Int J Adv Manuf Technol (1998) 14:750–759

The International Journal of
Advanced Manufacturing Technology

Deadlock Prevention Using Petri Nets and their Unfoldings

A. Taubin*, A. Kondratyev* and M. Kishinevsky†

*The University of Aizu, Aizu-Wakamatsu, Japan; †Strategic CAD Labs. Intel Corp. Hillsboro, OR, USA

Unfoldings of Petri nets (PN) provide a method for the analysis of concurrent systems without restoring the state space of a system. This allows one to overcome the "state explosion" problem. Many properties of the initial PN (boundedness, safety, persistency and hazards) can be checked by constructing the unfolding. A deadlock prevention procedure first detects deadlocks using an unfolding. Then, the first method reduces the unfolding to a set of deadlock-free subunfoldings that cover all live behaviours. The second method uses a direct transformation at the level of the original PN. The methods are implemented as subroutines in the Berkeley program SIS. Although the deadlock detection problem is known to be NP-complete, experimental results show that for highly parallel specifications deadlock prevention by unfoldings is typically more efficient than deadlock prevention based on symbolic BDD (binary decision diagrams) traversal of the corresponding reachability graph.

Keywords: Deadlock prevention; Petri nets; Unfoldings

1. Introduction

In this paper a Petri net (PN) model is used to solve the problem of deadlock prevention. Contrary to Banaszak and Krogh [1] and Ezpleta et al. [2], the specific features of the processes modelled by PNs are not relied on here. They can be of any nature: control processes, manufacturing processes, etc. Thus, the task of deadlock prevention is stated for the class of general PNs.

Following [1], a straightforward (brutal) approach based on the excluding of deadlock states from the reachability graph of PN markings was investigated. The size of the reachability graph is an inherent barrier for applying such an approach to large PNs. There are several well-known techniques to avoid the "state explosion problem". Methods based on partial orders avoid generation of the corresponding reachability graph [3,4]. Instead of the reachability graph, a finite prefix (called unfolding) of the equivalent occurrence net (acyclic net where all places have not more than one input transition) is generated.

The problem of deadlock detection by unfolding was first investigated in [3]. Even though the problem is NP-complete, it was shown that it is readily solved in practice for fairly large PNs.

This paper suggests two methods for transforming a PN unfolding that aims at excluding deadlocks (if any).

Both methods first detect a deadlock using McMillan's algorithm [3]. Then, the first method reduces the unfolding to a set of deadlock-free subunfoldings that cover all live behaviours. The modified unfolding is further folded to a cyclic deadlock-free PN. For the class of reinitialised processes (in which every cycle in a reachability graph passes through the home marking – this requirement is somewhat similar to the conditions on sequential processes with resources [2] or production Petri net model [1]) it is shown that our method for deadlock prevention gives a solution that preserves all weakly live (i.e. cyclic) behaviours of the original PN.

The technique of a deadlock prevention used in the second method is different: instead of manipulating deadlock-free subunfoldings we use here direct transformation at the level of the original PN. This transformation is based on inserting additional places and transitions to control the switching of alternative behaviours that may lead to a deadlock. The final cyclic deadlock-free net is equivalent to the initial net in the following weak sense: complete cyclic behaviours retain their "most important" representatives ("reduced live equivalence"); however, concurrency can be reduced and some live and non-live behaviours can be removed. In this procedure of deadlock prevention, unfoldings serve as a tool for detecting deadlocks and for constructing ordering relations between places and transitions.

The paper is organised as follows. Sections 2 and 3 introduce PNs and unfoldings. In Section 4 a deadlock prevention by a reachability graph of a PN is considered. The shortcomings of this method are discussed and a deadlock prevention based on partial orders is motivated. Section 5 presents the first method for deadlock prevention using unfolding. Several useful properties are introduced to guide the procedure of deadlock prevention. Section 6 presents the general idea for deadlock prevention using PN transformation (second method). Section 7 presents experimental results, including about 30 typical examples from

Correspondence and offprint requests to: Dr A. Taubin, The University of Aizu, Aizu-Wakamatsu, 965-80, Japan. E-mail: taubin@u-aizu.ac.jp

manufacturing systems. Section 8 gives a conclusion and recommendations for future research.

2. Basic Notions

Let $N = <P,T,F,m_0>$ be a Petri net (PN) [5], where P is the set of places, T is the set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and m_0 is the initial marking.

A transition $t \in T$ is enabled at marking m_1 if all its input places are marked. An enabled transition t may fire, producing a new marking m_2 with one less token in each input place and one more token in each output place. The new marking m_2 can again make some transitions enabled. The set of all markings reachable in N from the initial marking m_0 can be represented by the reachability graph (RG). It is also possible to discuss the sequences of transitions that fire under the markings reachable from the initial marking m_0. Such sequences of transitions will be called *feasible traces* or *traces*. A trace is called cyclic if, starting from some point, it repeats itself (hence, a cyclic trace is infinite).

If no transition is enabled at marking m this marking is called *terminal*. A transition, t, is *strongly live* if, for every reachable marking m, there exists marking m' reachable from m which enables t. In other words a transition is strongly live if it can always occur again. A transition is *weakly live* if it belongs to a cyclic feasible trace. In other words, a transition is weakly live if it can cyclicly occur an infinite number of times; however, the system can also evolve along other traces which may never enable this transition. In deadlock prevention, we aim to preserve the weak liveness, i.e. if a transition is weakly live in the initital PN, then it should be weakly live in the transformed deadlock-free PN.

The set of input places of transition t is denoted by $\bullet t$ and the set of output places by $t\bullet$. Similarly, $\bullet p$ and $p\bullet$ stand for the sets of input and output transitions of place p. A place p is called a *choice* place if it has more than one output transition.

A PN is *acyclic* if there are no cycles in the graph of the PN. In an acyclic PN there are some places without input transitions. It is assumed that all these places are initially marked with one token and no other places are initially marked. The examples of a cyclic and an acyclic PN are shown in Fig. 1(*a*,*b*).

A PN is called:

k-bounded, if, for every reachable marking, the number of tokens in any place is not greater than k (the place is called k-bounded if for every reachable marking the number of tokens in it is not greater than k), and *bounded*, if there is a finite k for which it is k-bounded.

safe, if it is 1-bounded (a 1-bounded place is called a safe place).

A transition t_i is in a *direct conflict* with another transition t_j if t_i and t_j are enabled in a reachable marking m and t_i becomes disabled after firing t_j.

Definition 2.1 (Ordering Relations). Let $N = <P,T,F>$ be an acyclic PN and $x_1, x_2 \in P \cup T$.

x_1 precedes x_2 (denoted by $x_1 \Rightarrow x_2$) if (x_1,x_2) belongs to the reflexive transitive closure of F, i.e. there is a path in the graph of a PN between x_1 and x_2. Note that the preceding relation is reflexive, i.e. $\forall x : x \Rightarrow x$.

