Deadlock Checking Using Net Unfoldings*

Stephan Melzer and Stefan Römer**

Abstract. McMillan presented a deadlock detection technique based on unfoldings of Petri net systems. It is realized by means of a backtracking algorithm that has its drawback for unfoldings that increase widely. We present an approach that exploits precisely this property. Moreover, we introduce a fast implementation of McMillan's algorithm and compare it with our new technique.

1 Introduction

In the field of static analysis of concurrent systems deadlock freeness is almost always a desirable property. Many research has been carried out to propose methods that check this property [3]. One of these was presented by McMillan in [8]. It is based on net unfoldings of Petri net systems. A net unfolding is class of partial order semantics of Petri nets, also known as branching process [4]. The heuristic used in McMillan's algorithm is particularly good where the unfolding grows more deeply than widely and thereby only few end points of the unfolding (i.e., cut-off points) have to be considered. These kinds of unfoldings correspond to systems with a more deterministic behaviour. In contrast, highly non-deterministic systems tend to yield *wide* unfoldings that slow McMillan's algorithm down.

We introduce an approach that exploits the characteristic of wide unfoldings (i.e., the number of cut-off points is high). Moreover, we present an implementation of McMillan's algorithm and compare both approaches by means of several examples. We use Corbett's benchmark examples [3] as well as McMillan's examples [8] which allows a direct comparison between his LISP implementation and our carried out in C [12].

The paper is organized as follows: In section 2 we give a brief introduction of the basic concepts of Petri nets and net unfoldings. In this section we fall back on the introduction given in [5]. Section 3 presents the deadlock detection method using a linear algebraic approach. In section 4 we give an implementation of McMillan's deadlock algorithm [8]. In section 5 we show some results and compare both approaches. Section 6 serves as a conclusion and gives an outlook on further work. All proofs are presented in appendix A.

* This work was supported by the Sonderforschungsbereich SFB-342 A3 SAM.

** Institut für Informatik, Technische Universität München,
e-mail: {melzers,roemer}@informatik.tu-muenchen.de

2 Basic Definitions

2.1 Petri Nets

A triple (P,T,F) is a *net* if P and T are disjoint sets and F is a subset of $(P \times T) \cup (T \times P)$. The elements of P are called *places* and the elements of T *transitions*. Places and transitions are generically called *nodes*. We identify F with its characteristic function on the set $(P \times T) \cup (T \times P)$. The *preset* of a node x, denoted by ${}^\bullet x$, is the set $\{y \in P \cup T \mid F(y,x) = 1\}$. The *postset* of x, denoted by $x^\bullet$, is the set $\{y \in P \cup T \mid F(x,y) = 1\}$. The generalization on sets of nodes $X \subseteq P \cup T$ is defined as ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$, respectively $X^\bullet = \bigcup_{x \in X} x^\bullet$.
A *marking* M of a net (P,T,F) is a mapping $M : P \to \mathbb{N}$. We identify a marking M with the multiset containing $M(p)$ copies of p for every $p \in P$. A four-tuple $\Sigma = (P,T,F,M_0)$ is a *net system* if (P,T,F) is a net and M_0 is a marking of (P,T,F) (called the *initial marking* of Σ). A marking M *enables* a transition t if $\forall p \in P: F(p,t) \leq M(p)$. If t is enabled at M, then it can *occur*, and its occurrence leads to a new marking M' (denoted $M \xrightarrow{t} M'$), defined by $M'(p) = M(p) - F(p,t) + F(t,p)$ for every place p. A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is an *occurrence sequence* if there exist markings $M_1, M_2, \ldots, M_n$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots M_{n-1} \xrightarrow{t_n} M_n$. M_n is the marking reached by the occurrence of σ, also denoted by $M_0 \xrightarrow{\sigma} M_n$. M is a *reachable marking* if there exists an occurrence sequence σ such that $M_0 \xrightarrow{\sigma} M$.
A marking M of a net is *n-safe* if $M(p) \leq n$ for every place p. A net system Σ is n-safe if all its reachable markings are n-safe.
A net system is called *deadlock-free* if every reachable marking enables at least one transition.
In this paper only nets with a finite number of places and transitions are considered. Moreover we assume that all transitions have neither an empty preset nor an empty postset.

