Unfoldings of Unbounded Petri Nets

Parosh Aziz Abdulla[1], S. Purushothaman Iyer[2], and Aletta Nylén[1]

[1] Dept. of Computer Systems
P.O. Box 325
Uppsala University
S-751 05 Uppsala, Sweden
{parosh,aletta}@docs.uu.se
[2] Dept of Computer Science
NC State University
Raleigh, NC 27695-7534
purush@csc.ncsu.edu

Abstract. Net unfoldings have attracted much attention as a powerful technique for combating state space explosion in model checking. The method has been applied to verification of 1-safe (finite) Petri nets, and more recently also to other classes of finite-state systems such as synchronous products of finite transition systems. We show how unfoldings can be extended to the context of infinite-state systems. More precisely, we apply unfoldings to get an efficient symbolic algorithm for checking safety properties of unbounded Petri nets. We demonstrate the advantages of our method by a number of experimental results.

1 Introduction

Model Checking has had a great impact as an efficient method for algorithmic verification of finite-state systems. A limiting factor in its application is the *state space explosion* problem, which occurs since the number of states grows exponentially with the number of components inside the system. Therefore, much effort has been spent on developing techniques for reducing the effect of state space explosion in practical applications. One such a technique is that of *partial orders* which is based on the observation that not all interleavings of a given set of independent actions need to be explored during model checking. Several criteria for independency has been given, e.g., *stubborn sets* [Val90], *persistent sets* [GW93] or *ample sets* [Pel93]. A method which has drawn considerable attention recently is that of *unfoldings* [McM95,ERV96,ER99]. Unfoldings are *occurrence nets*: unrollings of Petri nets that preserve their semantics. Although unfoldings are usually infinite, it is observed in [McM95] that we can always construct a finite initial prefix of the unfolding which captures its entire behaviour, and which in many cases is much smaller than the state space of the system. Unfoldings have been applied to n-safe (i.e., finite-state) Petri nets, and more recently to other classes of finite-state systems such as synchronous products of finite transition systems [LB99,ER99]

In a parallel development, there has been numerous efforts, to extend the applicability of model checking to the domain of infinite-state systems. This has resulted in several highly nontrivial algorithms for verification of timed automata, lossy channel systems, (unbounded) Petri nets, broadcast protocols, relational automata, parametrized systems, etc. These methods operate on symbolic representations, called *constraints* each of which may represent an infinite set of states. However, in a manner similar to finite-state verification, many of these algorithms suffer from a *constraint explosion* problem limiting their efficiency in practical applications. As the interest in the area of infinite-state systems increases, it will be important to design tools which limit the impact of constraint explosion. With this in mind, we have considered [AKP97,AJKP98] a refinement of the ample set construction and applied it to infinite-state systems such as Petri nets and lossy channel systems.

In this paper, we show how the unfolding technique can be made to work in the context of infinite state systems. More precisely, we present an unfolding algorithm for symbolic verification of unbounded Petri nets. We adapt an algorithm described in [AČJYK96] for backward reachability analysis which can be used to verify general classes of safety properties. Instead of working on individual markings (configurations) of the net (as is the case with the previous approaches [McM95,ERV96,ER99,LB99]) we let our unfolding algorithm operate on constraints each of which may represent an (infinite) upward closed set of markings. We start from a constraint describing a set of "final" markings, typically undesirable configurations which we do not want to occur during the execution of the net. From the set of final markings we unroll the net backwards, generating a *Reverse Occurrence Net (RON)*. In order to achieve termination we present an algorithm to compute a postfix of the RON, which gives a complete characterization of the set of markings from which we can reach a final marking. Using concepts from the theory of well quasi-orderings we show that the postfix is always finite. In fact, our method offers the same advantages over the algorithm in [AČJYK96], as those offered by the algorithms of [McM95,ERV96] in the context of finite-state systems.

Based on the algorithm, we have implemented a prototype, whose results on a number of simple examples are encouraging.

Outline In the next section we give some preliminaries on Petri nets. In Section 3 we introduce Reverse Occurrence Nets (RONs). In Section 4 we describe the unfolding algorithm. In Section 5 we describe how to compute a finite postfix of the unfolding. In Section 6 we report some experimental results. Finally, in Section 7 we give some conclusions and directions for future research.