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 $t_1 \Rightarrow x_1$, and $t_2 \Rightarrow x_2$. If $x \# x$ (where $x \in P \cup T$), then x is in self-conflict.

x_1 and x_2 are concurrent (denoted by $x_1 \| x_2$), if they are neither in precedence, nor in conflict.

Let us consider the introduced relations for the example of an acyclic PN in Fig. 1(b). Directly from Definition 2.1 one can find that $b \Rightarrow c \Rightarrow d$, $p1 \| p2$, $p1 \| p4$, $a \| b$, $a \# e$.

Acyclic PNs are easier to analyse because their properties can be completely specified by ordering relations [6]. This suggests an effective way for the analysis and/or manipulation of a general PN: first the cyclic PN is unfolded into an equivalent acyclic description and then the finite prefix of the acyclic net is used.

The equivalence between two PNs, on the basis of set of traces they realise, is defined below.

Definition 2.2 (Trace Equivalence). PNs $N1$ and $N2$ with sets of transitions $T1$ and $T2$ are trace equivalent with respect to partition r of $\{T1\} \times \{T2\}$ iff for any trace $s = t_1,\ldots,t_k,\ldots$ feasible in $N1$ there exists a trace $q = \tau_1,\ldots,\tau_k,\ldots$, feasible in $N2$ such that $\tau_i \; r \; t_i$ for every $i = 1,\ldots,k,\ldots$ and vice versa [7].

3. Unfolding

Definition 3.1 (Occurrence Net). An occurrence net is an acyclic net $N = <P,T,F>$ in which every place $p \in P$ has at most one input transition $|\bullet p| \leq 1$.

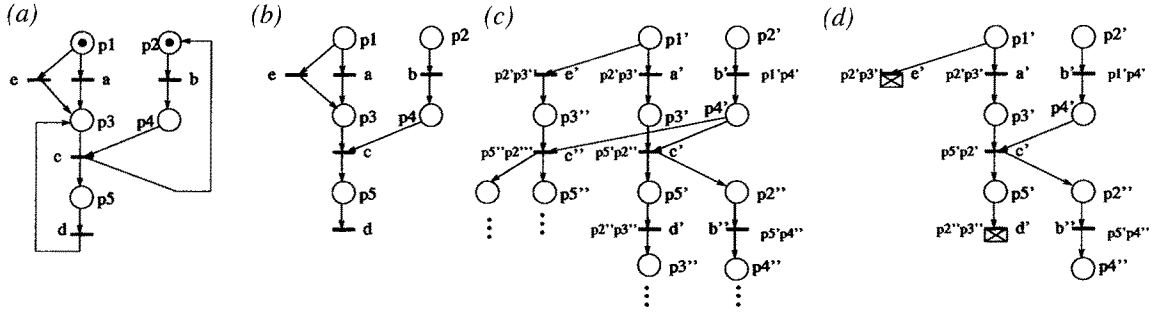

Fig. 1. (*a*) Cyclic, (*b*) acyclic PNs, (*c*) occurrence net, and (*d*) unfolding.

This definition corresponds to the definition of occurrence nets given in [4]. Figure 1(*c*) shows an occurrence net for the PN from Fig. 1(*a*). Each transition t_i of the initital PN has a set of corresponding transitions in the occurrence net t_i', t_i'', t_i''', ..., that are called *instantiations* of t_i. Similarly, for each place p_j the occurrence net contains the set of the corresponding instantiations p_j', p_j'', p_j''', ... It can be shown that under the partition r that associates each transition t of PN with its instantiations t', t'', ..., the original PN is trace equivalent to its occurrence net. Further, we will refer to any object in the occurrence net equivalent to the object in the original cyclic PN by adding one or more apostrophes (or by adding a superscript) to its name. For example, t' and t are the corresponding transitions in an occurrence net and a PN, m' is a marking in the occurrence net and m is the corresponding marking in the PN, etc. Given an occurrence of a transition t' (a place, p', or a marking m'), the original transition t (the place, p, or the marking m) is called a *prototype* for this occurrence.

Although, the occurrence net for a cyclic PN can be infinite, it is always possible for a bounded PN to truncate the occurrence net up to a finite "complete" subgraph which possesses all the information about the original PN (e.g. contains all reachable markings of the original net). This complete subgraph is called an *unfolding*.

For truncating an occurrence net let us introduce several notions.

Definition 3.2 (Configurations) [3].

A set of transitions $C' \subset T'$ is a *configuration* in an occurrence net if: (i) for each $t' \in c'$ the configuration C' contains t' together with all its predecessors; (ii) C' contains no mutually conflicting transitions.

The minimal configuration that contains t' and all the transitions preceding t' is called a *local configuration* of the transition t' (denoted $\{\Rightarrow t'\}$).

Each configuration C' corresponds to a marking that is reachable from m_0 after all the transitions from C' have been fired. This marking is called the *final marking* of C' and is denoted by $FM(C')'$ in [3].

Definition 3.3 (Final and Basic Markings). Let C' be a configuration of an occurrence net. A final marking of C', denoted $FM(C')'$, is a marking reachable from the initial marking after all transitions from C' and only those transitions are fired. A final marking of a local configuration of t' is called a *basic marking* of t' and denoted $BM(t')'$.

In the occurence net of the PN from Fig. 1(*c*) the local configuration $\{\Rightarrow c'\}$ for transition c' is equal to $\{a',b',c'\}$. The basic marking for c' is $BM\ (c')' = \{p5',p2''\}$. Marking $\{p3',p4'\}$ does not correspond to any local configuration; however, it corresponds to the configuration $\{a',b'\}$.

Occurrence nets are truncated by the *cut-off transitions* (Fig. 1(*d*)). Criteria for choosing these transitions for different classes of PNs were suggested in [3,4,6]. The necessary condition for a transition t_i' to be a cut-off is that the basic marking of t_i' repeats the basic marking of some other transition t_j' that has been generated earlier in an occurrence net (t_j' is called an *image* of t_i'). Intuitively, it means that in traversing along the reachable states of PN by firing t_i, the marking that has already been reached afer firing t_j is produced. As shown in [3,4,6] more precautions have to be taken in determining cut-offs. In further discussions, it is assumed that a cut-off criterion is complete, i.e. the unfolding represents all the reachable markings of the original PN.

4. Deadlock Prevention by Reachability Graph

The idea of a deadlock prevention can be formulated as follows: for a given specification A, construct a specification B that will be deadlock-free and equivalent to A on non-deadlock behaviours.