2.2 Occurrence Nets

Let (P,T,F) be a net and let $x_1, x_2 \in P \cup T$. The nodes x_1 and x_2 are in *conflict*, denoted by $x_1 \# x_2$, if there exist distinct transitions $t_1, t_2 \in T$ such that ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, and (t_1,x_1), (t_2,x_2) belong to the reflexive and transitive closure of F. In other words, x_1 and x_2 are in conflict if there exist two paths leading to x_1 and x_2 which start at the same place and immediately diverge (although later on they can converge again). For $x \in P \cup T$, x is in *self-conflict* if $x \# x$.
An *occurrence net* is a net $N = (B,E,F)$ such that:

- for every $b \in B$, $|{}^\bullet b| \leq 1$,
- F is acyclic, i.e. the (irreflexive) transitive closure of F is a partial order,
- N is finitely preceded, i.e., for every $x \in B \cup E$, the set of elements $y \in B \cup E$ such that (y,x) belongs to the transitive closure of F is finite, and
- no element $e \in E$ is in self-conflict.

The elements of B and E are called *conditions* and *events*, respectively. $Min(N)$ denotes the set of minimal elements of $B \cup E$ with respect to the transitive closure of F.
The (irreflexive) transitive closure of F is called the *causal relation*, and denoted by $<$. The symbol $\leq$ denotes the reflexive and transitive closure of F. Given two nodes $x, y \in B \cup E$, we say x *co* y if neither $x < y$ nor $y < x$ nor $x \# y$.

2.3 Branching Processes

Branching processes are "unfoldings" of net systems containing information about both concurrency and conflicts. They were introduced by Engelfriet in [4]. We quickly review the main definitions and results of [4].
Let $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$ be two nets. A *homomorphism* from N_1 to N_2 is a mapping $h: P_1 \cup T_1 \to P_2 \cup T_2$ such that:

- $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$, and
- for every $t \in T_1$, the restriction of h to ${}^\bullet t$ is a bijection between ${}^\bullet t$ (in N_1) and ${}^\bullet h(t)$ (in N_2), and similarly for $t^\bullet$ and $h(t)^\bullet$.

In other words, a homomorphism is a mapping that preserves the nature of nodes and the environment of transitions.
A *branching process* of a net system $\Sigma = (N, M_0)$ is a pair $\beta = (N', h)$ where $N' = (B, E, F)$ is an occurrence net, and h is a homomorphism from N' to N such that

(i) The restriction of h to $Min(N')$ is a bijection between $Min(N')$ and M_0,
(ii) for every $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $h(e_1) = h(e_2)$ then $e_1 = e_2$.

2.4 Configurations and Cuts

The central concept of branching processes is that of a configuration. It describes a possible partial run of an occurrence net. Accordingly, a configuration contains the whole history of the partial run, i.e., all events that have to occur during the run. Moreover, a configuration has to be conflict-free, because events that are in conflict lead to divergence which does not represent a single partial run. More precisely: A *configuration* C of an occurrence net is a set of events satisfying the two conditions: (i) C is causally closed, i.e., $e \in C \Rightarrow \forall e' \leq e: e' \in C$; ($ii$) C is conflict-free, i.e., $\forall e, e' \in C: \neg(e \# e')$.
A set B' of conditions of an occurrence net is a *co-set* if its elements are pairwise in *co* relation. A maximal co-set B' with respect to set inclusion is called a *cut*. Finite configurations and cuts are tightly related. Let C be a finite configuration of a branching process $\beta = (N, h)$. Then the co-set $Cut(C)$, defined below, is a cut: $Cut(C) = (Min(N) \cup C^\bullet) \setminus {}^\bullet C$. In particular, given a finite configuration C the set[3] of places $h(Cut(C))$ is a reachable marking, which we denote by $Mark(C)$.

[3] Remember that this is a multiset.

