TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Checking System Properties via Integer Programming

Stephan Melzer, Javier Esparza

TUM-I9526
SFB-Bericht Nr.342/13/95 A
September 1995

TUM–INFO–09-95-I26-0/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
Arcisstr. 21 / Postfach 20 24 20
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Checking System Properties via Integer Programming*

Stephan Melzer and Javier Esparza
Institut für Informatik
Technische Universität München
Arcisstr. 21, D-80290 München
e-mail:{melzers,esparza}@informatik.tu-muenchen.de

Abstract. The marking equation is a well known verification method in the Petri net community. It has also be applied by Avrunin, Corbett *et al.* to automata models. It is a semidecision method, and it may fail to give an answer for some systems, in particular for those communicating by means of shared variables. In this paper, we complement the marking equation by a so called trap equation. We show that both together significantly extend the range of verifiable systems by conducting several case studies.

1 Introduction

The use of linear algebra and integer programming for verification purposes has a long tradition in Petri net theory [6, 16, 15, 17]. One of the best known techniques is the *state* or *marking equation* [6, 17]. This is a linear equation which can be easily derived from the description of the net and its initial marking (in linear time). It can be seen as a set of linear constraints $\mathcal{L}$ that every reachable marking must satisfy. In other words, the solutions of $\mathcal{L}$ are a superset of the reachable markings. In order to use the marking equation, we add to it new linear constraints $\mathcal{L}_P$, which specify the markings which do *not* satisfy a desirable property P.[2] Then, we use integer programming to solve the system $\mathcal{L} \cup \mathcal{L}_P$: if the system has no solution, every reachable marking satisfies P.

The disadvantage of this technique is the fact that the markings satisfying $\mathcal{L}$ are only a superset of the reachable markings: the solutions of $\mathcal{L} \cup \mathcal{L}_P$ may or may not correspond to a reachable marking. Therefore, the marking equation is only a *semidecision method*. Its main advantage is that it does not explore the state space, and therefore it avoids the state explosion problem. It can also be used to verify systems having infinite state spaces.

The marking equation can be applied to many different models of concurrency, not only to Petri nets. Actually, the most comprehensive study of its applications for verification has been carried out by Avrunin, Corbett *et al.* using coupled automata as a model [2, 3, 8]. They have developed the Constrained

* This work was partially supported by the Sonderforschungsbereich SFB-342 A3.
[2] It is also possible to impose linear constraints on the occurrence sequence leading to those markings. This is a very useful feature, but we omit it here for the sake of simplicity.

Expression Toolset, later updated to the Inequality Necessary Condition Analyzer (INCA), a tool for the verification of a large class of safety and liveness properties. It is easy to see that the basis of the technique implemented in INCA is equivalent to the marking equation. In [7], Corbett shows that INCA is able to prove deadlock freedom for 19 different examples taken from different sources, and can compete with symbolic and partial order theorem provers.

One of the main limitations of the marking equation is that it tends to fail for systems which communicate via shared variables. For instance, it cannot prove mutual exclusion of any of the most popular mutual exclusion algorithms (Dekker's, Dijkstra's, Knuth's, Peterson's etc.) without user's help. The reason is that the method is not sensitive to the guards which allow to perform an action only if a variable has a certain value, in the sense that the systems with or without the guards are assigned the same set of constraints. Since the correctness of these algorithms crucially depends on these guards, the method fails.

In this paper, we show how to obtain a set of constraints which better approximate the set of reachable markings, and are sensitive to these guards. We then test the improved algorithm on a number of examples. In particular, we automatically prove mutual exclusion of five mutual exclusion algorithms.

This refined set of constraints is derived from some results of Petri net theory concerning so called *traps*. Therefore, it is convenient to present our results in Petri net terms. However, there would be no problem in recasting them for, say, the communicating automata of Corbett [7], the synchronized products of transition systems of Arnold and Nivat (see, for instance, [1]), or for CCS processes of the form $(P_1 \mid \ldots \mid P_n)\backslash L$, where the P_i are regular. All of them can be easily translated into (1-safe) Petri nets. The common idea of the translations is simple: each sequential component is modelled by means of a Petri net, just mapping states to places and transitions of the transition system into transitions of the Petri net. Communication is then modelled by merging transitions.

The paper is organised as follows. In Section 2 we introduce some basic definitions. Section 3 describes the marking equation. In Section 4 we introduce traps, and present our improved method. In Section 5 we apply the results to examples. In Section 6 we present a result on checking deadlock freedom. Finally, we present our conclusions in Section 7.