The informal term here is the "equivalence on non-deadlock behaviours". Clearly this notion of equivalence must differ from the trace equivalence because the goal of a deadlock prevention is to eliminate those traces that lead to deadlocks. Moreover, it is also undesirable to introduce new behaviours (traces) in a modified deadlock-free process. The following notion of equivalence corresponds to preserving the weak liveness.

Definition 4.1 (Live Equivalence). PN $N2$ with a set of transitions $T2$ is live equivalent to PN $N1$ with a set of transitions $T1$ with respect to partition r of $\{T1\} \times \{T2\}$.

1. If for any trace, $s = t_1,\ldots,t_k,\ldots$, feasible in $N2$ there exists a trace $q = \tau_1,\ldots,\tau_k,\ldots$, feasible in $N1$ such that $\tau_i\ r\ t_i$ for every $i = 1,\ldots,k,\ldots$
2. If for any infinite trace $p = \tau_1,\ldots,\tau_k,\ldots$, feasible in $N1$ there exists an infinite trace $s = t_1,\ldots,t_k,\ldots$ feasible in $N2$ such that $T_i\ r\ \tau_i$ for every $i = 1,\ldots,k,\ldots$

Informally condition 1 of Definition 4.1 states that all traces of $N2$ form a subset of the set of traces of $N1$, and condition 2 guarantees that $N2$ preserves all infinite traces of $N1$, i.e. cyclic behaviours of $N1$ and $N2$ coincide.

Definition 4.1 suggests a straightforward method for obtaining a deadlock-free PN $N2$ by the original PN $N1$. It is based on generating the reachability graph (RG) of $N1$, removing from it all the terminal markings and synthesising a deadlock-free PN $N2$ corresponding to the modified RG.

It is easy to see that by an iterative removal of terminal markings with all their input arcs, one cannot remove any infinite trace from $N1$. Therefore, all the cyclic behaviours of the original process are preserved in the final RG D. After removing all terminal markings, a new PN $N2$ corresponding to the modified RG D could be synthesised.

This approach has a few shortcomings: an exponential explosion of markings in the generated RG; a problem of PN synthesis is also computationally expensive; a modified deadlock-free PN can be much more complex than the original PN.

The first two problems can be practically solved for many applications by the existing tools (e.g. *petrify*) [8]. However, the size of the reachability graph is an inherent barrier for these methods.

The third problem is even more important. To illustrate this, let us consider a well-known dining philosophers' problem [5]. Figure 2(*a*) shows a PN for three dining philosophers.

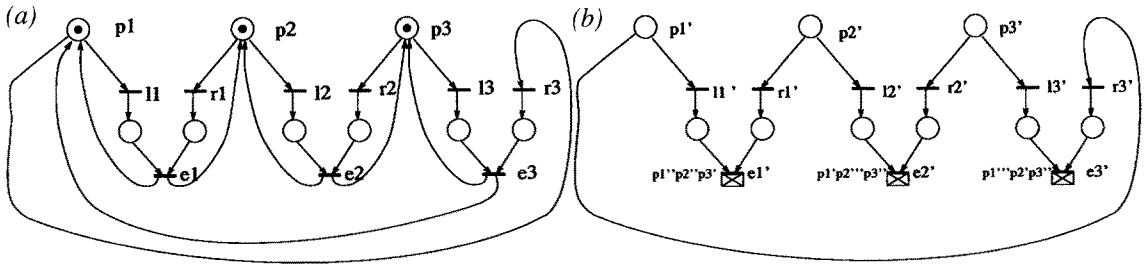

Fig. 2. (*a*) PN for three dining philosophers, and (*b*) its unfolding.

Transitions are interpreted as follows: r_i, (l_i), philosopher i takes righthand (lefthand) fork; and e_i, philosopher i is eating. This PN has two terminal states: all philosophers take lefthand fork and all philosophers take the righthand forks. These markings are removed from the reachability graph and then a PN can be extracted from the modified RG (with the help of *petrify*, for example). The result is a deadlock-free net that has multiple labels of transitions (e.g. two transitions labelled with e_3) and its overall complexity is much higher than the complexity of the PN from Fig. 2(*a*).

Individual markings from an RG of a net are removed. Since separate marking has no direct correspondence with places and transitions of a PN, this strategy can lead to a significant modification of the original net. This explains the problem of a deadlock-free net explosion and provides the major motivation for developing deadlock preserving methods at the level of PNs and their unfoldings.

5. Deadlock Prevention by Unfoldings

Let PN N specify the original process. The goal of a deadlock prevention by PN is to forbid the firing of transitions that lead to terminal markings. In an unfolding N' it corresponds to: finding in N' a configuration C' corresponding to a terminal marking m and removing from C' transitions leading to a deadlock m. A resulting deadlock-free PN can be obtained by folding of a modified unfolding.

Let us consider each step in detail.

5.1. Deadlock Detection by Unfoldings

The problem of a deadlock detection by unfoldings was first investigated in [3]. Here the results from [3] will be reformulated and extended.

Definition 5.1 (Maximal Configuration). A configuration C' of unfolding N' is said to be maximal if it is not a subset of any other configuration of N'.

In order words, no transition of N' can be added to C' without violating the conditions for a configuration.

Property 5.1. Marking m of PN N is terminal iff in the unfolding N' a configuration C' with a final marking m' is (i) maximal and (ii) contains no cutoffs. Proofs of this and subsequent properties and propositions are given in [9].

Definition 5.2 (Maximal Spoiler). A transition $t1' \in N'$ is called a *maximal spoiler* for a transition $t2' \in N'$ if: (i) $t1'$ is in conflict with $t2'$ ($t1'\#t2'$); (ii) there is no transition $t3' \in N'$ such that: $t3'\#t2'$ and $t3' \Rightarrow t1'$.

Informally, a maximal spoiler for t' is the "first" transition that is in conflict with t'. One transition can have several maximal spoilers.

It follows from Property 5.1 and Definition 5.2 that a configuration of a cyclic net that corresponds to a terminal marking must contain a maximal spoiler for each cut-off [3]. Indeed, if a configuration contains a transition $t1'$ that is in conflict to some considered transition $t2'$, then by definition it contains the transition $t3'$, $t3' \Rightarrow t1'$, that is a maximal spoiler for $t2'$.

If an unfolding contains no cut-offs, then according to Property 5.1 any maximal configuration of an unfolding corresponds to a terminal marking. However, such PNs contain no cyclic behaviour and these specifications are not of much interest for a deadlock prevention.

The algorithm from [3] uses branch and bound techniques to find a set T_s of maximal spoilers in a configuration corresponding to a terminal marking (if any).