A marking M of a system Σ is *represented* in a branching process β of Σ if β contains a finite configuration C such that $Mark(C) = M$. It is easy to prove that every marking represented in a branching process is reachable, and that every reachable marking is represented in the unfolding of the net system.
In order to calculate a finite prefix of a branching process, a termination condition is required where the construction of the unfolding can be stopped. If an event is reached during the construction where something would be unfolded which is already represented in the already unfolded prefix, then the construction can be aborted at this event. We call these events *cut-off events*. More precisely: For a given event e, we define the local configuration $[e]$ by the set of all events e' such that $e' \leq e$. Moreover, we call an event a *cut-off event* of a branching process β if β contains a local configuration $[e']$ such that the corresponding markings are equal, i.e., $Mark([e]) = Mark([e'])$ and the configurations $[e']$ is smaller[4] than $[e]$, i.e., $[e'] \subset [e]$.
We say that a branching process β of a net system Σ is *complete* if and only if for every reachable marking M there exists a configuration C in β without any cut-off event such that:

- $Mark(C) = M$ (i.e., M is represented in β), and
- for every transition t enabled by M there exists a configuration $C \cup \{e\}$ such that $e \notin C$ and e is labelled by t.

Figure 1 shows a 1-safe net system (part (a)), a branching process (b) and a complete and finite prefix (c), where e_3, e_5 and e_6 are cut-off events. The homomorphism h is indicated by the corresponding place/transition names inside the nodes. Hereby, the set $\{e_1, e_3, e_4\}$ is a configuration while $\{e_1, e_2, e_3, e_4\}$ has a conflict and is thereby not a configuration.

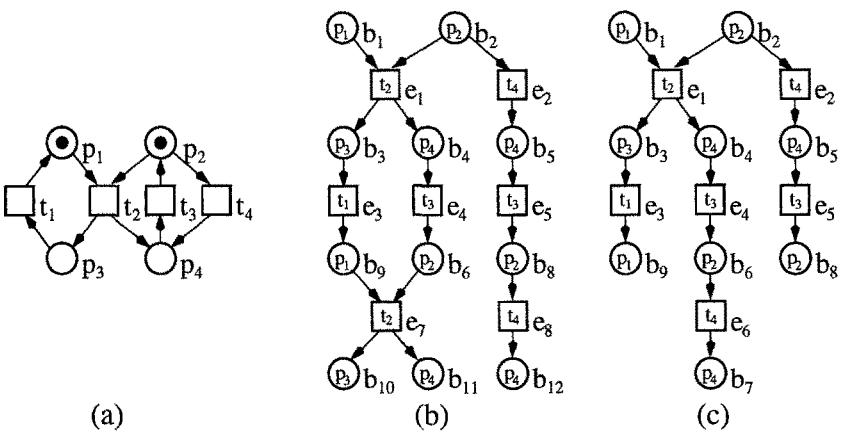

Fig. 1. A net system and two of its branching processes.

[4] In [5] a more precise definition of 'smaller' is given yielding a total order on configurations. For sake of simplicity we use the definition of McMillan [8].

Theorem 1. *Let $\Sigma = (P, T, F, M_0)$ be an n-safe net system. There exists a finite and complete branching process Unf_Σ of Σ.*

Proof. See [8] for the n-safe case and [5] for an improvement of the 1-safe case.

In the sequel we use the term *unfolding* to denote Unf_Σ if Σ is given by the context.

3 A new Method based on Linear Algebra

Each place p of a net is associated with 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 $\nu(\sigma, t)$ the number of occurrences of a transition t in σ, then we can write the *token conservation* equation for p as:

$$M(p) = M_0(p) + \sum_{t \in {}^\bullet p} \nu(\sigma, t) F(t, p) - \sum_{t \in p^\bullet} \nu(\sigma, t) F(p, t).$$

The token conservation equations for every place are usually written in the following matrix form: $M = M_0 + \mathbf{N}.\vec{\sigma}$, where $\vec{\sigma} = (\nu(\sigma, t_1), \ldots, \nu(\sigma, t_m))$ is called the *Parikh vector* of σ, and $\mathbf{N}$ denotes the *incidence matrix* of N, a $P \times T$ integer matrix given by $\mathbf{N}(p, t) = F(p, t) - F(t, p)$.
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 $X := \vec{\sigma}$.

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

The equation $M = M_0 + \mathbf{N}.X$ (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.
The inverse of the implication is not valid for arbitrary net systems, but we will see in the following theorem that the marking equation yields a sufficient condition for reachability in acyclic nets.

Theorem 2. *Let Σ be an acyclic net system and let M be a marking. M is reachable from the initial marking if and only if the marking equation has a nonnegative solution.*

Accordingly, we can use the marking equation for an algebraic representation of the set of reachable markings of an acyclic net system. We call a property $\mathcal{P}$ on markings *linear* if $\mathcal{P}$ can be expressed as a system of linear inequalities $L_\mathcal{P}$. A linear property is valid for all reachable markings of an acyclic net system if and only if the marking equation and $L_{\neg\mathcal{P}}$ have no common solution.