2 Preliminaries

Let $\mathbb{N}$ be the set of natural of numbers. For $a, b \in \mathbb{N}$, we define $a \ominus b$ to be equal to $a - b$ if $a \geq b$, and equal to 0 otherwise. A *bag* over a set A is a mapping from A to $\mathbb{N}$. Relations and operations on bags such as $\leq$, $+$, $-$, $\ominus$, etc, are defined as usual. Sometimes, we write bags as tuples, so (a, b, a) represents a bag B with

$B(a) = 2$, $B(b) = 1$, and $B(d) = 0$ if $d \neq a, b$. A set A of bags is said to be *upward closed* if $B_1 \in A$ and $B_1 \leq B_2$ imply $B_2 \in A$. The *upward closure* of a bag A is the set $\{A' \mid A \leq A'\}$. We may interpret a set S as a bag with $S(a) = 1$ if $a \in S$ and $S(a) = 0$ if $a \notin S$. We use $|S|$ to denote the size of the set S. In this paper, we shall use the terminology of [ERV96] as much as possible.

A *net* is a triple (S, T, F) where S is a finite set of *places*, T is a finite set of *transitions*, and $F \subseteq (S \times T) \cup (T \times S)$ is the *flow relation*. By a *node* we mean a place or a transition. The preset ${}^\bullet x$ of a node x is the set $\{y \mid (y, x) \in F\}$. The postset $x^\bullet$ is similarly defined. A *marking* M is a bag over S. We say that a transition t is *enabled* in a marking M if ${}^\bullet t \leq M$. We define a transition relation on the set of markings, where $M_1 \longrightarrow M_2$ if there is $t \in T$ which is enabled in M_1 and $M_2 = M_1 - {}^\bullet t + t^\bullet$. We let $\xrightarrow{*}$ denote the reflexive transitive closure of $\longrightarrow$. We say that a marking M_2 is *coverable* from a marking M_1 if $M_1 \xrightarrow{*} M_2'$, for some $M_2' \geq M_2$. A *net system* is a tuple $N = (S, T, F, M_{init}, M_{fin})$, where (S, T, F) is a net and M_{init}, M_{fin} are markings, called the *initial* and the *final* marking of N respectively. In this paper, we consider the *coverability problem* defined as follows.

Instance A net system $(S, T, F, M_{init}, M_{fin})$.

Question Is M_{fin} coverable from M_{init}?

Using standard methods [VW86,GW93], we can reduce the problem of checking safety properties for Petri nets to the coverability problem.

To solve the coverability problem, we perform a backward reachability analysis. We define a *backward* transition relation [AČJYK96], such that, for markings M_1 and M_2 and a transition t, we have $M_2 \leadsto_t M_1$ if $M_1 = (M_2 \ominus t^\bullet) + {}^\bullet t$. We let $\leadsto = \cup_{t \in T} \leadsto_t$, and let $M \overset{k}{\leadsto} M'$ denote that $M = M_0 \leadsto M_1 \leadsto \cdots M_k = M'$, for markings $M_0, \ldots, M_k$. We define $\overset{*}{\leadsto}$ to be the reflexive transitive closure of $\leadsto$. Observe that, for each marking M_2 and transition t, there is a marking M_1 with $M_2 \leadsto_t M_1$, i.e., transitions are always enabled with respect to $\leadsto$. The following lemma relates the forward and backward transition relations.

Lemma 1.

1. *If $M_1 \longrightarrow M_2$ and $M_2' \leq M_2$ then there is $M_1' \leq M_1$ such that $M_2' \leadsto M_1'$.*
2. *If $M_2 \leadsto M_1$ and $M_1' \geq M_1$ then there is M_2' such that $M_2' \geq M_2$ and $M_1' \longrightarrow M_2'$.*

3 Reverse Occurrence Nets

In this section we introduce *Reverse Occurrence Nets (RONs)*. A RON corresponds to "unrolling" a net *backwards*. Formally, a *RON* R is a net $(\mathsf{C}, \mathsf{E}, \mathsf{F})$ satisfying the following three conditions

(i) $|c^\bullet| \leq 1$ for each $c \in \mathsf{C}$.

(ii) there is no infinite sequence of the form $c_1 \mathsf{F} e_1 \mathsf{F} c_2 \mathsf{F} \cdots$. This condition implies that there are no cycles in the RON, and that there is a set $\max(\mathsf{F})$ of nodes which are maximal with respect to F.

