USING OPTIMAL SIMULATIONS TO REDUCE REACHABILITY GRAPHS

Ryszard Janicki
Department of Computer Science and Systems
McMaster University
Hamilton, Ontario, Canada, L8S 4K1

Maciej Koutny
Computing Laboratory
The University of Newcastle upon Tyne
Newcastle upon Tyne NE1 7RU, U.K.

ABSTRACT

We here discuss an approach which uses the optimal simulation - a kind of reachability relation - to enable reasoning about important dynamic properties of a concurrent system. The optimal simulation usually involves only a very small subset of the possible behaviours generated by the system, yet provides a sufficient information to reason about a number of interesting system's properties (such as deadlock-freeness and liveness). In this paper we show how the optimal simulation might be used to generate a reachability graph which is usually much smaller than the standard reachability graph of the system; however, both graphs essentially convey the same information about its dynamic behaviour.

INTRODUCTION

High complexity of the design of concurrent programs, such as inherently concurrent communication protocols, made apparent the need for appropriate formal specification methods, and specialised verification techniques enhanced by computer-aided tools for automated analysis of concurrent programs. Examples of the verification techniques include algebraic transformations of CSP and CCS [Hoa85,HM85]; temporal logic model checkers [CG87, CES86]; and invariant methods developed for Petri nets [MS82].

The process of verification of dynamic properties of concurrent systems often involves some kind of reasoning about the complete state-space of the system, e.g. proving deadlock-freeness requires showing that it is not possible to reach a state in which no transition is enabled. Reasoning about the complete state-space of concurrent systems has one serious drawback which is a combinatorial explosion of the state-space. Even a simple concurrent system can generate many hundreds or thousands of states. Moreover, the higher the degree of concurrency of the system is (the degree of concurrency is roughly the number of sequential subsystems) the faster its state-space becomes unmanageable. To cope with this problem a number of sophisticated techniques have been developed, such as *induction* [Kel76] which employs invariants to prove that a property is true in all the states of the system, and *reduced state-space analysis* [Jen87,Val89,God90] in which reasoning about the complete state-space is replaced by the analysis of its reduced representation [MR87].

In [JK89] and [JK89a] we discussed a possibility of defining a fully expressive reachability relation on the system's histories which would be a 'small' subset of the complete reachability relation. We defined such a reduced reachability relation and called it the *optimal simulation*. It enables reasoning about a number of dynamic properties of a concurrent system, and at the same time requires a minimal computational effort. Optimal simulation has been defined in a very general trace-based setting which makes it applicable to different models for concurrency, such as Petri nets, CCS, CSP, or automata-based models.
The reachability graphs of finite-state systems can be regarded as *finite representations of reachability relations*. Since the optimal simulation provides the same information about relevant dynamic properties of the system as the full reachability relation [JK89,JK89a], the reachability graph of optimal simulation (i.e. its finite representation) and the full reachability graph may be considered as equivalent. The optimal simulation is always a subset of the full reachability, but of course it does not mean that the reachability graph of optimal simulation is always (much) smaller than the full reachability graph. However, we do claim that in the case of concurrent systems exhibiting high degree of concurrency (i.e. those with many sequential components), the reachability graph of optimal simulation is much smaller than the full reachability graph. Thus it is advantageous to use optimal simulation as a tool to *reduce the size of reachability graphs of concurrent systems.*

Unfortunately, as opposed to the full reachability graph, in the general case it is not clear how to generate the reachability graph for optimal simulation in an efficient way. In some sense this is a negative side-effect of the above mentioned generality of optimal simulation. In this paper we will outline how such a graph (as we claim, reduced in the majority of cases) can be constructed for Petri nets which can be decomposed onto finite state machines.
We will not prove here that optimal simulation is indeed behaviourally equivalent to the full reachability, nor justify the construction of the optimal simulation relation. These issues have been dealt with in [JK89] which is widely available (Lecture Notes in Computer Science 366), and in [JK89a] which can be sent on request. Proofs of technical results presented in this paper can be found in [JK89,JK90].
Note that our approach is based on the assumption that concurrent behaviours (histories) can be modelled by causal partial orders. We will represent those partial orders by certain equivalence classes of step sequences, generalising the notion of traces of [Maz86].

We would like to point out that the major methodological difference between our approach to minimise reachability graphs and those developed in [Jen87,Val89,God90] is that we do not try to minimise (or even deal with) the full reachability graph in an explicit way. All what we are trying to do is to build a reachability graph which represents the optimal simulation relation (a subset of full reachability). We then make a claim, based on the general properties of optimal simulation, that in majority of cases, such a graph is much smaller than the original full reachability graph.