2 Basic notations

A *net* is a triple $N = (P, T, W)$ where $P \cap T = \emptyset$ and $W\colon (P \times T) \cup (T \times P) \to \mathbb{N}$. P is the set of *places* (symbolized by circles), T the set of *transitions* (symbolized by rectangles) and W is the *weight function*. The *pre-set* of $x \in P \cup T$ is ${}^\bullet x = \{y \in P \cup T \mid W(y, x) > 0\}$. The *post-set* of $x \in P \cup T$ is $x^\bullet = \{y \in P \cup T \mid W(x, y) > 0\}$. The pre- and post-set of a subset of $P \cup T$ are the union of the pre- and post-sets of its elements.

All the examples of Section 5 (and all the examples of [7]) can be modelled by *ordinary* nets, in which the weight function has codomain $\{0, 1\}$. However, more general weight functions play an important role in the development of the results of Section 4, and that is why we define nets in this generality.

A function $M : P \to \mathbb{N}$ is called a *marking*. A *Petri net* is a pair (N, M_0) where N is a net and M_0 a marking of N called *initial marking*. A transition $t \in T$ is *enabled at* M iff $\forall p \in {}^{\bullet}t : M(p) \geq W(p,t)$. If t is enabled at M, then t may *fire* or *occur*, yielding a new marking M' (denoted $M \xrightarrow{t} M'$), where $M'(p) = M(p) + W(t,p) - W(p,t)$.

A sequence of transitions, $\sigma = t_1 t_2 \ldots t_r$ is an *occurrence sequence* of (N, M_0) iff there exist markings $M_1, \ldots M_r$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots \xrightarrow{t_r} M_r$. The marking M_r is said to be *reachable* from M_0 by the occurrence of σ (denoted $M \xrightarrow{\sigma} M_r$).

A Petri net (N, M_0) is *safe* iff $M(p) \leq 1$ for every reachable marking M and every place p.

A *linear programming problem* or linear problem is a system $A \cdot X \leq B$ of linear (in)-equalities called the *constraints*, plus maybe a linear function $C^T \cdot X$ called the *objective function*. A solution of the problem is a vector of rational numbers that satisfy the constraints. A solution is *optimal* if it maximises the value of the objective function (over the set of all solutions).

An *integer programming problem* consists of the same elements as a linear programming problem, but only integer solutions are allowed. In a *mixed programming problem*, some variables may take rational values, and some only integer ones.

A linear, integer or mixed programming problem is *feasible* if it has a solution. Otherwise it is infeasible.

3 The marking equation

Each place p of a net has associated a *token conservation* equation. Given an occurrence sequence $M_0 \xrightarrow{\sigma} M$, the number of tokens that p contains at the marking M is equal to the number of tokens it contains at M_0, plus the tokens added by (the firings of) the input transitions of p, minus the tokens removed by the output transitions. If we denote by $\#(\sigma, t)$ the number of times that a transition t occurs in σ, we can write the *token conservation* equation for p as:

$$M(p) = M_0(p) + \sum_{t \in {}^{\bullet}p} \#(\sigma,t) W(t,p) - \sum_{t \in p^{\bullet}} \#(\sigma,t) W(p,t)$$

The token conservation equations for every place are usually written in the following matrix form:

$$M = M_0 + \mathbf{N} \cdot \overrightarrow{\sigma}$$

where $\overrightarrow{\sigma} = (\#(\sigma, t_1), \ldots, \#(\sigma, t_m))$ is called the *Parikh vector* of σ, and $\mathbf{N}$ denotes the *incidence matrix* of N, a $P \times T$ integer matrix given by

$$\mathbf{N}(p,t) = W(p,t) - W(t,p)$$

If a given marking M is reachable from M_0, then there exists a sequence σ satisfying $M_0 \xrightarrow{\sigma} M$. So the following problem has at least one solution, namely $X := \overrightarrow{\sigma}$.

$$\begin{array}{l} \text{Variables: } X, \text{ integer.} \\ M = M_0 + \mathbf{N} \cdot X \\ X \geq 0 \end{array}$$

The equation $M = M_0 + \mathbf{N} \cdot X$ (and, by extension, the whole problem) is called the *marking equation*. If the marking equation has no solution, then M is not reachable from M_0.