(iii) $\max(\mathsf{F}) \subseteq \mathsf{C}$.

In a RON, the places and transitions are usually called *conditions* and *events* respectively. A set of events $E \subseteq \mathsf{E}$ is considered to be a *configuration* if $e \in E$ and $e\mathsf{F}^*e'$ imply $e' \in E$.

Remark 1. In [McM95,ERV96], a configuration E is upward closed in the sense that if an event e belongs to E, then all events above e (with respect to F) also belong to E. In our case, configurations are downward closed. Furthermore, in [McM95,ERV96], configurations are required to be *conflict free*, i.e., for all events $e_1, e_2 \in E$ we have ${}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset$. Notice that this property is always satisfied by our configurations, since we demand that $|c^\bullet| \leq 1$ for each condition.

Consider a net system $N = (S, T, F, M_{init}, M_{fin})$ and a RON $(\mathsf{C}, \mathsf{E}, \mathsf{F})$, and let $\mu : \mathsf{C} \cup \mathsf{E} \to S \cup T$ such that $\mu(c) \in S$ if $c \in \mathsf{C}$ and $\mu(e) \in T$ if $e \in \mathsf{E}$. For $C \subseteq \mathsf{C}$, we define $\#C$ to be a marking such that, for each place s, the value of $\#C(s)$ is equal to the size of the set $\{c \in C \mid \mu(c) = s\}$. In other words $\#C(s)$ is the number of conditions in C labeled with s. We say that $(\mathsf{C}, \mathsf{E}, \mathsf{F}, \mu)$ is a *(backward) unfolding* of N if the following two conditions are satisfied: (i) $\#\max(\mathsf{F}) = M_{fin}$, i.e., the set of conditions which are maximal with respect to F correspond to the final marking; and (ii) μ preserves F, viz., if $(x, y) \in \mathsf{F}$ then $(\mu(x), \mu(y)) \in F$.

For a configuration E, we define $Cut(E)$ to be the set

$$(\{{}^\bullet e \mid e \in E\} \ \cup \ \max(\mathsf{F})) \ - \ \{e^\bullet \mid e \in E\}$$

We define the marking $mark(E) = \#(Cut(E))$.

In Figure 1, we show a net system N with seven places, $s_1, \ldots, s_7$, and four transitions, $t_1, \ldots, t_4$. We also show an unfolding[1] U of N, assuming a final marking (s_1, s_7). Examples of configurations in U are $E_1 = \{e_2, e_4\}$ with $mark(E_1) = (s_1, s_2, s_3)$, and $E_2 = \{e_1, e_2, e_3, e_4\}$ with $mark(E_2) = (s_1, s_2, s_2, s_3)$.

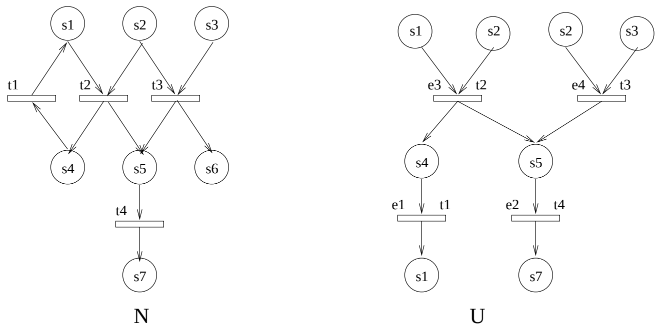


Fig. 1. A net system and one of its unfoldings

[1] To increase readability, we show both names and labels of events in the figure, while we omit name of conditions.

4 An Unfolding Algorithm

We present an algorithm (Figure 2) which, for a given net system $N = (S, T, F, M_{init}, M_{fin})$, generates an unfolding of N in an incremental manner. In a manner similar to [ERV96], an unfolding $U = (\mathsf{C}, \mathsf{E}, \mathsf{F}, \mu)$ is represented as a list of objects corresponding to conditions and events in the underlying RON. An event e is represented as an object (C, t) where t is the label $\mu(e)$ of e and C is its set $e^\bullet$ of post-conditions. A condition c is represented by an object (e, s) where s is the label $\mu(c)$ of c, and e is its (single) post-event $c^\bullet$. We observe the flow relation F and the labeling function μ are included in the encoding.