In 1-safe Petri nets each 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 $\sum_{s \in {}^\bullet t} M(s) < |{}^\bullet t|$ for every transition t. We call these constraints *deadlock-inequalities.*

Proposition 3. *Given a 1-safe and acyclic net system $\Sigma = (P, T, F, M_0)$, an integer vector M is solution of the following system of inequalities if and only if M corresponds to a dead reachable marking of Σ:*

$$\begin{aligned} &\textit{Variables: } M, X\textit{: integer.} \\ &\qquad M = M_0 + \mathbf{N}.X \\ &\sum_{p \in {}^\bullet t} M(p) \leq |{}^\bullet t| - 1 \qquad \textit{for all } t \in T \\ &\qquad X \geq 0 \end{aligned}$$

Observe that the application of proposition 3 to cyclic nets yields *just* a superset of reachable dead markings [9], because in cyclic nets the marking equation is not a sufficient condition for reachability. In this case we can interpret the solutions of the marking equation as a linear upper approximation of the state space.
Now we are able to apply proposition 3 to unfoldings of net systems. Since occurrence nets can be seen as acyclic and 1-safe net systems where all places of $Min(N)$ are initially marked we obtain the following theorem.

Theorem 4. *Let $Unf_\Sigma = (N, h)$ with $N = (B, E, F)$ be a finite and complete prefix of the branching process of a given n-safe net system Σ. Σ is deadlock-free if and only if the following system of inequalities has no solution:*

$$\begin{aligned} &\textit{Variables: } M, X\textit{: integer.} \\ &\qquad M = Min(N) + \mathbf{N}.X \\ &\sum_{p \in {}^\bullet e} M(p) \leq |{}^\bullet e| - 1 \qquad \textit{for all } e \in E \\ &\qquad X(e) = 0 \qquad \textit{for all cut-offs } e \\ &\qquad X \geq 0 \end{aligned}$$

The key idea is based on the fact that all markings of Σ are represented in the unfolding Unf_Σ and moreover, the local net structure of each transition is preserved in Unf_Σ. Some dead markings conditional on the finiteness and acyclicity of Unf_Σ do not correspond to dead markings of Σ, because they are located beyond a cut-off event. But these *artificial* deadlocks are not solutions of the inequalities of theorem 4, because we only consider markings of Unf_Σ that can be reached without an occurrence of any cut-off event. Thereby we can identify dead markings of Σ with dead markings of Unf_Σ, and vice versa.

Now we want to focus our attention on implementation issues concerning the feasibility test of theorem 4. Since we know that $\mathbf{N}$ and $Min(N)$ are integers, we can conclude that if X is integer then so is M. Accordingly, we only have to demand that X has to be integer and M can be treated as a rational vector. Since we know that every event can only occur once, we can even demand X to be binary.

Since all variables of X which correspond to cut-off events are equal to zero, we can omit all these variables in the marking equation and thereby in the whole problem. The reader should note that we cannot remove any cut-off event from the prefix at all, because we need the information about its preset in order to formulate the *deadlock-inequalities* correctly.
Using a linear-program-solver like CPLEX [2] we obtain a straightforward implementation of the infeasibility test of theorem 4. Since the complexity of mixed-integer-programs (MIP), i.e., linear programs with rational as well as integer variables, is equal to the complexity of an algorithm to decide if the system of inequalities has a mixed-integer solution, we can make use of the good heuristic implemented in many MIP-solvers. The inherent complexity is NP-complete and more precisely, it is exponential in the number of integer variables.
As we mentioned in the previous paragraph we can formulate theorem 4 just with integer variables for non-cut-off events. Accordingly, the method for searching for a solution of the system of inequalities of the reformulated problem of theorem 4 using MIP-solvers like CPLEX, promises to yield good performance if the number of cut-off events is high regarding the total number of events.