In [3] it was proved that the problem of a deadlock detection by unfolding is NP-complete. However, it was shown that this problem is readily solved in practice even for very large unfoldings.

The algorithm from [3] gives a configuration which corresponds to one terminal marking. That is, our procedure of prevention, iteratively refines the unfolding, such that at each iteration, at least one deadlock is removed.

5.2 Folding

By merging in an unfolding all instantiations of the same transition (place) into one transition (place), an original PN can always be obtained.

However, in deadlock prevention by the unfolding approach, a folding is applied after removing deadlocks. Hence, some instantiations of transitions and places are removed from the unfolding. In such a case, different instantiations of the same transition (place) of the original PN may no longer be equivalent and therefore the complete folding is incorrect.

For folding of modified unfoldings we will propose another procedure which folds an acyclic net by basic markings of cut-offs. In this case, places from the basic marking of a cut-off transition are merged with the places of a cut-off image.

Algorithm 1. Folding into a Cyclic PN

1. Let N' be an unfolding, T'_c be the set of its cut-off transitions.
2. **while** T'_c is non-empty **do**
3. Delete from T'_c its first cut-off transition t'.
4. Find an image transition $t1'$ such that $BM(t') = BM(t1')$.
5. Set a one-to-one correspondence between each place $p^i \in BM(t')$ and $p^j \in BM(t1')'$ (p^i and p^j are instantiations of the same place p of the original PN).
6. Delete from N' all transitions and places that succeed any place from $BM(t')'$.
7. Merge instantiations of the same places from $BM(t')'$ and $BM(t1')'$, denote the obtained net as $N1'$; $N' = N1'$; **end do**.
8. Mark with a token all places that correspond to places of unfolding without input arcs.

The following proposition proves the soundness of Algorithm 1. It states that the net obtained after folding will be equivalent to the original PN.

Proposition 5.1. The PN $N1'$ obtained as a result of Algorithm 1 is trace equivalent to the original PN N under the partition that associates each transition $t \in N$ with its instantiations in $N1'$.

5.3 Deadlock Removal

Let us first confine ourselves with a class of safe PNs. This will not restrict the applicability of the approach because any bounded PN can be represented by a trace equivalent safe PN using the so-called pipelining of unsafe places [10]. For a safe PN there is an unfolding method which permits one image per cut-off [4]. Therefore, the result of the folding procedure is uniquely defined.

A deadlock in an unfolding corresponds to a configuration with a terminal final marking (such a configuration will be also called *terminal*). A deadlock can be excluded by removing from a terminal configuration each maximal spoiler t' (together with succeeding transitions and places) for one of the cut-offs $t1'$. This allows addition to a configuration of the cut-off transition $t1'$, which was spoiled by t', and permits the process to go beyond deadlock by firing cut-off $t1'$. Note that if a cut-off has several maximal spoilers in the terminal configuration, all of them must be removed from the unfolding to allow the cut-off firing.

Let $T1'$ be a set of transitions in an unfolding. A modified unfolding, obtained after removal of $T1'$ and the corresponding places, is denoted by $N'/T1'$. A folding of $N'/T1'$ by Algorithm 1 is denoted by $N^f/T1'$.

A deadlock removal procedure should satisfy the following requirements.

1. The live equivalence between the original PN N and the $N^f/T1'$ must be preserved.
2. New deadlocks should not appear in $N^f/T1'$.

The following properties give sufficient conditions for checking liveness of a transition in terms of unfolding.

Property 5.2. If transition t' of the unfolding N' does not precede any cut-off, then PN N^f/t' is live equivalent to the original PN N.

The proof follows trivially from the following observation: if t' does not procede any cut-off, its removal cannot influence any infinite trace in the occurrence net.

Property 5.2 gives a sufficient condition for choosing good candidates for removal among spoilers — all maximal spoilers which do not precede cut-offs can be removed without violating live equivalence.

If no spoilers satisfy the condition of Property 5.2, finer consideration of the liveness of maximal spoilers and cut-offs is necessary to make a proper choice.

A reachable marking M of PN N is called *cyclic* if there exists trace q such that $M \stackrel{q}{\Rightarrow} M$. Note that a non-live transition can produce a cyclic marking. Figure 1(*d*) shows the unfolding of PN from Fig. 1(*a*). Transition a' has a cyclic basic marking $p2p3$, however a can fire only once. The problem of the weak liveness of a transition is NP-complete. Therefore, the following sufficient condition for checking the weak liveness will be used. This condition will be used for selecting weakly live cut-offs whose spoilers can be removed from the unfolding without a risk that new terminal configurations may appear.

Property 5.3. Let transition $t1'$ be a cut-off in an unfolding N' and its image $t2'$ precede $t1'$ ($t2' \Rightarrow t1'$) then any transition t' such that $t' \not\Rightarrow t2'$ and $t' \Rightarrow t1'$ is a weakly live transition.

Property 5.2 gives a sufficient condition for a non-liveness of a transition. If this transition is a maximal spoiler, then it can be safely removed from a terminal configuration without violating the live equivalence. Property 5.3 gives a sufficient condition for the weak liveness of a transition. Adding a weakly live cut-off to a terminal configuration never produces a new deadlock.

The following algorithm removes deadlocks from an original PN based on these properties.

Algorithm 2. Removal of Deadlocks from PN

1. Let N' be an unfolding, and T'_c the set of its cut-off transitions.
2. **while** N' is not deadlock-free **do**
3. Find a terminal configuration C'; let T'_s be the set of maximal spoilers for T'_c in configuration C'.
4. Find in T'_s all the spoilers $T1'_s$ that do not precede any cut-off; $T'_s = T'_s - T1'_s$.
5. Remove from C' all the spoilers $T1'_s$ and their successors.
6 If any cut-off can be added to modified C' **goto** step 10 with $N' = N'/T1'_s$.
7. Find the set T'^r_c of weakly live cut-offs (by Property 5.3).
8. **if** $T'^r_c = \emptyset$ **then** $T'^r_c = T'_c$ (if Property 5.3 fails we consider all cut-offs as potentially weakly live ones).
9. Choose in T'^r_c a cut-off with the fewest number of spoilers in T'_s (denote these spoilers $T1'_s$); **goto** step 5.
10. **end do**.

Example. Dining Philosophers. The unfolding is shown in Fig. 2(*b*).