Consider a set of conditions C of U to be t-enabled provided there exists a configuration E such that $C \subseteq \mathit{Cut}(E)$ and $0 < \#C \leq t^\bullet$, i.e., there is a configuration E such that $C \subseteq \mathit{Cut}(E)$ and all the conditions in C are in the postset of t. Furthermore, consider C to be maximally t-enabled provided there is no other set C' such that $C \subset C' \subseteq \mathit{Cut}(E)$ and C' is t-enabled. We will write $\mathsf{ME}_t(C)$ to denote that C is maximally t-enabled. We define $\mathit{Xtnd}(U)$ to be the set of events by which U can be extended and is formally defined as follows:

$$\mathit{Xtnd}(U) = \{(C, t) \mid \mathsf{ME}_t(C) \text{ and } (C, t) \notin U\}$$

Observe that the definition implies that there are no redundancies in the unfolding. In other words we will not have two different events both having the same label and the same postcondition.

The unfolding algorithm is shown in Figure 2. It maintains two variables, namely the current unfolding U (initialized to the final marking M_{fin}), and a set X of events by which the unfolding can be extended. The algorithm proceeds by considering the events in X in turn (this procedure is fair in the sense that each event added to X will eventually be considered). At each iteration an event in X is picked and moved to U. Furthermore, the possible extensions of the new unfolding are computed, using the function Xtnd, and added to X. Notice that the algorithm does not necessarily terminate.

The unfolding algorithm gives a symbolic representation of upward closed sets from which M_{fin} is coverable. More precisely (Theorem 1), the upward closure of the markings appearing in U, gives exactly the set of markings from which M_{fin} is coverable. Notice that each event in the unfolding corresponds to a step in the backward unrolling of the net. The efficiency we gain through applying unfoldings on upward closed sets, as compared to the standard symbolic algorithm based on the backward transition relation $\rightsquigarrow$, can be explained in a manner similar to the finite state case [McM95,ERV96]; namely the addition of a set of concurrent events to the unfolding corresponds to an exponential number of applications of the $\rightsquigarrow$ relation.

In the sequel we let U^i denote the value of the variable U after i iterations of the loop. The following lemmas (the proof of which can be found in the appendix) relate unfoldings with the backward transition relation $\rightsquigarrow$.

Lemma 2. *If $M_{fin} \overset{k}{\rightsquigarrow} M$ then there is an ℓ and a configuration E in U^ℓ such that $\mathit{mark}(E) \leq M$*

Input: net system $N = (S, T, F, M_{init}, M_{fin})$, where $M_{fin} = (s_1, \ldots, s_n)$.
var U: unfolding of N; X: set of events.
begin
    $U := (s_1, \emptyset), \ldots, (s_m, \emptyset)$
    $X := Xtnd(U)$
    while $(X \neq \emptyset)$ **do**
        Pick and delete $e = (C, t)$ from X
        Add (C, t) to U and also add $\forall s \in {}^\bullet t$ a new condition (s, e) to U
        $X := X \cup Xtnd(U)$
end

Fig. 2. Unfolding Algorithm

We now present the lemma in the other direction which shows that the marking associated with every configuration in an unfolding is backwards reachable.

Lemma 3. *For each ℓ and configuration E in U^ℓ, there is a marking M such that $M \leq mark(E)$ and $M_{fin} \overset{*}{\rightsquigarrow} M$.*

From Lemma 1, Lemma 2, and Lemma 3 we get the following theorem.

Theorem 1. *M_{fin} is coverable from a marking M if and only if there is an ℓ and a configuration E in U^ℓ such that $mark(E) \leq M$.*

Notice that as a special case we can take M in Theorem 1 to be equal to M_{init}.

5 Termination

In this section we show how to compute finite postfixes of unfoldings. We define special types of events which we call *cut-off points*. In Theorem 2 we show that cut-off points do not add any markings to the upward closed sets characterized by the unfolding. This means that, in the unfolding algorithm (Figure 2) we can safely discard all cut-off points, without ever adding them to the unfolding U. Furthermore, we use concepts from the theory of well quasi-orderings (Theorem 2) to show that, if all cut-off points are discarded, then the variable X in the unfolding algorithm eventually becomes empty implying termination of the algorithm. We start with some definitions and auxiliary lemmas.