1 MOTIVATION

Execution paths generated by Petri nets can be represented by *step sequences* - each step being a finite set of transitions executed simultaneously. Consider the Petri net in Fig. 1.1. Its behaviour might be briefly described in the following way: All step sequences must begin with transition a. After that one can simultaneously execute transitions b and c, or execute b followed by c, or execute c followed by b. The net generates three step sequences leading to a deadlock, $\pi_1 = \{a\}\{b,c\}$, $\pi_2 = \{a\}\{b\}\{c\}$ and $\pi_3 = \{a\}\{c\}\{b\}$. Suppose now that we were about to find

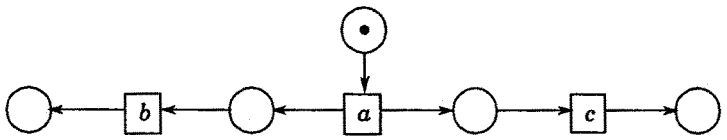


Fig. 1.1

all the deadlocks of the net by following as few step sequences as possible and by selecting possibly shortest step sequences. An exhaustive search would include π_1, π_2 and π_3. But we may observe that all these lead to the same deadlocked marking, and that π_1 is shorter than both π_2 and π_3. Hence an efficient search should include just one path, π_1.

The above example is an instance of the following general problem: *Is there a way of executing a net which is both expressive and efficient?* By an expressive execution we mean one providing enough information to verify relevant properties of the system, e.g. liveness or termination, whereas by an *efficient* execution we mean one which requires minimal computational effort, e.g., by avoiding execution paths providing redundant information. Referring to our example, one may observe that π_1 has a straightforward operational interpretation as it follows the rule: always choose a *maximal* set of independent transitions to be executed next, a rule which characterises *maximally concurrent* execution. Employing maximal concurrency is an attractive idea, both conceptually and from the point of view of implementation. Unfortunately, there are cases in which maximally concurrent execution is not sufficiently expressive (see [JLKD86] for necessary and sufficient condition where it is). To show this we take the net in Fig. 1.2. The maximally concurrent execution can find only one deadlocked marking of the net, by following step sequence $\rho_1 = \{a,b\}\{d\}$. The other deadlocked marking, which might be reached by following $\rho_2 = \{b\}\{c\}$, is left undetected.

In [JK89,JK89a] we have defined, by generalising maximally concurrent execution, the *optimal simulation* which is both expressive and efficient way of executing the net for verification purposes. Fig. 1.2 shows both the full reachability graph of the net and the reachability graph of the optimal simulation. The latter one is smaller, but because in this case only two transitions a and b can be fired concurrently, the difference in size is not significant. However, in the case of net in Fig. 1.3, the reachability graph of the optimal simulation is isomorphic to that in Fig. 1.2, while (as one may easily check) the full reachability graph would hardly fit on a single page.