The set of its cut-offs is $T'_c = \{e1', e2', e3'\}$, one of the terminal configuration is $C' = \{r1', r2', r3'\}$. The set of its maximal spoilers is $T'_s = \{r1', r2', r3'\}$. All transitions $r1', r2', r3'$ precede some of the cut-offs and therefore one cannot apply Property 5.2 to remove any of them. All the basic markings of cut-offs repeat the initial marking and according to Property 5.3 all cut-offs are weakly live. Transitions $e1', e', e3'$ have the same number of spoilers in T'_s and one can choose any of them to be added to C', e.g. $e1'$. Cut-off $e1'$ is spoiled in C' by transition $r3'$ and for adding $e1'$ transition $r3'$ has to be removed from the unfolding together with a succeeding transition $e3'$. The result of this procedure is shown in Fig. 3(*a*).

In this modified unfolding there exists another terminal configuration $C1' = \{l1', l2', l3'\}$ with a set of maximal spoilers $T'_s = \{l1', l2', l3'\}$. Among these spoilers $l3'$ does not precede any cut-off and according to Property 5.2 it can be safely removed. After this, a cut-off transition $e2'$ can be added to $C1'$ and the modified unfolding becomes deadlock-free. The PN obtained after its folding is shown in Fig. 3(*b*). Our procedure successfully ensures deadlock prevention, however it "kills" one of the philosophers.

This example clearly shows the shortcoming of Algorithm 2: it does not preserve the live equivalence because by removing the spoilers one can also remove some live transitions. The following section shows how the live equivalence can be preserved for the so-called reinitialised processes.

5.4 Live Equivalence

Assume that the initial marking m_0 of a PN is contained in every cycle of the corresponding RG. Such PNs are called *reinitialised* and they are very common in manufacturing systems [1,2], which is one of the major application fields for a deadlock prevention. Reinitialised processes have the following attractive properties for our deadlock prevention algorithm:

Any deadlock-free process obtained by Algorithm 2 from reinitialised PN N will be also a reinitialised PN with respect to the same initial marking m_0. Indeed, by Proposition 5.1 no new traces can appear in the folded PN, thus all cycles of its RG pass through the same marking m_0.

The weak liveness is easy to determine in a reinitialised PN. Property 5.3 works well for such PNs since they always contain cut-off transitions with basic markings equal to m_0.

A reinitialised PN can be covered with a set of deadlock-free components. Each deadlock-free component specifies a partial behaviour of a PN, their compositions give a complete weakly live behaviour. The following algorithm extracts a cover set of deadlock-free components from an original PN.

Algorithm 3. Extraction of Components

1. Let N' be an unfolding and T'^{c}_{c} the set of its live cut-offs; find a deadlock-free component $N1'$ using Algorithm 2.
2. $T'^{c}_{c} = T'^{c}_{c}$, transitions $(N1')$ (find live cut-offs that are not present in $N1'$).
3. **while** $T'^{c}_{c} \neq \emptyset$ **do** Take first $t' \in T'^{c}_{c}$; find a deadlock-free component $N2'$ which will cover t' (use Algorithm 2 with a restriction that t' is chosen to be added to a terminal configuration).
4. $T'^{c}_{c} = T'^{c}_{c}$, transitions $(N2')$ **end do**.

Algorithm 3 produces a set of deadlock-free components that cover all weakly live cut-offs. These components covers all the weakly live behaviours of the PN.

Since the initial PN is reinitialised, all the cover components are also reinitialised PNs with the same initial markings which correspond to the initial marking of the PN, m_0. Figure 4(*a*) presents three deadlock-free components with the initial markings corresponding to marking $p1p2$ of the original PN. The composition of such components is illustrated in Fig. 4(*a*).

The new transitions $i1$, $i2$ and $i3$ are those with the only input place $p0$ which enters the initial markings in components $N1$, $N2$ and $N3$. Components are chosen non-deterministically by a free choice. After a component receives its initial marking it can either fire some internal transitions or through firing transitions $o1$, $o2$, $o3$ return the token to the place $p0$. This choice is again made non-deterministically. After each cycle one can switch between different deadlock-free components. As the set of components covers all cyclic behaviours of the original PN N the result of this transformation will be a live equivalent to N.

Example. Dining Philosophers. Continued.

Let us apply the above technique to the dining philosophers problem. The first deadlock-free component obtained by Algorithm 2 is shown in Fig. 3(*b*). This component has only one missing live cut-off transition, $e3'$. By giving priority for the cuf-off $e3'$ we can obtain another deadlock-free component which will be similar to the component in Fig. 3(*b*) except that philosopher number 1 will be "killed" in it. These two components contain all the live cut-offs, and so form the complete cover set. Their composition will give all the weakly live behaviours of an original PN and is shown in Fig. 4(*b*).

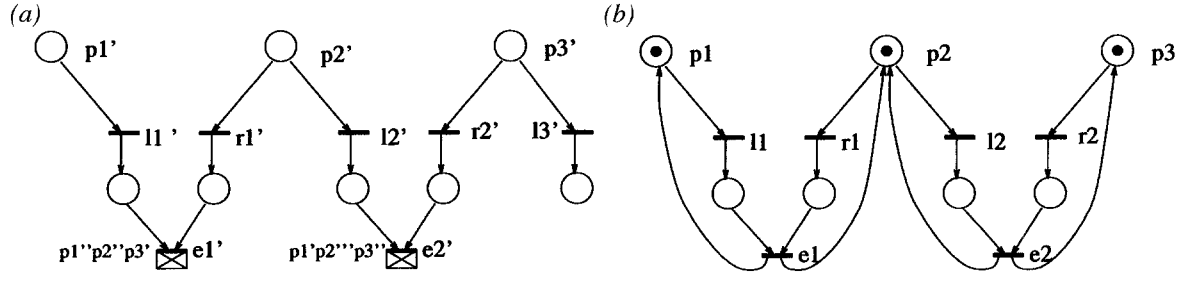

Fig. 3. A deadlock-free component for three dining philosophers.

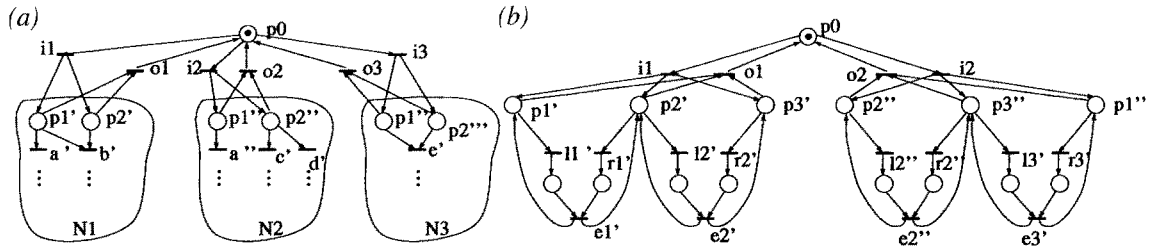

Fig. 4. Composition of deadlock-free components: (*a*) general, (*b*) for the dining philosophers.