We assume a net system N and an unfolding U of N. For an event, we use $e\downarrow$ to denote the configuration $\{e' \mid e\mathsf{F}^* e'\}$. For configurations E_1 and E_2, we use $E_1 \prec E_2$ to denote that $|E_1| < |E_2|$ and $mark(E_1) \leq mark(E_2)$. For an event e, we say that e is a *cutoff point* in U if there is a configuration E in U such that $E \prec e\downarrow$.

We recall from the previous section that U^i denotes the value of the variable U in the unfolding algorithm, after i iterations of the loop. In order to prove the cutoff theorem we need the following lemma (the proof of which can be found in the appendix).

Lemma 4. *Consider configurations E_1, E_2, and E_2' in U^k where $E_1 \prec E_2$ and $E_2 \subseteq E_2'$. There is an ℓ and a configuration E_1' in U^ℓ such that $E_1' \prec E_2'$.*

Now we are ready to show in the following theorem that cutoff points can be discarded safely.

Theorem 2. *For each k and configuration E_2 in U^k, there is an ℓ and configuration E_1 in U^ℓ where $mark(E_1) \leq mark(E_2)$ and E_1 does not contain any cutoff points.*

Proof. We use induction on $|E_2|$. The base case is trivial. If E_2 does not contain any cutoff points, then the proof is trivial. Otherwise let e_2 be a cutoff point in E_2. Clearly, $e_2\!\downarrow \subseteq E_2$. Since e_2 is a cut-off point, we know that there is a configuration E in U^k such that $E \prec e_2\!\downarrow$. By Lemma 4 there is an ℓ and a configuration E_1 in U^ℓ such that $E_1 \prec E_2$, i.e., $|E_1| < |E_2|$ and $mark(E_1) \leq mark(E_2)$. The claim follows by induction hypothesis.

To prove termination of the unfolding algorithm, we use the fact that markings are well quasi-ordered (consequence of Dickson's lemma [Dic13]), i.e., for any infinite sequence $M_0, M_1, \ldots$ of markings, there are i and j with $i < j$ and $M_i \leq M_j$.

Theorem 3. *The unfolding algorithm terminates if all cut-off points are discarded.*

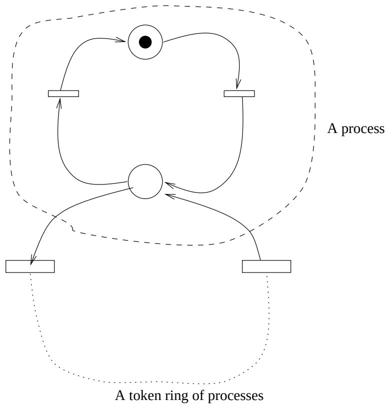
Proof. Suppose that the algorithm does not terminate. Since all nodes are finitely branching we have an infinite sequence $e_0, e_1, e_2, \ldots$, of events where $e_{i+1} \mathsf{F} c_i \mathsf{F} e_i$, for some condition c_i. Notice that $|e_j\!\downarrow| > |e_i\!\downarrow|$, whenever $j > i$. By Dickson's lemma, it follows that there are i and j with $i < j$ and $mark(e_i\!\downarrow) \leq mark(e_j\!\downarrow)$. This implies that e_j is a cut-off point, which is a contradiction.

Remark Theorem 1, Theorem 2, and Theorem 3 give a complete terminating procedure for checking coverability in unbounded Petri nets: use the unfolding algorithm discarding all cutoff points. The final marking M_{fin} is coverable from the initial marking M_{init} iff a configuration E appears in the unfolding with $mark(E) \leq M_{init}$.

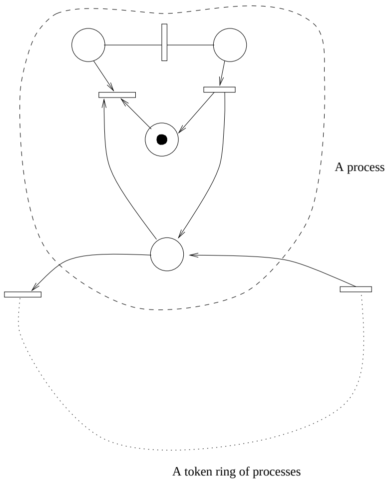
6 Experimental Results

In this section we report on some of the issues that we had to solve in implementing the unfolding algorithm. While our implementation borrows ideas from [McM95,ERV96], there are several issues that are peculiar to our backward reachability. To wit, they are:

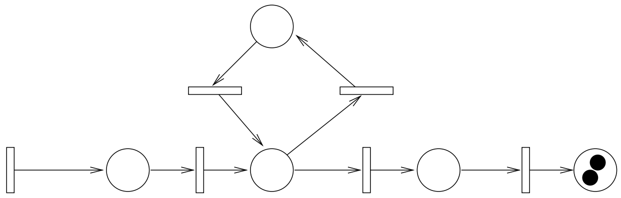
- **Implementation of $Xtnd$:** The abstract algorithm (presented in Section 4) implies that $Xtnd$ is computed in every iteration. However, in the implementation a queue of possible sets of conditions that could be the postset of a (potential) event are maintained. As new conditions are generated we check



A. Token Ring



B. Token Ring, version 2



C. Buffer

Fig. 3. Examples of nets considered

whether these new conditions can be added to already existing (partial) sets of post-conditions to form larger sets of postconditions. By doing so we reduce a seemingly combinatorial problem to a depth-first search on the unfolding.

- **Checking termination:** As a new event e is generated we calcluate $|e{\downarrow}\,|$ and $mark(e{\downarrow})$. We compare this information against $mark(e'{\downarrow})$ and $|mark(e'{\downarrow})|$ for all events e' currently in the unfolding. While our definition of a cut-off event calls for comparing against all configurations in the unfolding, our implemention is sound, though not effecient (as otherwise, by Dickson's Lemma there would be sequence of events e_i such that $e_i{\downarrow} \prec e_{i+1}{\downarrow}$).

Given that the hypothesis of our paper is

> for nets with a great deal of concurrency the storage required to build (reverse) occurence nets would be smaller than implementations that consider all interleavings

we compute (a) the maximum number of markings that need to be maintained for the traditional backward analysis [AJ98] and (b) the total number of nodes generated by the unfolding algorithm. Given that the storage requirements of a node is bounded by the storage required for a marking, comparing the number of markings from backwards analysis against the total number of nodes in an unfolding is appropriate.

We now report on the results of our experimentation. In Figure 3 we present two versions of token-rings and a buffer. A process in a token ring can be active when it has a token. We considered several experiments with varying numbers of tokens available in the ring and varying numbers of processes. For the buffer example we varied the number of tokens available. The result of our experimentation is reported in Figure 4, where we provide the time taken in seconds and the number of nodes/markings using unfoldings and backward analysis. As can be seen these results do support our hypothesis that when there is a lot of concurrency in a net then unfoldings would require lesser amount of storage than traditional backward analysis (which considers all possible interleavings).

7 Conclusions and Future Work

We have shown how to extend the technique of unfoldings in order to obtain an efficient implementation of a symbolic algorithm for verification of unbounded Petri nets. In contrast to earlier approaches, the algorithm operates over an infinite-state space, using constraints to characterize upward closed sets of markings. Since our algorithm relies on a small set of properties of Petri nets which are shared by other computation models, we believe that our approach can be lifted to a more general setting. In particular we aim to develop a theory of unfoldings for *well-structured systems* [AČJYK96,FS98] a framework which has been applied for verification of several types of infinite-state systems such as timed automata, broadcast protocols, lossy channels systems, relational automata, etc.

Token Ring					
# Proc	# Tokens	Rev. Occ. Net		Backward Analysis	
		# nodes	Time in 10^{-2} secs	# nodes	Time in 10^{-2} secs
4	1	15	3	46	1
4	2	30	13	124	10
4	3	45	29	320	100
4	4	60	60	744	747
4	5	75	105	1573	4344
4	6	90	167	3055	21049
4	7	105	265	5672	84682
4	8	120	357	9976	295815
4	2	30	13	124	10
8	2	62	78	504	334
12	2	94	263	1096	3010
16	2	126	658	1912	14820
20	2	158	1379	2952	52892

A. Experimentation on Token Ring

Token Ring 2					
# Proc	# Tokens	Rev. Occ. Net		Backward Analysis	
		# nodes	Time in 10^{-2} secs	# nodes	Time in 10^{-2} secs
4	1	18	5	97	4
4	2	36	210	458	176
4	3	54	52	1973	4161
8	2	68	108	2112	8724
12	2	100	324	4896	90622
16	2	132	777	*	*

B. Experimentation on Token Ring, Version 2

Buffer				
# Tokens	# RON nodes	Time in 10^{-2} secs (RON)	# Bkwrd nodes	Time in 10^{-2} secs (Bkwrd)
5	50	28	215	39
10	100	177	1367	2591
15	150	549	4898	53507
20	200	1289	*	*

C. Experimentation with buffer

Note:* denotes non-termination after a reasonable amount of time.