We wish to verify that every reachable marking satisfies a desirable property, or, equivalently, that no marking satisfying the negation of this desirable property is reachable. The negation of the property can often be expressed by means of *linear constraints* on the markings of the net. Here are two examples:

- Mutual exclusion.
  In Petri net models of mutual exclusion algorithms the possible states of a process (idle, requesting, critical, ...) are modelled by places which can hold at most one token. The process is in the critical section if the corresponding place is marked. If $s_1, \ldots, s_n$ are the places corresponding to the critical sections, then the reachable markings that violate the mutual exclusion property are those satisfying
  $$M(s_1) + \ldots + M(s_n) \geq 2$$
- Deadlock freedom in safe Petri nets.
  A marking is a deadlock if it does not enable any transition. In safe Petri nets a place can hold at most one token, and therefore a transition is enabled if and only if the total number of tokens in its input places is at least equal to the number of input places. In other words, the reachable deadlocked markings satisfy
  $$\sum_{s \in {}^\bullet t} M(s) \leq |{}^\bullet t|$$
  for every transition t.

A *linear property* of N is a predicate $\mathcal{P}$ on the markings of N (or, equivalently, a subset of the markings of N) such that

$$\mathcal{P}(M) \Leftrightarrow A \cdot M \leq b$$

for some matrix A and vector b. We can use the marking equation to verify properties whose negation is linear. If some marking satisfying $\mathcal{P}$ is reachable from M_0, then the *generalised marking equation*

$$\begin{array}{l} \text{Variables: } M, X\text{: integer} \\ \quad M = M_0 + \mathbf{N} \cdot X \\ A \cdot M \leq b \\ M, X \geq 0 \end{array}$$

has a solution.[3] Therefore, if the generalised marking equation is infeasible, every reachable marking satisfies the negation of P. We can use integer programming to check infeasibility.

The implication "infeasibility $\Rightarrow$ P holds for every reachable marking" still holds if M and X are allowed to take rational values. So, in principle, one may try to use ordinary linear programming to check infeasibility. Unfortunately, the experiments show that in most cases even though the desired property holds, the marking equation has non-integer solutions, and therefore linear programming is of little use. Using integer programming leads to much better results [7, 8].

Unfortunately, the marking equation still fails very often when the Petri net models a distributed system with shared variables. The components of this kind of systems test the value of a variable to determine the flow of control. Now, consider the two Petri nets of Figure 1. The Petri net of the left models a component which may change of state, from s_0 to s_1, only if the variable x has value 0, which happens not to be the case. In the Petri net on the right, the component can change its state independently of the value of x. Obviously, the marking $\{s_1\}$ is not reachable on the left, and reachable on the right. However, the marking equations of these two nets *coincide*. Therefore, the generalised marking equation cannot be used to prove that $\{s_1\}$ is not reachable on the left.

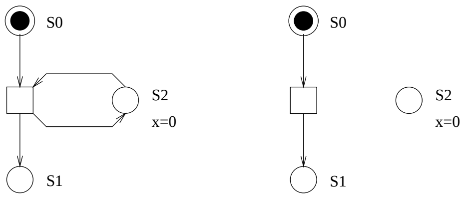

Fig. 1. A limitation of the marking equation

We could of course prove this by constructing the reachability graph, which is very small in this example, but may grow exponentially in the size of the net (or be infinite). An alternative is the use of *traps* [21, 9].

Definition 1. *Traps*

A set R of places of a net is a trap if $R^\bullet \subseteq {}^\bullet R$. ■ 1

In the sequel, we shall use the letter Θ to denote traps. Traps have the following fundamental property:

[3] Since M is in fact a linear function of X, it would still be more general to add a constraint of the form $C \cdot X \leq d$, and this is in fact the approach of [8]. Since the examples of this paper only consider constraints on markings, we will use the constraint shown above for clarity.

Proposition 2. *Marked traps remain marked*

Let (N, M_0) be a Petri net, and let Θ be a trap of N. If Θ is marked at M_0 (i.e., if $\sum_{p \in \Theta} M_0(p) > 0$), then Θ remains marked at every reachable marking. ■ 2

The set $\{s_0, s_2\}$ is a trap of the net on the left, and this trap is marked at the initial marking $\{s_0\}$. However, the trap is not marked at $\{s_1\}$. Therefore, the marking $\{s_1\}$ is not reachable.

If a marking marks every trap that is marked at M_0 we say that it satisfies the *trap property*. Proposition 2 states that, on top of the marking equation, a reachable marking must satisfy the trap property as well. We have thus a refined test of non-reachability.

In order to check that every marking satisfying a linear property $\mathcal{P}$ violates the trap property we may compute all the traps marked at M_0, say $\Theta_1, \ldots, \Theta_n$, and then compute iteratively the subsetes $\mathcal{P}_i$ of $\mathcal{P}$ that mark the traps $\Theta_1, \ldots, \Theta_i$ for $1 \leq i \leq n$. However, this method is very inefficient, because the number of traps may be exponential in the size of the net[4]. In order to make traps useful for automatic verification, we have to find an alternative, which we present in the next section.