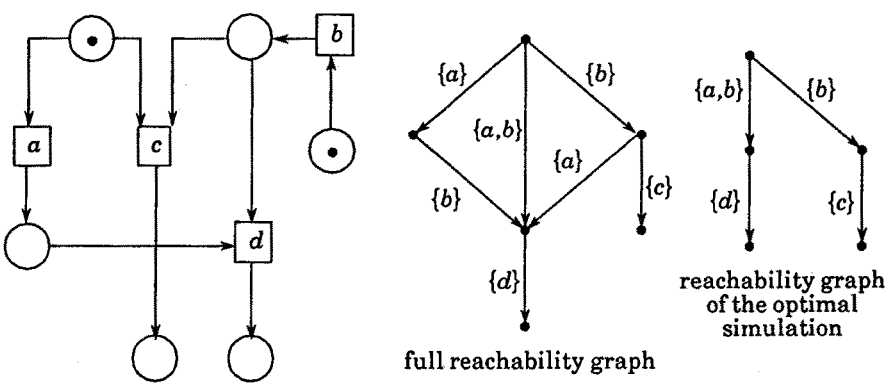


Fig. 1.2

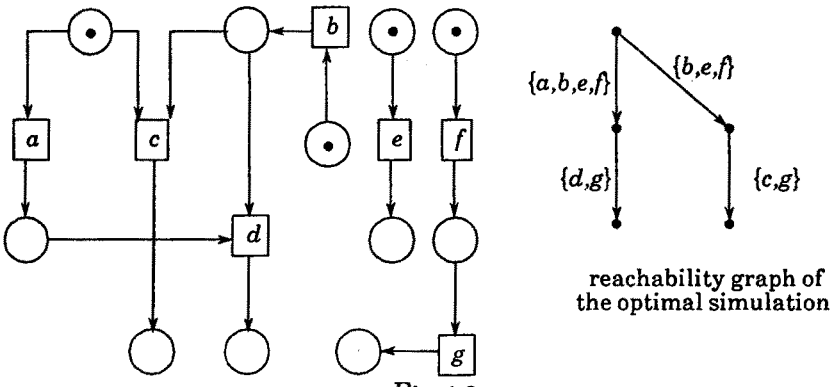


Fig. 1.3

2 SMD NETS AND THE OPTIMAL SIMULATION

In our discussion we will use state machine decomposable (SMD) nets which model non-sequential systems composed of a number of sequential subsystems which synchronise by means of common transitions. SMD can provide semantical basis for more complex Petri nets [Rei85] and other models like COSY [LSC81], CCS [HM85], and CSP [Hoa85] (see [Tau89]).
In the SMD nets, a sequential subsystem is represented by a *finite state machine* which is a triple (S_i,T_i,F_i) such that S_i and T_i are disjoint finite sets of *places* and *transitions*, and $F_i \subseteq S_i \times T_i \cup T_i \times S_i$ is the *flow* relation such that for every $t \in T_i$ there is exactly one s and exactly one p satisfying $(s,t) \in F_i$ and $(t,p) \in F_i$.
An *SMD-net* is a tuple $N=(FSM_1,...,FSM_n,M_{init})$ such that each $FSM_i=(S_i,T_i,F_i)$ is a finite-state machine, $S_i \cap S_j = \emptyset$ for $i \neq j$, and M_{init} is the *initial* marking. (Marking is a set of places which has exactly one place in common with each S_i.) In what follows we will assume that N is fixed and denote: $S=S_1 \cup ... \cup S_n$, $T=T_1 \cup ... \cup T_n$ and $F=F_1 \cup ... \cup F_n$.
For a set of transitions A we denote: $A^\bullet = \{s \mid \exists t \in A.\ (t,s) \in F\}$ and ${}^\bullet A = \{s \mid \exists t \in A.\ (s,t) \in F\}$.
Fig. 2.1 shows an SMD net. As usual, places are represented by circles, transitions by boxes, the flow relation by arcs, and marking by tokens.

Let $ind \subseteq T \times T$ be the set of all pairs of transitions (a,b) such that there is no T_i comprising both a and b. Such a and b are interpreted as *independent*, and only independent transitions can be executed simultaneously.
Let *Ind* be the set of *steps*, each step being a non-empty set of mutually independent transitions, i.e. $Ind=\{A \subseteq T \mid A \neq \emptyset \wedge \forall a,b \in A.\ a=b \vee (a,b) \in ind\}$.
For the net of Fig. 2.1 we have $ind=\{(a,b),(b,a),(a,c),(c,a)\}$ and $Ind=\{\{a\},\{b\},\{c\},\{d\},\{a,b\},\{a,c\}\}$.
We could have defined N as $N=(S,T,F,M_{init})$ for which there are finite state machines (S_i,T_i,F_i) such that $S=S_1 \cup ... \cup S_n$, $T=T_1 \cup ... \cup T_n$, $F=F_1 \cup ... \cup F_n$, and M_{init} is a marking

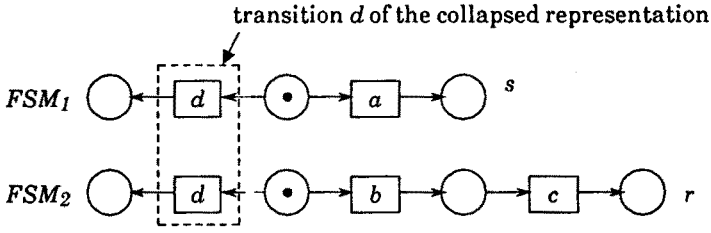


Fig. 2.1

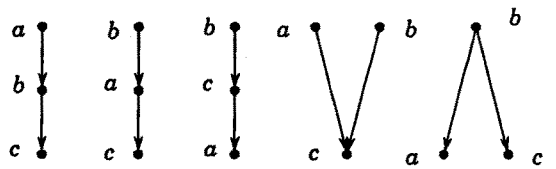
with exactly one token in each S_i (see [Tau89]); however, the 'collapsed' representation - $N=(S,T,F,M_{init})$ - makes the definition of independent transitions less readable.