6. Deadlock Prevention by Transformation

In this approach a deadlock prevention will be performed by an original PN, and not by an unfolding. The PN has to be modified in such a way that firing of transitions leading to terminal markings are forbidden. It is possible to recognise the transitions leading to the deadlock using unfolding. A resulting deadlock-free PN can be obtained by adding extra "switch" places and transitions. Roughly speaking, this is a composition with a special "control" PN that reduces the set of traces of initial PN (reducing concurrency, for example).

6.1 A Structure of Spoiling

Let us call all transitions that are in conflict with some spoiler its *victims*.

Definition 6.1 (Maximal Victim). Let $t1' \in N'$ be a maximal spoiler for victim $t2'$. Transition $t3' \in N'$ is called a *maximal victim* for $(t1',t2')$ if: (i) $t3'$ is in conflict with $t1'$ $(t3'\#t1')$; (ii) $t3'$ precedes $t2'$ or coincides with $t2'$ $(t3' \Rightarrow t2')$; (iii) there is no transition $t4' \in N'$ such that: $t4'\#t1'$ and $t4' \Rightarrow t3'$.

Intuitively, a maximal victim is the first victim of the maximal spoiler. A maximal spoiler and its maximal victim share input places, called sources of spoiling.

Definition 6.2 (Source of Spoiling). A place $p' \in N'$ is called a *source of spoiling* for a maximal spoiler $t1' \in N'$ and its maximal victim $t2' \in N'$ if: (i) p' precedes $t1'$ and $t2'$ $(p' \Rightarrow t1', p' \Rightarrow t2')$; (ii) there is no place $p1' \in N'$ such that: $p' \Rightarrow p1'$ and $p1' \Rightarrow t1'$ and $p1' \Rightarrow t4'$.

If a cyclic PN has a deadlock, then its unfolding contains a configuration corresponding to the terminal marking which includes a maximal spoiler for each cut-off. As follows from Definition 6.1, and Definition 6.2, each cut-off is preceded by the corresponding maximal victim and a source of spoiling.

Let us denote by T_{PMS} the set of all prototypes of the maximal spoilers for the PN N; T_{PMV}, the set of all prototypes of all maximal victims; and P_{PSS}, the set of all prototype places corresponding to sources of spoiling. The prototype of a maximal spoiler (victim) will be called a PM-spoiler (a PM-victim). It is clear that the same transition $l \in N$ can be a prototype of different instantiations, in particular it can be a prototype of the maximal spoiler $l' \in N'$ and also a prototype of a maximal victim $l'' \in N'$.

6.2 A Safe Example

A deadlock prevention by the transformation (DPT) method for safe PNs will be presented in Section 6.4. Here, an informal intuitive introduction to the method based on the dining philosophers' example [5] will be given. Figure 2(*a*) shows a PN for three dining philosophers. Its unfolding is shown in Fig. 2(*b*). The set of its cut-offs is $T'_c = \{e1',e2',e3'\}$. There are two terminal configurations $C'_{t1} = \{r1',r2',r3'\}$ and $C'_{t2} = \{l1',l2',l3'\}$. The sets of its maximal spoilers are $T'_{s1} = \{r1',r2',r3'\}$ and $T'_{s2} = \{l1',l2',l3'\}$. Consider, for example, cut-off $e1'$. There are two maximal spoilers $l2'$ and $r3'$, which belong to two different terminal configurations. Transition $r1'$ is a maximal victim for spoiler $l2'$, and $p1'$ is the corresponding source of spoiling. Hence, transitions $l2$ and $r3$ are *prototypes of maximal spoilers* (PM-spoilers) and transitions $l1$ and $r1$ are *prototypes of maximal victims* (PM-victims) for the cut-off $e1'$. Similarly, transition $l3$ and $r1$ are PM-spoilers and transitions $l2$ and $r2$ are PM-victims for cut-off $e2'$. Transitions $l1$ and $r2$ are PM-spoilers and transitions $l3$ and $r3$ are PM-victims for cut-off $e3'$. Hence, each of li or ri serves as a PM-spoiler and as a PM-victim for different cut-offs.

The result of our deadlock prevention strategy (before optimisation) is shown in Fig. 5(*a*). In comparison with the original net, this net contains additional places and transitions. First, we have identified local configurations of cut-offs and tried to check whether some of them can be joined in one larger configuration. Such joint configurations are called *maximal union cut-off configurations* (MUC-configurations). In this example this is not possible, since all cut-offs are in pairwise conflict. Hence, each local configuration of cut-offs is to be made alternative to every other configuration of cut-offs in the final deadlock-free PN. Then, each maximal spoiler and the maximal victim corresponding to it is supplied with an additional "block" input place (places $b1$–$b6$) and additional "unblock" output places. Block places serve to control the choice between the spoiler and its victim in the following way. If at the current step a local configuration of the cut-off (say, $e1$) is chosen to occur, then its spoilers ($l2$ and $r3$) are not able to fire since their corresponding "block" places are out of tokens and the victims preceding $e1$ (transitions $l1$ and $r1$) can occur, since their "block" places will be provided with tokens.

Block places for each MUC-configuration are supplied with tokens by new unblocked transitions, d_i. These transitions have one free-choice place p_{in} as an input place. Place p_{in} serves as a shared resource place for all alternative MUC-configurations.

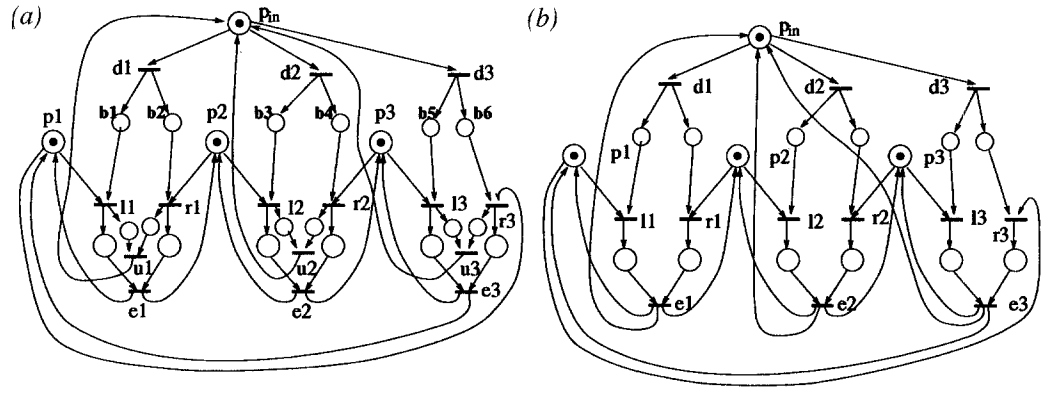

Fig. 5. (*a*) Deadlock prevention for three dining philosophers, (*b*) optimised.