Fig. 4. Result of experimentation

This would allow us to extract common concepts, and provide a guideline for developing unfolding algorithms for these classes of systems. Another important direction of future research is the design of efficient data structures for implementation of the unfolding algorithm, and to carry out experiments to study the performance of the algorithm on more advanced examples. We hope to adapt, to our context, reasoning techniques for unfoldings, based on integer programming and constraints, that have been considered in the literature [MR97,Hel99].

Acknowledgements

We wish to thank Javier Esparza and Stefan Römer for patiently answering our questions about unfoldings and their implementations.

This work is supported in part by ARO under grant P-38682-MA, by STINT– The Swedish Foundation for International Cooperation in Research and Higher Education, and by TFR– The Swedish Research Council for Engineering Sciences.

References

[AČJYK96] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc.* 11^{th} *IEEE Int. Symp. on Logic in Computer Science*, pages 313–321, 1996.

[AJ98] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems, 1998. To appear in the journal of Theoretical Computer Science.

[AJKP98] Parosh Aziz Abdulla, Bengt Jonsson, Mats Kindahl, and Doron Peled. A general approach to partial order reductions in symbolic verification. In *Proc.* 10^{th} *Int. Conf. on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 379–390, 1998.

[AKP97] Parosh Aziz Abdulla, Mats Kindahl, and Doron Peled. An improved search strategy for Lossy Channel Systems. In Tadanori Mizuno, Nori Shiratori, Teruo Hegashino, and Atsushi Togashi, editors, *FORTE X / PSTV XVII '97*, pages 251–264. Chapman and Hall, 1997.

[Dic13] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. J. Math.*, 35:413–422, 1913.

[ER99] J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In *Proc. CONCUR '99,* 9^{th} *Int. Conf. on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 2–20. Springer Verlag, 1999.

[ERV96] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In *Proc. TACAS '96,* 2^{th} *Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 87–106. Springer Verlag, 1996.

[FS98] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere. Technical Report LSV-98-4, Ecole Normale Supérieure de Cachan, April 1998.

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, 1993.

[Hel99] K. Heljanko. Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets. *Fundamenta Informaticae*, 37:247–268, 1999.

[LB99] R. Langerak and E. Brinksma. A complete finite prefix for process algebra. In *Proc. 11th Int. Conf. on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 184–195. Springer Verlag, 1999.

[McM95] K.L. McMillan. A technique of a state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.

[MR97] S. Melzer and S. Römer. Deadlock checking using net unfoldings. In *Proc. 9th Int. Conf. on Computer Aided Verification*, 1997.

[Pel93] D. Peled. All from one, one for all, on model-checking using representatives. In *Proc. 5th Int. Conf. on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*, pages 409–423. Springer-Verlag, 1993.

[Val90] A. Valmari. Stubborn sets for reduced state space generation. In *Advances in Petri Nets*, volume 483 of *Lecture Notes in Computer Science*, pages 491–515. Springer-Verlag, 1990.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

A Proof of Lemmas

Proof of Lemma 2 By induction on k.

Base case: Follows from the first step of the algorithm (initializing the value of U).

Induction case: Suppose that $M_{\mathit{fin}} \overset{k}{\rightsquigarrow} M' \rightsquigarrow_t M$. By the induction hypothesis there is an ℓ' and a configuration E' in $U^{\ell'}$ such that $\mathit{mark}(E') \leq M'$. Let C_1 be a maximal subset of $\mathit{Cut}(E')$ such that $\#C_1 \leq t^\bullet$.

There are two cases.

(1) if C_1 is empty. We define $E = E'$. We show that $\mathit{mark}(E)(s) \leq M(s)$ for each place s. We have two subcases.

(1a) if $s \notin t^\bullet$, then $\mathit{mark}(E)(s) = \mathit{mark}(E')(s) \leq \mathit{mark}(E')(s) + {}^\bullet t(s) \leq M'(s) + {}^\bullet t(s) = M(s)$.