A *step sequence*, $\sigma \in Steps$, is a sequence of steps $\sigma = A_1 \ldots A_k$ for which there are markings $M_0, M_1, \ldots, M_k$ such that $M_0 = M_{init}$ and for all i, ${}^\bullet A_i \subseteq M_{i-1}$ and $M_i = (M_{i-1} - {}^\bullet A_i) \cup A_i^\bullet$. Later we will denote $mar_\sigma = M_k$. The empty step sequence will be denoted by λ.
For the net of Fig. 2.1 we have $\{a,b\}\{c\} \in Steps$ and $mar_{\{a,b\}\{c\}} = \{s,r\}$, but $\{a,b\}\{d\} \notin Steps$.

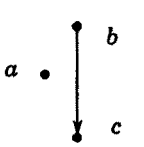
Let $\simeq$ be the least equivalence relation on *Steps* containing all pairs of non-empty step sequences (σ,ω) such that $\sigma = \sigma_1 A \sigma_2$ and $\omega = \sigma_1 A_1 A_2 \sigma_2$, where $A_1 \cap A_2 = \emptyset$ and $A_1 \cup A_2 = A$. The equivalence class of $\simeq$ containing step sequence σ will be denoted by $[\sigma]$. Each equivalence class H of $\simeq$ will be called a *history*, $H \in Hist$. For the net of Fig. 2.1 we have $[\{a,b\}] = \{\{a\}\{b\},\{b\}\{a\},\{a,b\}\}$ and $Hist = \{[\lambda],[\{d\}],[\{a\}],[\{b\}],[\{a,b\}],[\{b\}\{c\}],[\{a\}\{b\}\{c\}]\}$.
Step sequences belonging to H can be seen as different realisations of an underlying concurrent history which itself may be represented by a partial order. This partial order is the intersection of all the partial orders induced by the step sequences in H. This is illustrated in Fig. 2.2 for history $H = [\{a\}\{b\}\{c\}]$ of the net of Fig. 2.1. Note that step sequence $\sigma = \{a\}\{b,c\}\{a\}$ induces a partial order in which the first occurrence of a precedes the occurrences of b and c, and the occurrences of b and c are un-ordered and both precede the second occurrence of a. Every partial order induced by step sequences has the following property: its disorder relation is transitive. Such orders are sometimes called *stratified partial orders*.

The concepts of $\simeq$ and a history $H \in Hist$ are natural generalisations of similar concepts from the theory of *partial commutative monoids* [CF69,Maz86,Zie89]. If we restricted step sequences to just sequences (interleavings) in the definition of *Hist* then we would get exactly the classical notion of Mazurkiewicz traces [Maz86] (the name 'trace' is sometimes used [Hoa85] to mean 'sequence', so we write 'Mazurkiewicz trace' to avoid any confusion). Our representation of a history as a partial order is a natural generalisation of the result of [Szp30] on the representation of partial orders by the set of their linearisations. We represent partial orders by the set of their *stratifications*. The basic advantage of the approach in [Maz86] is that a causal partial order may be represented by just one interleaving. In our case, every history may be represented by just one step sequence. In particular, we may choose the shortest one, i.e. maximally concurrent. This idea will be developed further when the definition of *canonical* step sequence - a very fundamental concept of our approach - will be given.

For every history H, *enabled*(H) is the set of all steps A such that σA is a step sequence for at least one $\sigma \in H$. It turns out that if $\sigma,\omega \in H$ and $A \in enabled(H)$ then $mar_\sigma = mar_\omega$, $\sigma A \in Steps$



partial orders induced by step sequences belonging to history $H = \{\{a\}\{b\}\{c\}, \{b\}\{a\}\{c\}, \{b\}\{c\}\{a\}, \{a,b\}\{c\}, \{b\}\{a,c\}\}$



partial order underlying history H

Fig. 2.2

and $\sigma A \simeq \omega A$. Hence we can define mar_H to be the marking mar_σ, and $H\circ[A]$ to be the history $[\sigma A]$.