4 McMillan's Method revisited

McMillan presented a way to detect deadlocks in an unfolding by means of looking at the configurations [8]. His observation was that there is no deadlock if each configuration of the unfolding can be extended to a configuration containing at least one cut-off event. In that case, all configurations can be infinitely often expanded.
In other words: The net contains a deadlock if and only if there exists a configuration being in conflict with every cut-off event of the unfolding. For that purpose we need the notion of *spoiler*: The set of all spoilers of a cut-off event e_c is defined as $S_{e_c} = \{e \in E \mid e \# e_c\} \cap ([e_c]^\bullet)^\bullet$, which contains all those events that are directly in conflict with an event from the local configuration of e_c.
McMillan described a branch and bound algorithm to construct a set of spoilers S that is in conflict with every cut-off element of the unfolding, if such a set exists. Then, S can be expanded to a configuration leading to a deadlock of the system. Otherwise, the set S is empty after termination of the algorithm, indicating the net as deadlock-free. This algorithm is sketched in figure 2.
The algorithm is exponential in the size of the unfolding in the worst case because of the backtracking. But by first taking the events with the smallest number of spoilers, this method quickly cuts down the number of possible choices in the average case.
For our implementation of both algorithms, the unfolding and the deadlock detection procedure, we used a special data structure to store and manipulate the nets in the C language [12]. Beside the improvements described in [5] concerning the size of the unfolding by introducing a total order of the configurations, we implemented some heuristic carried out in the combinatorial search of new

```
S := ∅; E_c := set of cut-off events;
while E_c ≠ ∅ do
    e_c := element of E_c with the fewest number of spoilers;
    if e_c has spoilers then
        choose an element e from the spoilers of e_c;
        S := S ∪ {e};
        delete all events in conflict with S
    else
        backtrack to the most recent choice of a spoiler
        and restore the deleted events
    endif
od
```

Fig. 2. McMillan's deadlock-detection algorithm.

events to be inserted in the finite prefix. The latter change drastically reduces the computation time: McMillan measured for the DME [7] consisting of 9 cells nearly 19000 seconds for the calculation of the unfolding in his LISP implementation, where we need only 132 seconds to construct the finite prefix, see Table 2. In this DME example, the size of the unfolding is independent from the chosen order of configurations, [8] or [5].
To check the deadlock property of DME(9), McMillan measured 6600 seconds; in our C implementation of his algorithm we get the result in 702 seconds.

5 Practical Results

In this section we want to compare both approaches. As already mentioned, the linear algebraic approach promises to be faster than McMillan's approach if the number of cut-off events is high w.r.t. the total number of events. In order to check our presumption we use the benchmark examples presented in Corbett's survey [3]. In Table 1 we use a representative subset of these examples to show the performance differences between both approaches w.r.t. the number of cut-off events. We use the abbreviations $\#c$,$\%c$, Unf, DC_{McM} and DC_{MIP} to denote the number of cut-off events, their percentage of cut-offs regarding to $|E|$, the time for the unfolding process[5], for McMillan's deadlock detection algorithm and for our approach, respectively. All results are measured on a SPARC 20/712 with 96 MBytes RAM. We used CPLEX^{TM} (version 3.0) as underlying MIP-solver. Binaries of the programs descripted in the paper are available from the authors. We can directly observe that McMillan's algorithm is faster if the unfolding has only few cut-off events (cp. DPD, RING). In contrast to the examples DPH, ELEVATOR, FURNACE and RW where the new approach overtakes McMillan's method. However, these benchmark results are not strong enough to demonstrate

[5] We used the unfolding proposed in [5] which is smaller or equal to McMillan's.
[6] The example ELEVATOR has a deadlock; all others are deadlock free.
[7] $mem(n)$ indicates the process aborted due to a memory overflow after n seconds.