4 The trap equation

In this section we obtain the *generalised trap equation* for a linear property $\mathcal{P}$. This is a linear equation which has a solution if and only if no marking satisfies simultaneously $\mathcal{P}$ and the trap property.

The first step towards our goal is to find a link between traps and linear algebra. Fortunately, we can profit from several existing results. In [14], Lautenbach showed that there exists a tight relation between the traps of a net N and the solutions of the equation $Y^T \cdot \mathbf{N}_\Theta = 0$, where N_Θ is obtained from N by means of a relatively complicated transformation. Later, Lautenbach's results were used and slightly improved by Esparza and Silva in [10]. Finally, Ezpeleta, Couvreur and Silva found another improvement [11]. They showed that Lautenbach's net N_Θ can be replaced by a simpler one. N and the new N_Θ have the same places, transitions and arcs: they only differ in the *weights* of some arcs leading from transitions to places.

Theorem 3. *Algebraic characterization of traps [11]*

Let $N = (P, T, W)$ be a net. Let $N_\Theta = (P, T, W_\Theta)$, where

$$W_\Theta(p, t) = W(p, t)$$
$$W_\Theta(t, p) = \begin{cases} \sum_{p' \in {}^\bullet t} W(p', t) & \textit{if } p \in t^\bullet \\ 0 & \textit{otherwise} \end{cases}$$

[4] In fact, it suffices to compute all minimal traps, which are the nonempty traps not included in any other trap. However, there may also be exponentially many minimal traps.

A set $\Theta \subseteq P$ is a trap of the net N if and only if the equation $Y^T \cdot \mathbf{N}_\Theta \geq 0$ has a nonnegative solution Y such that $||Y|| = \Theta$. ■ 3

We illustrate this result on the Petri net of Figure 2. The vectors Y_1^T and Y_2^T satisfy the equation of Theorem 3, and therefore $\{s_3, s_5\}$ and $\{s_3, s_4, s_6\}$ are traps of the net. The vector Y_3^T does not satisfy it, and in fact $\{s_1, s_2\}$ is not a trap.

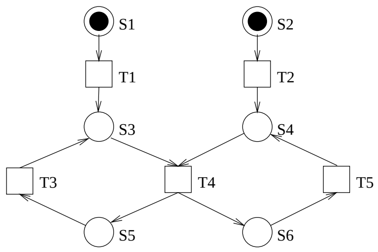

Fig. 2. An example.

$$\mathbf{N}_\Theta = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 2 & -1 \end{pmatrix} \qquad \begin{aligned} Y_1^T &= (1,0,1,0,1,0) \\ Y_2^T &= (0,0,1,1,0,1) \\ Y_3^T &= (1,1,0,0,0,0) \end{aligned}$$

We can use Proposition 3 to test if a marking M violates the trap property.

Proposition 4.

Let (N, M_0) be a Petri net, and let M be a marking of N. M satisfies the trap property if and only if the problem below is infeasible.

Variables: Y: rational.

$$\begin{aligned} Y^T \cdot \mathbf{N}_\Theta &\geq 0 \\ Y &\geq 0 \ \langle \Theta = ||Y|| \textit{ is a trap} \rangle \\ Y^T \cdot M_0 &> 0 \ \langle \Theta \textit{ is initially marked} \rangle \\ Y^T \cdot M &= 0 \ \langle \Theta \textit{ is not marked at } M \rangle \end{aligned}$$

Proof: By Proposition 3, a solution of the problem corresponds to a trap initially marked, but unmarked at M, and vice versa. ■ 4

Now, in order to test if M violates the trap property we solve a linear programming problem instead, which intensionally checks if *every* initially marked trap remains marked at M.

However, Proposition 4 is not directly useful when we consider linear properties. If M becomes a variable subject to the linear condition $A \cdot M \leq b$, then the equation $Y^T \cdot M = 0$ becomes non-linear, which very much complicates the verification. To remove this difficulty we shall use one of the many versions of the Minkowski-Farkas Lemma (see, for instance, [22]).