In [JK89,JK89a] we introduced the notion of a *simulation* which is a kind of reachability relation on the histories representing a possible mode of executing the net. In this paper we will deal only with two simulations, the *full* and *optimal* ones. The full simulation is simply defined as $FULL=\{(G,H)\in Hist\times Hist \mid \exists A.\ H=G\circ[A]\}$. *FULL* represents the dynamic behaviour of N in a complete way. Its advantage is relatively straightforward definition and natural interpretation, its disadvantage is the size of its reachability graph. Even for small nets the graph grows beyond any manageable size, making the formal verification of the net's properties extremely difficult. It was our goal in [JK89,JK89a] to find possibly smallest simulation which could be used for the verification of relevant net properties. As a solution we proposed the *optimal* simulation, *OPT*. There are three reasons why *OPT* can be regarded as the optimal simulation:

(1) There are a number of behavioural properties which are common to *FULL* and *OPT*. For example, *FULL* and *OPT* generate the same sets of deadlocked markings. It is also possible to verify liveness using *OPT*. Indeed, as we claimed in [JK89a], *FULL* and *OPT* essentially capture *the same behavioural properties of the net* (a concurrent system, in general).

(2) *OPT* involves a minimal set of histories, i.e. each proper subset of *OPT* is less expressive than *FULL*, and it may not be used, e.g., to verify the deadlock-freeness.

(3) The information about the net is generated in *OPT* using the shortest step sequences. For instance, each deadlocked marking will be generated by following the shortest step sequence leading to it.

Moreover, there is no other simulation which would satisfy (1)-(3). *OPT* is defined as follows.

A step sequence $\sigma=A_1...A_k$ is *canonical* if for all $i\geq 2$ and $a\in A_i$ there is $b\in A_{i-1}$ such that $(a,b)\notin ind$. Intuitively, in canonical step sequence the execution of transitions is never delayed (no transition can be moved from A_i to A_{i-1}). It can be shown [CF69,JLKD86] that every history H contains exactly one canonical step sequence, $can(H)$. To define *OPT* we first introduce an auxiliary reachability relation on histories: $CAN=\{(G,H) \mid can(H)=can(G)A\}$. We also define $Hist_{max}$ to be the set of all histories H whose canonical step sequence ends with a maximal step, i.e. if $can(H)=\sigma A$ and $A\subseteq B\in enabled([\sigma])$ then $A=B$.

The *optimal* simulation *OPT* is defined as the smallest subset of *CAN* such that for every $H\in Hist_{max}$ there are histories $H_1,...,H_m$ satisfying: $H_1=[\lambda]$, $H_m=H$, and $(H_i,H_{i+1})\in OPT$, for all $i<m$. We also denote $Hist_{opt}=\{[\lambda]\}\cup\{H \mid (G,H)\in OPT\}$. For the net of Fig. 2.1 we have $Hist_{max}=\{[\{d\}],[\{a,b\}],[\{a,b\}\{c\}]\}$ and $OPT=\{([\lambda],[\{d\}]),([\lambda],[\{a,b\}]),([\{a,b\}],[\{a,b\}\{c\}])\}$.

3 Reachability Graph of the Optimal Simulation

A reachability graph of the full simulation can be defined as: $RG_{FULL}=(V,Arcs,M_{init})$, where $V=\{mar_H \mid H\in Hist\}$ is the set of nodes; $Arcs=\{(mar_H,A,mar_{H\circ[A]}) \mid A\in enabled(H)\}$ is the set of arcs; and M_{init} is the initial node.

The above definition is not very useful in the case of the optimal simulation. The reason is that even if G and H two are histories in $Hist_{opt}$ satisfying $mar_G=mar_H$ and $(H,H\circ[A])\in OPT$, then it does not necessarily follow that $(G,G\circ[A])\in OPT$ (see [JK90] for an example). Hence the construction used to define RG_{FULL} in which histories leading to the same marking were assigned the same node of the graph would not work. To guarantee that $(H,H\circ[A])\in OPT \Leftrightarrow (G,G\circ[A])\in OPT$ we strengthen the condition $mar_G=mar_H$, as follows.

Let $\nabla \subseteq CAN$ comprise all pairs of histories $(G,G\circ[A])$ such that there is a non-empty set $I \subseteq \{1,...,n\}$ satisfying the following (below $R = \cup_{i\in I} S_i$, $U = \cup_{i\in I} T_i$ and $V = \cup_{i\notin I} T_i$):

(3.1) $A \cap U = \emptyset$.

(3.2) There is $t \in U\text{-}V$ such that $\{t\} \in enabled(G)$.

(3.3) There is no $u \in U \cap V$ such that ${}^\bullet u \cap R \subseteq mar_G$.