Unblock places of all prototypes of maximal victims are synchronised by post-victim transitions $u1$, $u2$, $u3$ that have place p_{in} as the only output place. Therefore, each time all PM-victims occur a choice can be made which MUC-configuration can work at the next cycle of operation.

This general solution can be often optimised, using ordering relations between places and transitions. Since in our example, each cut-off serves as a synchroniser to all PM-victims of the corresponding MUC-configuration, post-victim transitions and their input unblock places are redundant and can be removed. Place p_{in} can be made input for the prototypes of cut-off transitions. Figure 5(*b*) shows an optimised result of a deadlock prevention for three dining philosophers.

6.3 Reduced Live Equivalence

Unfolding, and the structure of spoiling, give an idea of how to prevent deadlocks by transforming the original PN. Additional complexity of a modified PN seems to be relatively low. However, some live traces can be lost and therefore one needs to check which behaviours are preserved during the transformation.

Let us examine which behaviours are lost in the example of the three dining philosophers. In the original specification, transition $l3$ can fire concurrently with transition $e1$ – the third philosopher can take the left hand fork while the first philosopher is eating. For the final deadlock-free net from Fig. 5(*b*) such a "prudent taking of a fork" is not possible. However, the third philosopher can eat after the first one: any "ordering of eating" allowed for in the original specification is also allowed by the final specification. Any ordering of cut-offs feasible in the original specification, is also allowed in the final specification.

Definition 6.3 (Cut-offs Configuration)

A local configuration of cut-off c' is denoted by $\{\Rightarrow]c'\}$).

A configuration that can be obtained as a union of some local configurations of cut-offs is called a *UC-configuration*.

A UC-configuration C'_U of unfolding N' is said to be maximal or *MUC-configuration* if it is not a subset of any other UC-configuration of N'.

It is clear that not every configuration is a UC-configuration. For example, in Fig. 2(*b*) the set $\{l1',r1',e1',l3'\}$ is a maximal configuration, but not a UC-configuration, and $\{l',r1',el'\}$ is a MUC-configuration.

Definition 6.4 (Traces of Configuration)

Trace $t_{C'}$ is called a *trace of configuration* C' if it consists only of transitions from the configuration C'.

Trace $t_{C'}$ of configuration $C'$is said to be maximal or MC-trace if it is not a subset of any other trace of configuration C'.

Trace s feasible in N is called a concatenated trace if it can be represented as a concatenation of prototypes for some MC-traces.

A concatenated trace is called a MUC-trace if it can be represented as a concatenation of prototypes for some maximal traces corresponding to MUC-configurations of the unfolding.

The next definition clarifies which behaviours are preserved in the above deadlock prevention procedure.

Definition 6.5 (Reduced Live Equivalence). PN $N2$ with a set of transitions $T2$ is a *reduced live equivalent* to PN $N1$ with a set of transitions $T1 \subseteq T2$ if:

1. For any trace $s = t_1,\ldots,t_k,\ldots$ feasible in $N2$ its projection to $T1$ is feasible in $N1$. (Trace q under $T1 \subseteq T2$ is said to be a projection of trace s under $T2$ if it is obtained from s by removing all transitions that do not belong to $T1$.)
2. For any MUC-trace p feasible in $N1$ there exists a trace s feasible in $N2$ such that the projection of s to $T1$ is equal to p.

Informally, condition 1 of Definition 6.5 states that no new behaviours can appear in the new net $N2$ and condition 2 guarantees that $N2$ preserves all MUC-traces of the original net $N1$, i.e. all fundamental cyclic behaviours of $N1$ are retained.

6.4 DPT Algorithm

This algorithm summarises the considerations that were discussed in the above example for a safe PN. An additional constraint, temporarily assumed for the class of considered PNs, is as follows: each PM-spoiler and PM-victim can have not more than one occurrence in every MUC-configuration.

Algorithm 4. Deadlock Prevention by Transforming Safe Nets

1. Let PN N specify the original process.
2. Generate unfolding N', with set B of MUC-configurations $B = \{C'_i\}$.
3. Find in N' all terminal configurations $\{C'_t\}$. **If** no C'_t exists **return** N.
4. **If** N is unsafe or there is a PM-spoiler or PM-victim with multiple occurrence in a MUC-configurations from B, **then return**. Deadlock prevention is interrupted.
5. For each MUC-configuration from B and for each terminal configuration from $\{C'_t\}$ **do**
6. {Find maximal spoilers and maximal victims and the corresponding prototypes: PM-spoilers and PM-victims.
7. For each $t \in$ PM-spoilers $\cup$ PM-victims if no input block place exists, then add it.
8. For each $t \in$ PM-victims if no output post-victim place exists, then add it }**end do**.
9. For each MUC from B add an unblock transition d_i and post-victim transition, u_i. Connect d_i with all block places of the MUC and all post-victim places with u_i.
10. Add to PN N the initially marked place p_{in} and connect it with all u_i transitions (inputs to p_{in}) and all d_i transitions (outputs from p_{in}). Optimise the resulting net.

Proposition 6.1. A PN $N2$ obtained as a result of Algorithm 4 is reduced live equivalent to the original PN $N1$.

6.5 General Case

In [9], it was shown how the deadlock prevention procedure based on PC transformations can be generalised to bounded (unsafe) PNs.

In general, one instantiation of transition $t1$ could serve as a spoiler ($t1''$) and the other as a victim ($t1'$) inside the same MUC-configuration. Therefore, the control of a token flow in the block places for victims and spoilers must be dynamic. A special mechanism [9] to identify whether an occurrence of a transition is a spoiler or a victim is needed.

For unsafe PNs multiple concurrent instantiations of places and transitions can occur inside one configuration of an unfolding. This makes a deadlock prevention harder to implement.

The procedure for deadlock prevention by PN transformation for general PN also can be found in [9].

7. Experimental Results

The major algorithms of the methods of deadlock prevention presented in this paper have been implemented inside the SIS tool [11] (the results below are obtained by the experimental version of the software tool, which is still under development). The efficiency of the approach presented was evaluated in comparison to the efficient technique of a PN traversal based on BDDs [12].

The results of a deadlock detection for three examples are examined in [9]: a PN for the dining philosophers [3], an n-stage Muller pipeline [13] with $n/3$ portions of information moving concurrently and an n user distributed mutual exclusion (DME) arbiter [12]. All examples are scalable, in such a way that the number of states of the system can be exponentially increased by iteratively repeating a basic pattern. It is worth noting that even though the task of deadlock detection by unfolding is NP-complete for the case of 50 dining philosophers, all deadlocks were found in less than 10 s of CPU time on a Sparc 10 workstation (the time for a PN traversal based on BDD for this example is more than 4 hours).