Theorem 5. *Minkowski-Farkas Lemma*

One and only one of the following two problems is feasible:

Variables: X*: rational.*	*Variables:* Y*: rational.*
$A \cdot X \leq b$ $X \geq 0$	$Y^T \cdot A \geq 0$ $Y^T \cdot b < 0$ $Y \geq 0$

In order to apply this theorem, we first have to modify the problem of Proposition 4. We observe that, since M is a nonnegative vector and any solution Y must also be nonnegative, the constraint $Y^T \cdot M = 0$ can be safely replaced by $Y^T \cdot M \leq 0$. So the problem is equivalent to (i.e., has the same solutions as):

$$\begin{aligned} &\text{Variables: } Y\text{: rational.} \\ &Y^T \cdot (\mathbf{N}_\Theta | -M) \geq 0 \\ &Y^T \cdot (-M_0) < 0 \\ &Y \geq 0 \end{aligned}$$

where $(\mathbf{N}_\Theta | -M)$ denotes the matrix obtained by adding $-M$ to $\mathbf{N}_\Theta$ as rightmost column.

Now, by Proposition 4 and the Minkowski-Farkas Theorem, M satisfies the trap property if and only if the following problem is feasible:

$$\begin{aligned} &\text{Variables: } X\text{: rational.} \\ &(\mathbf{N}_\Theta | -M) \cdot X \leq -M_0 \\ &X \geq 0 \end{aligned}$$

Notice that the dimension of X is equal to the number of transitions of N plus 1, because of the addition of the column M. Define $X = (X' \mid x)$, i.e., X' is the vector containing all the components of X but the last, and x is the last component of X. With these notations, we can rewrite the problem as:

$$\begin{aligned} &\text{Variables: } X', x\text{: rational.} \\ &xM \geq M_0 + \mathbf{N}_\Theta \cdot X' \\ &X', x \geq 0 \end{aligned}$$

Assume that this problem has a solution for $x = 0$. Then, since M is nonnegative, it also has a solution for every $x > 0$. So we can replace $x \geq 0$ by $x > 0$, and the resulting problem is still feasible if and only if M satisfies the trap property. Now, since $x > 0$, we can divide the first inequality by it. Redefining $X := \frac{1}{x}X'$ and then $x := \frac{1}{x}$, we finally get the *trap equation*:

$$\begin{aligned}
&\text{Variables: } M \text{ :integer; } X,\ x\text{: rational.}\\
&M \geq xM_0 + \mathbf{N}_{\Theta} \cdot X\\
&X \geq 0\\
&x > 0
\end{aligned}$$

We have reached our goal: the trap equation is linear, and M appears isolated on the left side, as in the marking equation. We can thus generalise it to linear properties by adding the constraint $A \cdot M \leq b$.

Theorem 6. *Generalised trap equation*

Let (N, M_0) be a Petri net, and let P be a linear property of the markings of N, characterised by the equation $A \cdot M \leq b$. If the problem below is infeasible, then no marking satisfies both P and the trap property.

$$\begin{aligned}
&\textit{Variables: } M\textit{: integer; } X,\ x\textit{: rational}\\
M &\geq xM_0 + \mathbf{N}_{\Theta} \cdot X\\
A \cdot M &\leq b\\
M, X &\geq 0\\
x &> 0
\end{aligned}$$

■ 6

Finally, putting together the marking and trap equations we obtain a negative test for linear properties:

Corollary 7.

Let (N, M_0) be a Petri net, and let P be a linear property of the markings of N, characterised by the equation $A \cdot M \leq b$. If the problem below is infeasible, then every reachable marking satisfies the negation of P.

$$\begin{aligned}
&\textit{Variables: } M,\ X_1\textit{: integer; } X_2,\ x\textit{: rational}\\
M &= M_0 + \mathbf{N} \cdot X_1\\
M &\geq xM_0 + \mathbf{N}_{\Theta} \cdot X_2\\
A \cdot M &\leq b\\
M, X_1, X_2 &\geq 0\\
x &> 0
\end{aligned}$$

■ 7

This problem can be solved using *mixed programming*, a combination of linear and integer programming. Mixed programming solves systems of the form $A \cdot X \leq b$, where part of the variables are required to take integer values, while others may be rational. The constraint $x > 0$ does not fit in this format, but this problem can be easily solved making use of the optimization facilities of mixed programming solvers: we solve the system with $x \geq 0$ as constraint, but search for the solution with maximal value of x. If this value is 0, then the original problem is infeasible.

5 Examples

In this section we show that a number of properties of several systems that could not be verified by the marking equation alone can be verified by the combination of the marking equation and the trap equation.

As a first case study, we consider five popular mutual exclusion algorithms taken from [20], namely those by De Bruijn, Dekker, Dijkstra, Knuth and Peterson. For each of them we verify deadlock freeness and mutual exclusion.