Problem(size)	States	Original net |P|	|T|	Unfolding |B|	|E|	#c	%c	Time [s] Unf	DC_{McM}	DC_{MIP}
DPD(4)	601	36	36	594	296	81	26	0.12	0.3	2.0
DPD(5)	3489	45	45	1582	790	211	26	0.58	1.9	17.3
DPD(6)	19861	54	54	3786	1892	499	27	3.35	20.2	82.8
DPD(7)	109965	63	63	8630	4314	1129	27	25.03	234.0	652.6
DPH(4)	513	39	46	680	336	117	35	0.15	0.3	1.8
DPH(5)	3113	48	67	2712	1351	547	40	1.48	10.5	42.9
DPH(6)	16897	57	97	14474	7231	3377	47	61.66	1907.6	1472.8
DPH(7)	79927	66	121	81358	40672	21427	53	1946.49	–	–
ELEVATOR(1)[6]	158	63	99	296	157	59	38	0.07	0.0	0.1
ELEVATOR(2)	1062	146	299	1562	827	331	40	0.65	0.9	2.3
ELEVATOR(3)	7121	327	783	7398	3895	1629	42	17.98	18.7	14.5
ELEVATOR(4)	43440	736	1939	32354	16935	7337	43	374.52	492.7	387.8
FURNACE(1)	344	27	37	535	326	189	58	0.12	0.2	0.3
FURNACE(2)	3778	40	65	5139	3111	1990	64	5.60	19.0	18.1
FURNACE(3)	30861	53	99	34505	20770	13837	67	270.02	$mem(811.1)$[7]	1112.5
RING(3)	87	39	33	97	47	11	23	0.02	0.0	0.1
RING(5)	1290	65	55	339	167	37	23	0.07	0.1	1.3
RING(7)	17000	91	77	813	403	79	20	0.23	0.3	17.1
RING(9)	211528	117	99	1599	795	137	17	0.93	1.1	71.2
RW(6)	72	33	85	806	397	327	82	0.08	0.5	0.7
RW(9)	523	48	181	9272	4627	4106	89	3.04	122.3	58.5
RW(12)	4110	63	313	98378	49177	45069	92	279.64	$mem(6004.9)$	24599.9

Table 1. Corbett's examples.

the advantages of both approaches, because all deadlock-free examples can be verified and even faster than via unfoldings by the application of proposition 3 to the original system. Although this corresponds *just* to a semi-decision method, it is strong enough for Corbett's examples. In other words, the gap between the state space and the linear upper approximation obtained by the marking equation is small enough to decide deadlock freeness.

Therefore we give two more case studies. Firstly, we take up the DME [7] example given in McMillan's original paper [8]. Secondly, we modelled the implementation of a readers/writers synchronization [6]. Both examples cannot be proved to be deadlock free by application of proposition 3 to the original system. Even a refinement of the marking equation that is proposed in [9] is not sufficient enough in this context, i.e., the gap between the state space and its upper linear approximation contains dead markings. This fact disables semi-decision methods based on the marking equation or its refinement.

Distributed Mutual Exclusion. In [7], an asynchronous circuit for distributed mutual exclusion (DME) is proposed. McMillan has already shown that the state space grows exponentially in the number of DME-cells while the unfolding increases just quadratically. Due to the fact that the unfolding has only few cut-off events the improvement fails. In Table 2 we list the results. The times DC_{McM} for the deadlock detection seem to increase exponentially, but the increment of DC_{McM} is much more slighter than the increment of DC_{MIP}. The new approach suffers from the small percentage of cut-off events and therefore we interrupted the example DME(7) after 12 hours.

The linear algebraic approach is not appropiated for these kind of systems. Asyn-

chronous circuits do not have such an abundance of non-determinism which is required to yield wide unfoldings.

Problem(size)	States	Original net		Unfolding				Time [s]		
		$\|P\|$	$\|T\|$	$\|B\|$	$\|E\|$	#c	%c	Unf	DC_{McM}	DC_{MIP}
DME(2)	$> 10^2$	135	98	487	122	4	3	0.07	0.07	1.9
DME(3)	$> 10^3$	202	147	1210	321	9	3	0.27	0.50	64.6
DME(4)	$> 10^4$	269	196	2381	652	16	2	1.23	1.67	216.1
DME(5)	$> 10^5$	336	245	4096	1145	25	2	3.92	7.83	1968.3
DME(6)	$> 10^6$	403	294	6451	1830	36	2	10.37	26.43	13678.3
DME(7)	$> 10^7$	470	343	9542	2737	49	2	28.45	97.80	--
DME(8)	$> 10^8$	537	392	13465	3896	64	2	68.16	251.52	--
DME(9)	$> 10^9$	604	441	18316	5337	81	2	131.88	701.74	--
DME(10)	$> 10^{10}$	671	490	24191	7090	100	1	240.57	1801.48	--
DME(11)	$> 10^{11}$	738	539	31186	9185	121	1	420.12	4682.36	--

Table 2. Distributed mutual exclusion-examples.