(1b) if $s \in t^\bullet$ then we know by maximality of C_1 that $\mathit{mark}(E')(s) = 0$, and hence $\mathit{mark}(E)(s) = \mathit{mark}(E')(s) \leq M(s)$.

(2) if C_1 is not empty. Notice that $\mathsf{ME}_t(C_1)$ holds. Given that our algorithm is fair in selecting transitions that are backward firable, an event $e = (C_1, t)$ will be chosen and added at some point ℓ. We define $E = E' \cup \{e\}$. Clearly, E is a configuration in U^ℓ. Observe that $\mathit{mark}(E) = (\mathit{mark}(E') \ominus t^\bullet) + {}^\bullet t$. This means that $\mathit{mark}(E) = (\mathit{mark}(E') \ominus t^\bullet) + {}^\bullet t \leq (M'(s) \ominus t^\bullet) + {}^\bullet t = M$. □

Proof of Lemma 3 By induction on ℓ.

Base case: Follows from the first step of the algorithm (initializing the value of U).

Induction case: Suppose that in step $\ell + 1$ we add an event $e = (C, t)$ (together with its preset) to U^ℓ. Take any configuration E in $U^{\ell+1}$. If $e \notin E$ then we know that E is also a configuration in U^ℓ, implying the result by induction hypothesis. Hence, we can assume that $E = E' \cup \{e\}$ where E' is a configuration in U^ℓ. By the induction hypothesis we know that there is a marking M' such that $M' \leq \mathit{mark}(E')$ and $M_{\mathit{fin}} \overset{*}{\rightsquigarrow} M'$. We define $M = (M' \ominus t^\bullet) + {}^\bullet t$. Clearly $M' \rightsquigarrow M$ and hence $M_{\mathit{fin}} \overset{*}{\rightsquigarrow} M$. Observe that $\mathit{mark}(E) = (\mathit{mark}(E') \ominus t^\bullet) + {}^\bullet t$. This means that $\mathit{mark}(E) = (\mathit{mark}(E') \ominus t^\bullet) + {}^\bullet t \geq (M'(s) \ominus t^\bullet) + {}^\bullet t = M$. □

Proof of Lemma 4 We show the claim for a configuration $E_2' = E_2 \cup \{e_2\}$ where $e_2 \notin E_2$. The result follows using induction on $|E_2'| - |E_2|$. Let e_2 be of the form (C_2, t). We know that $\mathit{mark}(E_2') = (\mathit{mark}(E_2) \ominus t^\bullet) + {}^\bullet t$. We define C_1 to be a maximal subset of $\mathit{Cut}(E_1)$ such that $\#C_1 \leq t^\bullet$.

There are two cases

1. If C_1 is empty we define $E_1' = E_1$. We have $|E_1'| = |E_1| < |E_2| < |E_2'|$. We show that $\mathit{mark}(E_1')(s) \leq \mathit{mark}(E_2')(s)$ for each place s. There are two subcases.

1a. if $s \notin t^\bullet$. We have $\mathit{mark}(E_2')(s) = \mathit{mark}(E_2)(s) + {}^\bullet t(s) \geq \mathit{mark}(E_1)(s) + {}^\bullet t(s) \geq \mathit{mark}(E_1)(s) = \mathit{mark}(E_1')(s)$.

1b. if $s \in t^\bullet$. By maximality of C_1 we know that $\mathit{mark}(E_1)(s) = 0$. This means that $\mathit{mark}(E_2')(s) = (\mathit{mark}(E_2)(s) \ominus {}^\bullet t(s)) + {}^\bullet t(s) \geq {}^\bullet t(s) \geq \mathit{mark}(E_1)(s) = \mathit{mark}(E_1')(s)$.

2. If C_1 is not empty then by fairness of the algorithm in selecting transitions that are backward firable, an event $e_1 = (C_1, t)$ will be chosen and added at some point ℓ. We define $E'_1 = E_1 \cup \{e_1\}$. It is clear that E'_1 is a configuration in U^ℓ. We have $|E'_1| = |E_1| + 1 < |E_2| + 1 \leq |E'_2|$. We know that $mark(E'_1) = (mark(E_1) \ominus t^\bullet) + {}^\bullet t$. this means that $mark(E'_2) = (mark(E_2) \ominus {}^\bullet t) + {}^\bullet t \geq (mark(E_1) \ominus {}^\bullet t) + {}^\bullet t = mark(E'_1)(s)$. $\square$