The algorithms are easily encoded in $\mathrm{B(PN)}^2$ (Basic Petri net Programming Notation), an imperative language designed to have a simple Petri net semantics [5]. 1-safe Petri nets are then automatically generated by the PEP-tool [4]. We then generate the corresponding mixed problems, which are solved using CPLEX^{TM} (version 3.0) on a SUN SPARC 20/712.

None of the properties can be proved using linear programming. However, we do not have to require both M and X_1 to be integer in Corollary 7: it suffices to require it for M. The results of the two tables below correspond to this case.

In the table on the left we have considered algorithms for two processes. On the right we have considered Dijkstra's algorithm for n processes.

Both tables have the same structure. The first column shows the name of example, e.g. *Dijkstra 5* means Dijkstra's mutex algorithm for 5 processes. The next two numbers indicate the numer of places and transitions of the Petri net. PEP generates a number of redundant places and transitions, which have not been removed for the case study. The fourth column describes the verified property: Deadlock (actually deadlock-freedom) or Mutex (mutual exclusion). The next column shows which constraints were needed to verify the property: ME (marking equation) or ME + TE (marking equation plus trap equation). The last column gives the CPU time in seconds.

Example	$\lvert P\rvert$	$\lvert T\rvert$	Property	Program	Time
Dekker	50	75	Mutex	TE + ME	0.27
			Deadlock	TE + ME	0.61
Peterson	40	69	Mutex	TE + ME	0.31
			Deadlock	ME	0.44
Dijkstra 2	64	89	Mutex	TE + ME	0.22
			Deadlock	ME	0.25
Knuth 2	74	140	Mutex	TE + ME	0.67
			Deadlock	ME	0.67
De Bruijn 2	80	166	Mutex	TE + ME	0.91
			Deadlock	ME	1.09

Example	$\lvert P\rvert$	$\lvert T\rvert$	Property	Program	Time
Dijkstra 2	64	89	Mutex	TE + ME	0.22
			Deadlock	ME	0.25
Dijkstra 3	98	160	Mutex	TE + ME	5.02
			Deadlock	ME	0.88
Dijkstra 4	134	257	Mutex	TE + ME	28.50
			Deadlock	ME	1.55
Dijkstra 5	172	386	Mutex	TE + ME	120.12
			Deadlock	ME	10.45
Dijkstra 6	212	553	Mutex	TE + ME	144.37
			Deadlock	ME	53.30

The next table shows results for a a slotted ring protocol described in [18], in which n processes are placed in a ring. In [18] the state space of the example was encoded into BDDs and then used to check different properties, one of which was deadlock freedom. The construction of the BDD for a ring of 9 processes (the largest ring considered in [18]) took 4080 seconds. Using our method we can prove deadlock-freedom in 0.68 seconds. The trap equation is not needed in this case. The example shows that linear constraint methods can compete with symbolic model checkers (there exist other examples (see [7]) in which BDD methods are more efficient).

Example	$\|P\|$	$\|T\|$	Property	Program	Time
Slotted Ring 2	20	20	Deadlock	ME	0.02
Slotted Ring 3	30	30	Deadlock	ME	0.03
Slotted Ring 4	40	40	Deadlock	ME	0.03
Slotted Ring 5	50	50	Deadlock	ME	0.07
Slotted Ring 6	60	60	Deadlock	ME	0.20
Slotted Ring 7	70	70	Deadlock	ME	0.32
Slotted Ring 8	80	80	Deadlock	ME	0.63
Slotted Ring 9	90	90	Deadlock	ME	0.68
Slotted Ring 10	100	100	Deadlock	ME	2.72

Finally, we consider a less academic example. We prove deadlock freedom of two versions of a call handling for intelligent telephone networks which is closely related to a *Basic Call State Model* [19] of the ITU–T (former CCITT) standardization committee. The systems are described in [13]. We have used the $B(PN)^2$ translations of [12]. The first version (Telephone) is the original protocol, while the second version (Telephone (par)) is a refinement which allows parallel communications.

Example	$\|P\|$	$\|T\|$	Property	Program	Time
Telephone	87	188	Deadlock	ME + TE	10.82
Telephone(par)	232	672	Deadlock	ME + TE	705.68

6 Siphons

In Petri net theory, traps are usually studied together with siphons [21, 9]. The results of Section 4 lead to 'dual' results about siphons. We study their possible applications in this section.