The last definition is illustrated in Fig. 3.1. The idea behind ∇ is that in all continuations of $G\circ[A]$ in CAN the tokens in the subnets FSM_i, for $i \in I$, will be 'frozen'. Hence, by (3.2), no such continuation can yield a step which is maximal.

For every history H, let Γ_H be the set of all A such that $(H,H\circ[A]) \in CAN\text{-}\nabla$. It can be shown that if $(H,H\circ[A]) \in OPT$ then $A \in \Gamma_H$. Hence when generating the reachability graph of OPT instead of taking $enabled(H)$ as the potential next steps for a history H we can restrict ourselves to the (usually much smaller) set Γ_H. What is, however, more important, Γ_H can be used to identify histories with identical continuations in OPT:

Let $\sim$ be a relation on histories such that $G \sim H$ if $mar_G = mar_H$ and $\Gamma_G = \Gamma_H$. It can be shown that if $G,H \in Hist_{opt}$ and $G \sim H$ then $(G,G\circ[A]) \in OPT$ implies $(H,H\circ[A]) \in OPT$ and $G\circ[A] \sim H\circ[A]$. Hence a reachability graph of the optimal simulation $RG_{OPT} = (V,Arcs,v_{init})$ can be defined in the following way:

(1) $V = \{(mar_H,\Gamma_H) \mid H \in Hist_{opt}\}$.

(2) $Arcs = \{((mar_H,\Gamma_H),A,(mar_{H\circ[A]},\Gamma_{H\circ[A]})) \mid (H,H\circ[A]) \in OPT\}$.

(3) $v_{init} = (M_{init},\Gamma_{[\lambda]})$.

The above definition is an operational one, i.e. it can be used to describe an efficient algorithm constructing RG_{OPT}.

Generating reachability graph RG_{FULL} is usually done in a loop which checks the already generated nodes and steps 'enabled' at those nodes (nodes are labelled with markings). If there exists a node labelled by mar_H and a step A which have not yet been tried, the algorithm generates marking $M = mar_{H\circ[A]}$. It then adds a new node labelled by M and an arc to the graph if M has not yet been a label; otherwise it draws an arc to the node labelled by M.

The algorithm generating RG_{OPT} follows in principle the same pattern. There is, however, one essential difference. An arc cannot be accepted as belonging to RG_{OPT} before another arc, labelled with a maximal step, is found which can be reached from that arc. Hence one first generates an auxiliary reachability graph in a similar way as it is done for RG_{FULL} and then prunes the arcs from which an arc labelled with a maximal step cannot be reached, obtaining RG_{OPT}. A formal description of this algorithm can be found in [JK90].

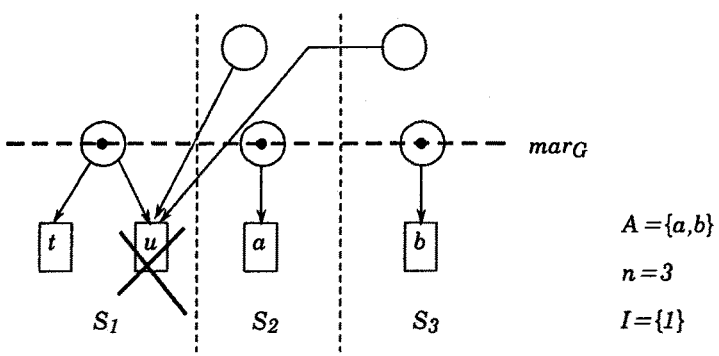


Fig. 3.1: u is excluded by (3.3)