Readers/Writers Synchronization. In [6], a scalable and bottleneck-free readers/writers synchronization algorithm for shared memory parallel machines is presented. We modelled a 4-bit implementation based on busy waiting semaphors. We used our methods to check deadlock freeness for a setting with one writer and two or three readers (SYNC). The results are depicted in Table 3. In contrast to the DME example we see that the application of the linear algebraic approach turns out to yield better results if the percentage of cut-off events is greater than one third.

Problem(size)	States	Original net		Unfolding				Time [s]		
		$\|P\|$	$\|T\|$	$\|B\|$	$\|E\|$	#c	%c	Unf	DC_{McM}	DC_{MIP}
SYNC(2)	17874	95	239	4007	2162	490	23	9.20	69.0	171.6
SYNC(3)	116446	106	270	29132	15974	5381	34	728.95	26621.7	11985.0

Table 3. Readers/writers-examples.

6 Conclusion

We have introduced a deadlock detection method based on net unfoldings using linear algebraic techniques. Moreover, we have presented an implementation of McMillan's deadlock algorithm and we pointed out the performance gap between McMillan's LISP implementation and our optimized C version. By means of several examples we have pointed out the strong and weak aspects of both approaches. The results show that the larger the percentage of cut-off events is, the more likely the new method will yield better performance than McMillan's. Our future work is to exploit some more CPLEX heuristic in order to speed up our implementation.

Acknowledgements. We thank Javier Esparza for drawing our attention to this problem and Ken McMillan for sending us his LISP sources of the DME generator.

References

1. E. Best and C. Fernández: Nonsequential Processes – A Petri Net View. EATCS Monographs on Theoretical Computer Science 13 (1988).
2. CPLEX 3.0 Manual, CPLEX Corp. (1995).
3. James C. Corbett: Evaluating Deadlock Detection Methods. University of Hawaii at Manoa (1994).
4. J. Engelfriet: Branching processes of Petri nets. Acta Informatica 28, pp. 575–591 (1991).
5. J. Esparza, S. Römer and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. Proc. of *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1055, 87–106 (1996).
6. H. Hellwagner: Scalable Readers/Writers Synchronization on Shared-Memory Machines, Esprit P5404 (GP MIMD), Working Paper (1993).
7. A.J. Martin: The Design of a self-timed Circuit of Distributed Mutual Exclusion. In Henry Fuchs, editor, 1985 *Chapel Hill Confernce on VLSI*, pp. 245–260. Computer Science Press (1985).
8. K.L. McMillan: Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits. Proc. *4th Workshop on Computer Aided Verification*, LNCS 663, 164–174 (1992).
9. S. Melzer and J. Esparza: Checking System Properties via Integer Programming. In Proc. of *European Symp. on Programming*, LNCS 1058, 250–264 (1996).
10. T. Murata: Petri nets: Properties, Analysis and Applications. In Proc. of the IEEE 77(4), pp. 541–580 (1989).
11. M. Nielsen, G. Plotkin and G. Winskel: Petri Nets, Event Structures and Domains. Theoretical Computer Science 13(1), pp. 85–108 (1980).
12. S. Römer: Implementation of a Compositional Partial Order Semantics of Petri Boxes. Diploma Thesis (in German). Universität Hildesheim (1993).

A Proofs

Auxiliaries. Given a multiset or vector X, we denote by $\|X\| = \{x \mid X(x) > 0\}$ the *support* of X. Given a net system $\Sigma = (P, T, F, M_0)$ and a nonnegative transition vector X. We denote by Σ_x the subsystem generated by the transitions of $\|X\|$, their input and output places, i.e., $\Sigma_x = (P_x, T_x, F_x, M_{0x})$ with: $P_x = P \cap ({}^\bullet\|X\| \cup \|X\|^\bullet)$, $T_x = \|X\|$, $F_x = F \cap ((P_x \times T_x) \cup (T_x \times P_x))$, $M_{0x} = M_0 \cap P_x$. We use $\mathbf{e_t}$ to denote the transition vector with $\mathbf{e_t}(t) = 1$ and $\mathbf{e_t}(t') = 0$ for all $t' \neq t$.

Proof of theorem 2.

The following proof is a more detailed realization of the proof sketch given in [10]. It is well known that the marking equation is a necessary condition for reachability. Hence, we only have to show the sufficiency. Suppose that there exists a solution X of the marking equation. Now we consider the net system Σ_x. It is obvious that Σ_x is also acyclic and moreover if M_x is reachable from M_{0x} then it is also reachable by M_0. We show that M_x is reachable from M_{0x} by induction over the length of X, i.e., $n = \sum_{t \in \|X\|} X(t)$.