The size of the unfolding is typically much smaller than the size of the reachability graph of the PN. For some properties, like the boundedness, the unfolding technique is very efficient while the BDD-based one is extremely inefficient. Our examples presented in [9] show that it is also right for a deadlock detection.

Table 1 shows the results of an unfolding tool application to the set of examples from "Manufacturing Systems world". The most interesting examples of corresponding PN were taken from [14–17], and it was confirmed that all of them show very small runtime.

In Table 1, "size" is of an original PN. P and T are the number of places and transitions. "Cut-off" is a number of cut-offs. "Time" is a CPU time for detection of cut-offs for sparcstation Sun 10. "Cyclic" presents, a cyclic net made from an acyclic net above it.

8. Conclusion

The approach for a deadlock prevention using PN unfolding has been presented. Several useful properties were introduced to guide the procedure of prevention.

An efficient strategy for a deadlock prevention for the wider class of processes needs to be developed. This should cover:

1. Unsafe PNs (without reduction to a safe form by place-pipelining [10]). It was observed that our methods are not efficient when applied to unsafe nets. This is because for unsafe PNs, unfoldings become large and the detection procedure could take a long time [9].

   The folding procedure from our first method becomes non-deterministic owing to the existence of several images for one cut-off. Further development of folding methods is required for unsafe PNs.
2. The problem of PN optimisation (re-synthesis using *petrify*) is computationally expensive. We hope to improve our methods of direct PN transformations to avoid a PN optimisation step.

Acknowledgments

We are grateful to Dr Luciano Lavagno for many useful discussions and assistance with the SIS tool and to Dr Jordi Cortadella for implementing the *petrify* tool. We are also grateful to Dr Sergei Ten for his work on the software implementation of the unfolding methods and to Mr Satoshi Iwata for his efforts in testing our software and obtaining experimental results.

Table 1. Deadlock prevention for manufacturing systems.

Example	Size		Unfolding				Example	Size		Unfolding			
	P	T	P	T	Cut-off	Time(s)		P	T	P	T	Cut-off	Time(s)
Petri net synthesis for discrete event control...							On deadlock in concurrent systems						
Initial	10	6	39	35	14	1	Trap	5	4	8	6	1	0
R1	9	8	9	8	2	0	CS	31	19	110	74	16	0
2-GPME	13	10	126	140	65	1	Chained	18	15	21	15	0	0
Exclusion	6	4	15	11	5	0	(Cyclic)	18	18	87	93	18	0
PME-a	6	4	17	12	5	0	Concurrent	14	8	25	14	1	0
PME-b	6	4	12	10	4	0	Empty	10	10	24	28	11	0
Z_b	7	5	9	6	2	0	L_2	16	10	47	35	10	0
GSME	13	10	56	54	23	1	(Cyclic)	18	18	87	93	18	0
C^1	12	10	33	29	10	0	AND-OR	41	25	157	103	26	1
Controller	17	9	21	9	1	0	(Cyclic)	41	26	117	90	9	0
Symposium on discrete events and manufacturing...							Petri nets in flexible and agile...						
Inter	9	7	14	10	2	0	ASPN	10	9	10	9	3	0
Stochastic	14	10	17	11	3	0	TPN	28	16	454	169	46	1
FMS	41	40	51	40	6	0	(Cyclic)	28	17	5084	4365	2157	874

References

1. Z. Banaszak and B. Krogh, "Deadlock avoidance in flexible manufacture systems with concurrently competing process flows", IEEE Transactions on Robotics and Automation, 6(6), pp. 724–734, December 1990.
2. J. Ezpleta, J. Colom and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacture systems", IEEE Transactions on Robotics and Automation, 11(2), pp. 173–184, April 1995.
3. K. L. McMillan, "A technique of state space search based on unfolding", Formal Methods in System Design, 6(1), pp. 45–65, 1995.
4. J. Esparza, S. Römer and W. Vogler, "An improvement of McMillan's unfolding algorithm", in Tools and Algorithms for the Construction and Analysis of Systems, vol. 1055 of Lecture Notes in Computer Science, pp. 87–106, Passau, Germany, Springer-Verlag, March 1996.
5. J. L. Peterson, Petri Nets, vol. 9, ACM Computing Surveys, 3, September 1977.
6. A. Kondratyev, M. Kishinevsky, A. Taubin and S. Ten, "A structural approach for the analysis of Petri nets by reduced unfoldings", in Applications and Theory of Petri Nets 1996, 17th International Conference, Proceedings, vol. 1091 of Lecture Notes in Computer Science, pp. 346–365, Osaka, Japan, June, 1996.
7. L. Pomello, G. Rozenberg and C. Simone, "A survey of equivalence notions for net based systems", Lecture Notes in Computer Science, 609, pp. 410–472, 1993
8. J. Cortadella, M. Kishinevsky, L. Lavagno and A. Yakovlev, "Syntheizing Petri nets from state-based models", in Proceedings of the International Conference on Computer-Aided Design, pp. 164–171, November 1995.
9. A. Taubin, A. Kondratyev and M. Kishinevsky, "Deadlock prevention using Petri nets and their unfoldings", Technical Report 97-2-004, The University of Aizu, 1997.
10. V. I. Varshavsky, M. A. Kishinevsky, V. B. Marakhovsky, V. A. Peschansky, L. Y. Rosenblum, A. R. Taubin and B. S. Tzirlin, Self-timed Control of Concurrent Processes, Kluwer, 1990 (Russian edn 1986).
11. E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis", Technical Report UCB/ERL M92/41, University of California, Berkeley, May 1992.
12. E. Pastor, O. Roig, J. Cortadella and R. Badia, "Petri net analysis using Boolean manipulation", in 15th International Conference on Application and Theory of Petri Nets, pp. 416–435, Zaragoza, Spain, June 1994.
13. D. E. Muller, "Asynchronous logics and application to information processing", in Proceedings of the Symposium on Application of Switching Theory in Space Technology, pp. 289–297, Stanford University Press, 1963.
14. M. Zhou and F. DiCesare, Petri Net Synthesis for Discrete Event Control of Manufacturing Systems, Kluwer, 1993.
15. Proceedings of CESA'96 IMACS Multiconference, Computational Engineering in System Applications. Symposium on Discrete Events and Manufacturing Systems, IEEE-SMC, 1996.
16. N. Cacutalua, On Deadlocks in Concurrent Systems: A Petri Net based Approach for Deadlock Prediction and Avoidance, GMD, 1993.
17. M. Zhou (ed.), Petri Net in Flexible and Agile Automation, Kluwer, 1995.