Definition 8. *Siphons, proper siphons*

A set R of places of a net is a siphon if ${}^\bullet R \subseteq R^\bullet$. A siphon is called proper if it is not the empty set. ■ 8

In the sequel, we shall use the letter Σ to denote siphons. Since a transition which puts tokens in the places of a siphon also removes tokens from them, we have the following fundamental property:

Proposition 9. *Unmarked siphons remain unmarked*

Let (N, M_0) be a Petri net, and let Σ be a siphon of N. If Σ is unmarked at M_0, then Σ remains unmarked at every reachable marking. ■ 9

Proposition 9 provides a further negative test for reachability: if M marks some siphon unmarked at M_0, then M is not reachable. Using another version of the Alternatives Theorem we can obtain a siphon equation, which may be added to the marking and trap equations. However, the siphon equation has little interest. The reason is the following: since a siphon Σ unmarked at M_0 remains unmarked, no transition of $\Sigma^\bullet$ can *ever* occur. This is usually undesirerable and a very serious design error. In all the Petri net models we have considered so far (correct or incorrect), the initial marking marks every siphon, and so the siphon equation does not add discriminating power.

Siphons do help in a different way. In Section 3 we showed that the set of deadlocked markings of a Petri net that put at most one token on a place is linear. It is easy to see that this property ceases to hold if the deadlocks may put more than one token. In general, all we can say is that the set of deadlocks is the union of a finite number of linear sets, namely those characterised by equations of the form

$$M(s_1) + \ldots + M(s_n) = 0$$

where the set $\{s_1, \ldots, s_n\}$ contains exactly one input place of each transition. So in principle we could verify deadlock freedom by solving as many integer problems as linear sets. However, this is very inefficient, because the number of linear sets may be exponential in the size of the net.

The following observation is the key to a better method:

Proposition 10.

Let $N = (P, T, W)$ be a net, and let M be a deadlocked marking of N. The set $\Sigma = \{p \in P \mid M(p) = 0\}$ is a proper siphon of N. ■ 10

By this proposition, in order to check deadlock freedom it suffices to verify that every proper siphon remains marked at every reachable marking. Moreover, this new property is not too strong: most correct systems satisfy it, because the input transitions of an unmarked siphon cannot occur anymore, and, once again, this is undesirable in all the examples we have examined.

We borrow again a result from [11] :

Theorem 11. *Algebraic characterization of siphons [11]*

Let $N = (P, T, W)$ be a net. Let $N_\Theta = (P, T, W_\Sigma)$, where

$$W_\Sigma(p,t) = \begin{cases} \sum_{p' \in t^\bullet} W(t,p') & \text{if } p \in {}^\bullet t \\ 0 & \text{otherwise} \end{cases}$$
$$W_\Sigma(t,p) = W(t,p)$$

A set $\Sigma \subseteq P$ is a siphon of the net N if and only if the equation $Y^T \cdot \mathbf{N}_\Sigma \leq 0$ has a nonnegative solution Y such that $||Y|| = \Sigma$. ■ 11

So a marking M of N satisfies the siphon property iff the problem

Variables: Y:rational.
$Y^T \cdot \mathbf{N}_\Sigma \leq 0$

$Y \geq 0$ $\langle \Sigma = ||Y||$ is a siphon.$\rangle$

$Y^T \cdot M = 0$ $\langle \Sigma$ is not marked at M.$\rangle$

is feasible. Using another version of the Alternatives Theorem and following a procedure similar to the one we used for the trap equation, we obtain that the markings satisfying the siphon property are the solutions of the equation $M > N_\Sigma \cdot X$, where $X \leq 0$. Then, the markings which violate the property are those satisfying $M_i \leq (N_\Sigma)_i \cdot X$, where M_i is the i-th component of M, and $(N_\Sigma)_i$ the i-th row of N_Σ. So we have:

Theorem 12.

Let (N, M_0) be a Petri net. If none of the problems below is feasible, then every reachable marking marks all siphons, and (N, M_0) is deadlock free.

$$\begin{aligned} &\textit{Variables: } M,\ X_1\textit{: integer; } X_2\textit{: rational} \\ &\quad M = M_0 + \mathbf{N} \cdot X_1 \\ &\quad M_i \leq (\mathbf{N}_\Sigma)_i \cdot X_2 \\ &M, X_1 \geq 0 \\ &\quad X_2 \leq 0 \end{aligned}$$

where M_i is the i-th component of M, and $(N_\Sigma)_i$ the i-th row of N_Σ.

■ 12

The number of inequation systems to solve is equal to the number of places of the net. So we have reduce the possibly exponential number of systems to linearly many.