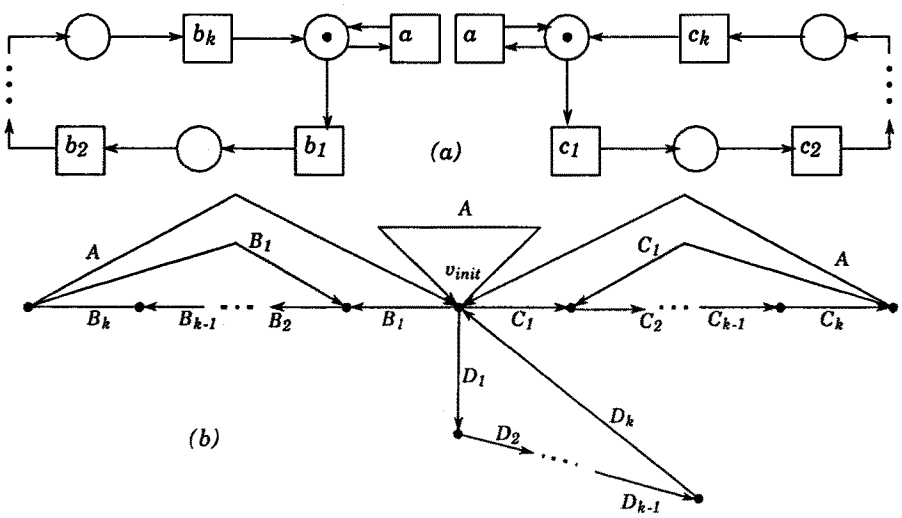


Fig. 3.2

To illustrate the last definition we take the net N_k of Fig. 3.2(a). Fig. 3.2(b) shows RG_{OPT}, where $A=\{a\}$, $B_i=\{b_i\}$, $C_i=\{c_i\}$ and $D_i=\{b_i,c_i\}$. RG_{OPT} has $3k$ vertices and $3k+5$ arcs which compares favourably with RG_{FULL} with k^2 vertices and $3k^2+1$ arcs.

CONCLUDING REMARKS

In this paper we presented the idea behind an algorithm generating reachability graph of the optimal simulation - a way of executing a system directly generalising the maximally concurrent execution [JK89]. Together with the results obtained in [JK89] and [JK89a], this gives a strong indication that the graph RG_{OPT} would in general case be much smaller than the full reachability graph. Furthermore, the higher the degree of concurrency the system exhibits, the more can be gained by using RG_{OPT} instead of RG_{FULL}. There can, however, be situations where RG_{OPT} does not have an apparent advantage over RG_{FULL}. For example, RG_{OPT} of a net can be bigger than RG_{FULL} (see [JK90]). Another problem can be identified by taking the net of Fig. 3.2. If we remove transition b_k and join b_{k-1} with the place holding a token, then although RG_{OPT} for the modified net will be smaller than RG_{FULL}, it will have $o(k^2)$ nodes and arcs. There are two points to be made which show that problems of such kind are less serious than it might look at the first glance.

Let MAX be the maximally concurrent simulation [JLKD86,JK89], and let RG_{MAX} be its reachability graph. (Formally, MAX is the maximal subset of CAN which only involves histories from $Hist_{max}$.) It is not difficult to see that $RG_{MAX} \subseteq RG_{FULL}$ and $RG_{MAX} \subseteq RG_{OPT}$, as well as $MAX \subseteq OPT \subseteq FULL$. Furthermore, OPT is a minimal subset of $FULL$ containing the same behavioural information as $FULL$ does and, intuitively, OPT is only 'slightly' bigger than MAX. As the result, RG_{OPT} is only 'slightly' bigger than RG_{MAX}. On the other hand, $RG_{MAX} \subseteq RG_{FULL}$, and the difference between RG_{MAX} and RG_{FULL} depends strongly on the degree of concurrency exhibited by the net. If the net contains only a few concurrent transitions then the difference between RG_{MAX} and RG_{FULL} is rather small, and in such a case RG_{OPT} might be bigger than RG_{FULL}. When the net contains many concurrent transitions the difference betwe า RG_{MAX} and RG_{FULL} increases dramatically, while RG_{OPT} *is still*

only 'slightly' bigger than RG_{MAX}. This can be well illustrated by taking the nets from Fig. 1.2 and 1.3. The latter has been obtained from the former by adding two simple concurrent subnets. This had no effect on the size of the reachability graph of the optimal simulation, while the size of the full reachability graph has increased significantly.
The problems with the modified net of Fig. 3.2(a) are essentially due to the generality of *OPT*. The optimal simulation has been defined in a pure transition-based setting. In particular, the lack of any reference to the states of a concurrent system makes *OPT* applicable to almost all models for non-sequential computation, but sometimes it may lead to less efficient solution as far as the reachability graphs are concerned. Referring to the modified net of Fig. 3.2(a), by taking into account the particular structure of the states (markings) for this net, we may further reduce RG_{OPT} to a graph which has $o(k)$ nodes and arcs and is still equivalent to the full reachability graph. What this clearly demonstrates is that for specific system models, e.g. for those which support the notion of state, it is possible to modify RG_{OPT} in a way which takes advantage of some particular properties of that model. RG_{OPT} should not therefore be regarded as a complete blueprint for an efficient reduction of the reachability graphs, but in some cases as particularly suitable starting point for developing algorithms for such a reduction.