Induction Base: $n = 1$. Let t be the transition with $X(t) = 1$. Due to $F_X(t, p) = 0$ for all $p \in {}^\bullet t$, we obtain from the marking equation that ${}^\bullet t \subseteq M_{0x}$ and thereby

t is enabled at marking M_{0x}. Hence, $M_{0x} \xrightarrow{t} M_x$.
Induction Step: $n+1$. We assume a transition t which is enabled at marking M_{0x}. The existence of such a transition is guaranteed by the acyclicity of Σ_x and by the fact that Σ_x is nonempty. Hence the occurrence of t yields a marking M'_x. Let $X' = X - \mathbf{e_t}$ be the vector where transition t fires one time less. Now we can conclude that $M_x = M_{0x} + \mathbf{N}.X = M_{0x} + \mathbf{N}.(X' + \mathbf{e_t})$. The last equation can be splitted into $M'_x = M_{0x} + \mathbf{N}.\mathbf{e_t}$ and $M_x = M'_x + \mathbf{N}.X'$. Since $\sum_{t \in \|X\|} X'(t) = n$ we can apply the induction hypothesis and get directly that M_x is reachable from M'_x. Together with the fact that $M_{0x} \xrightarrow{t} M'_x$ we finally get the reachability of M_x from M_{0x}. □

Proof of proposition 3.
Due to theorem 2 and the acyclicity of Σ, the set of reachable markings is represented by the solutions of the marking equation. Moreover, due to the 1-safeness we can express the fact that a transition t is enabled at a marking M by the linear constraint $\sum_{p \in {}^\bullet t} M(p) \geq |{}^\bullet t|$. Hence, a dead marking, i.e. a marking that enables no transition can be described by $\sum_{p \in {}^\bullet t} M(p) < |{}^\bullet t|$ for all $t \in T$ that is logically equivalent to $\sum_{p \in {}^\bullet t} M(p) \leq |{}^\bullet t| - 1$ since we are using integer variables for M. □

Proof of theorem 4.
We consider the net system $\Sigma' = (N, Min(N))$ with $N = (B, E, F)$. Since N is an occurrence net, Σ' is acyclic. Moreover, $Min(N)$ is a 1-safe marking and due to the acyclicity of N and the fact that each place has at most one incoming arc we can conclude that Σ' is a 1-safe system. Due to theorem 1 we know that each marking M of Σ is represented by a finite configuration C with $Mark(C) = M$ where C does not contain any cut-off event. Moreover, we know that the set of cuts corresponding to finite configurations without cut-off events coincides with the set of markings reachable from $Min(N)$ without occurring any cut-off event. Therefore it remains to show that a deadlocked system Σ' implies the existence of a finite configuration without cut-off events which corresponds to a cut where no event is enabled, and vice versa.
($\Rightarrow$) Suppose that the net system Σ is not deadlock-free, then there exists a reachable marking M such that no transition is enabled at M, i.e., for all $t \in T$, ${}^\bullet t \not\subseteq \|M\|$. Due to theorem 1 we know that there exists a finite configuration C without any cut-off events such that for all $t \in T$, ${}^\bullet t \not\subseteq \|Mark(C)\|$ still holds. Because $h(E) \subseteq T$, we can conclude that for all $e \in E$ ${}^\bullet h(e) \not\subseteq \|Mark(C)\|$ is also satisfied. Due to the monotony of h w.r.t. set inclusion we obtain that the preset of no transition is a subset of $Cut(C)$. Hence all events which are no cut-off event are disabled at $Cut(C)$.
($\Leftarrow$) Suppose the existence of a configuration C without cut-off events such that no event is enabled at marking $Cut(C)$. Then we can conclude that there exists no event $e \in E \setminus C$ such that its local configuration can be embedded in an extension of C by E, i.e., $[e] \not\subseteq C \cup \{e\}$. Hence there exists no configuration $C \cup \{e\}$ for an arbitrary event e. Due to theorem 1 we obtain that no transition $t \in T$ exists such that ${}^\bullet t \subseteq \|Mark(C)\|$. This means that the corresponding reachable marking $M = Mark(C)$ is a dead marking. □