7 Conclusion

We have extended the range of systems that can be verified using linear constraints by adding to the marking equation a new trap equation. The new equation proves to be very useful for the analysis of systems communicating by means of shared variables. We have proved properties of five mutual exclusion algorithms and a telephone communication protocol, none of which could be automatically proved before by linear methods.

We have also given a natural solution to a limitation of the method, namely the fact that deadlock-freedom is not a linear property for arbitrary Petri nets. We have introduce a slightly stronger property, in practice as desirable as deadlock freedom, which can be computed more easily.

References

1. André Arnold. Verification and comparison of transition systems. In M.C. Gaudel and J.P. Jouannaud, editors, *TAPSOFT '93: Theory and Practice of Software Development*, volume 668 of *Lecture Notes in Computer Science*, pages 121–135. Springer-Verlag, 1993.
2. G. S. Avrunin, J. C. Corbett, and U. A. Buy. Integer Programming in the Analysis of Concurrent Systems. In K.G. Larsen and A. Skou, editors, *Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 92–102, 1991.

3. G.S. Avrunin, U.A. Buy, J.C. Corbett, L.K. Dillon, and J.C. Wileden. Automated Analysis of Concurrent Systems with the Constrained Expression Toolset. *IEEE Transactions in Software Engineering*, 17(11):1204–1222, 1991.
4. E. Best and H. Fleischhack (eds.). Pep: Programming environment based on nets. Technical report, University of Hildesheim, Germany, 1994.
5. E. Best and R. P. Hopkins. $B(PN)^2$ – A Basic Petri Net Programming Notation. In *Proc. of PARLE-93*, volume 694 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 1993
Also: Hildesheimer Informatik Fachbericht 27/92 (1992).
6. G.V. Brams. *Réseaux de Petri: Theorie et Practique, Vols. I and II.* Masson, 1982.
7. J.C. Corbett. Evaluating Deadlock Detection Methods for Concurrent Software. In T. Ostrand, editor, *Proceedings of the 1994 International Symposium on Software Testing and Analysis*, pages 204–215, New York, 1994.
8. J.C. Corbett and G.S. Avrunin. Using Integer Programming to Verify general Safety and Liveness properties. *Formal Methods in System Design*, 6(1):97–123, 1995.
9. J. Desel and J. Esparza. *Free-choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.
10. J. Esparza and M. Silva. A Polynomial-Time Algorithm to Prove Liveness of Bounded Free Choice Nets. *Theoretical Computer Science*, 102:185–205, 1992.
11. J. Ezpeleta, J. M. Couvreur, and M. Silva. A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Colored Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 674 of *Lecture Notes in Computer Science*, pages 126–147. Springer Verlag, 1993.
12. B. Grahlmann. Verifying telecommunication protocols with pep (draft). Technical report, University of Hildesheim, Germany, 1995.
13. Stephan Kleuker. A gentle introduction to specification engineering using a case study in telecommunications. In P.D. Mosses, M. Nielsen, and M.I. Schwartzbach, editors, *TAPSOFT '95*, volume 915 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
14. K. Lautenbach. Linear algebraic calculation of deadlocks and traps. In H.J. Genrich K. Voss and G. Rozenberg, editors, *Concurrency and Nets*, pages 315–336. Springer-Verlag, 1987.
15. K. Lautenbach. Linear Algebraic Techniques for Place/Transition Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advance in Petri Nets 1986*, volume 254 of *Lecture Notes in Computer Science*, pages 142–167. Springer-Verlag, 1987.
16. G. Memmi and G. Roucairol. Linear Algebra in Net Theory. In W. Brauer, editor, *Net Theory and Applications*, volume 84 of *Lecture Notes in Computer Science*, pages 213–223. Springer-Verlag, 1980.
17. Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–580, 1989.
18. Enric Pastor, Oriol Roig, Jordi Cortadella, and Rosa M. Badia. Petri net analysis using boolean manipulation. In Robert Valette, editor, *Application and Theory of Petri Nets 1994*, number 815 in Lecture Notes in Computer Science, pages 416 – 435. Springer-Verlag, 1994.
19. CCITT Recommendations Q.1200. Intelligent networks, final version. Technical report, 1992.
20. M. Raynal. *Algorithms for Mutual Exclusion.* North Oxford Academic, 1986.

21. W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1985.
22. A. Schrijver. *Theory of Linear and Integer Programing.* Series in Discrete Mathematics. Wiley, 1986.

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

Reihe A

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
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?

Reihe A

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

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

Reihe A

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

Reihe A

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

Reihe A

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

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

Reihe A

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

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

This article was processed using the LaTeX macro package with LLNCS style