Although in this paper we consider only nets which can be decomposed onto finite state machines, our approach can be extended to other kinds of nets and models. In [JK90], where all the proofs are given, we use asynchronous automata of [Zie89] as a model of concurrent system. Moreover, [Tau89] enables a translation of our results to CCS and TCSP. In fact, if the behaviour of a concurrent system can be adequately modelled in terms of Mazurkiewicz traces [Maz86], then the approach presented above can always be applied.

Our final comment is that the fusion of our approach with one of the approaches which deal explicitly with reachability graphs, as those of [Val89] and [God90], is likely to lead to highly efficient algorithms for reduced reachability graph generation.

ACKNOWLEDGMENT
We would like to thank Antti Valmari for his comments on the modified net of Fig.3.2. The work of the first author was supported by a grant from NSERC No. OGP 0036539. The work of the second author was supported by ESPRIT BRA 3148 Project DEMON.

REFERENCES

[CF69] Cartier P., Foata D., *Problemes combinatoires de communication et rearrangements*, Lecture Notes in Mathematics 85, Springer 1969.

[CG87] Clarke E.M., Grümberg O., *Research on Automatic Verification of Finite-State Concurrent Systems*, Ann. Rev. Comp. Sci. 2(1987), 269-290.

[CES86] Clarke E.M., Emerson E.A., Sistla A.P., *Automatic Verification of Finite-State Systems using Temporal Logic Specifications*, ACM Transactions on Programming Languages and Systems 8(1986), 244-263.

[God90] Godefroid P., *Using Partial Orders to Improve Automatic Verification Methods*, Proc. of CAV'90, this volume.

[HM85] Hennessy M. and Milner R., *Algebraic Laws for Nondeterminism and Concurrency*, JACM 32(1985), 136-161.

[Hoa85] Hoare C.A.R., *Communicating Sequential Processes*, Prentice-Hall, 1985.

[JLKD86] Janicki R., Lauer P.E., Koutny M., Devillers R., *Concurrent and Maximally Concurrent Evolution of Non-Sequential Systems*, Theoretical Computer Science 43(1986), 213-238.

[JK89] Janicki R., Koutny M., *Towards a Theory of Simulation for Verification of Concurrent Systems*, Lecture Notes in Computer Science 366, Springer 1989, 73-88.

[JK89a] Janicki R., Koutny M., *Optimal Simulation for Verification of Concurrent Systems*, Technical Report No. 89-05, McMaster University,Hamilton, Ontario, 1989.

[JK90] Janicki R., Koutny M., *On Some Implementation of Optimal Simulations*, Technical Report No. 90-07, McMaster University,Hamilton, Ontario, 1990 (also to appear in the ACM/AMS DIMACS series).

[Jen87] Jensen K., *Coloured Petri Nets*, LNCS 254, Springer 1987, pp. 248-299.

[Kel76] Keller R.M., *Formal Verification of Concurrent Programs*, CACM 19(7), 1976, 371-384.

[LSC81] Lauer P.E., Shields M.W., Cotronis J.Y., *Formal Behavioural Specification of Concurrent Systems without Globality Assumptions*, Lecture Notes in Computer Science 107, Springer 1981, 115-151.

[MS82] Martinez J., Silva M., *A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri Net*, Informatik-Fachberichte 52, Springer 1982, 301-310.

[Maz86] Mazurkiewicz A., *Trace Theory*, Lecture Notes in Computer Science 255, Springer 1986, 297-324.

[MR87] Morgan E.T, Razouk R.R., *Interactive State-Space Analysis of Concurrent Systems*, IEEE Transactions on Software Engineering 13(10), 1987.

[Rei85] Reisig W., *Petri Nets*, Springer 1985.

[Szp30] Szpilrajn-Marczewski E., *Sur l'extension de l'ordre partial*, Fundamenta Mathematicae 16 (1930), pp. 386-389.

[Tau89] Tauber D., *Finite Representations of CCS and TCSP Programs by Automata and Petri Nets*, Lecture Notes in Computer Science 369, Springer 1989.

[Val89] Valmari A., *Stubborn Sets for Reduced State Space Generation*, Proc.of the 10th International Conference on Application and Theory of Petri Nets, Bonn, June, 1989.

[Zie89] Zielonka W., *Safe Executions of Recognizable Trace Languages by Asynchronous Automata*, Lecture Notes in Computer Science 363, Springer 1989.