conjecture that not only weak bisimulation equivalence between a BPP and a finite state LTS is decidable (as proved in section 3) but that weak bisimulation equivalence between two BPPs is decidable as well. Another problem is that although some problems about bisimulation equivalence for BPP have been proved decidable the algorithms are not primitive recursive. This does not necessarily mean that these problems are unsolvable in practice, as there are no hardness results yet.

References

[1] S. Christensen. *Decidability and Decomposition in Process Algebras.* PhD thesis, Edinburgh University, 1993.

[2] Javier Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 1995.

[3] Javier Esparza. Petri nets, commutative context-free grammars and basic parallel processes. In Horst Reichel, editor, *Fundamentals of Computation Theory*, number 965 in LNCS. Springer Verlag, 1995.

[4] J. F. Groote and H. Hüttel. Undecidable equivelences for basic process algebra. *Information and Computation*, 1994.

[5] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, 1979.

[6] P. Jančar. Decidability questions for bisimilarity of petri nets and some related problems. Technical Report ECS-LFCS-93-261, Edinburgh University, April 1993.

[7] P. Jančar. Undecidability of bisimilarity for petri nets and related problems. *Theoretical Computer Science*, 1995.

[8] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[9] H. Hüttel S. Christensen and C. Stirling. Bisimulation equivelence is decidable for all context-free processes. In W.R. Cleaveland, editor, *Proceedings of CONCUR 92*, number 630 in LNCS. Springer Verlag, 1992.

[10] Y. Hirshfeld S. Christensen and F. Moller. Bisimulation equivalence is decidable for basic parallel processes. In E. Best, editor, *Proceedings of CONCUR 93*, number 715 in LNCS. Springer Verlag, 1993.

[11] Y. Hirshfeld S. Christensen and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In *Proceedings of LICS93.* IEEE Computer Society Press, 1993.

[12] M. Jerrum Y. Hirshfeld and F. Moller. A polynomial algorithm for deciding bisimulation of normed context free processes. Technical report, LFCS report series 94-286, Edinburgh University, 1994.

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

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

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

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

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

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

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

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