COMPUTING LABORATORY

Net Implementation of Optimal Simulations

R. Janicki, M. Koutny

TECHNICAL REPORT SERIES
No 312 June, 1990
Net Implementations of Optimal Simulations

R. Janicki, M. Koutny

Abstract

We here discuss optimal simulations which provide the designer of a concurrent system with a sound and efficient verification technique enabling reasoning about important dynamic properties of the system. The optimal simulation, involving only a subset of the possible behaviours generated by the system, provides sufficient information to reason about a number of interesting system's properties (such as deadlock-freeness and liveness), and requires a minimal computational effort. In this paper we consider the implementation aspects of the optimal simulation. Our main contribution is an algorithm generating a reachability graph of the optimal simulation for the class of systems which can be adequately modelled by Petri nets composed of finite state machines. Such a reachability graph is usually much smaller than the standard reachability graph of the net, yet both graphs essentially convey the same information about the system's behaviour.


Series Editor: M.J. Elphick

©1990 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

JANICKI, Ryszard

Net implementation of optimal simulations. [By] R. Janicki and M. Koutny

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1990.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 312)

Added entries
KOUTNY, Maciej
UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 312

Abstract

We here discuss optimal simulations which provide the designer of a concurrent system with a sound and efficient verification technique enabling reasoning about important dynamic properties of the system. The optimal simulation, involving only a subset of the possible behaviours generated by the system, provides sufficient information to reason about a number of interesting system's properties (such as deadlock-freeness and liveness), and requires a minimal computational effort. In this paper we consider the implementation aspects of the optimal simulation. Our main contribution is an algorithm generating a reachability graph of the optimal simulation for the class of systems which can be adequately modelled by Petri nets composed of finite state machines. Such a reachability graph is usually much smaller than the standard reachability graph of the net, yet both graphs essentially convey the same information about the system's behaviour.

About the author

Dr. R. Janicki has been at the Department of Computer Science and Systems, McMaster University, as an Associate Professor since March 1986. Prior to this he was at Aalborg University Center, Denmark, in 1984-86 and at Warsaw Technical University, Poland, in 1975-84.

Dr. M. Koutny has been at the Computing Laboratory as a Research Associate from March 1985 until Nov. 1986 and as a Lecturer from December 1986. Prior to this he was at Warsaw Technical University in Poland from 1982.

Suggested keywords

CONCURRENCY
FINITE STATE MACHINES
OPTIMAL SIMULATION
PETRI NETS
REACHABILITY GRAPH

Suggested classmarks (primary classmark underlined)

Dewey (18th): 511.5 001.64404
U.D.C. 519.711 519.687
**Net Implementation of Optimal Simulations**

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 optimal simulations which provide the designer of a concurrent system with a sound and efficient verification technique enabling reasoning about important dynamic properties of the system. The optimal simulation, involving only a subset of the possible behaviours generated by the system, provides a sufficient information to reason about a number of interesting system's properties (such as deadlock-freeness and liveness), and requires a minimal computational effort. In this paper we consider the implementation aspects of the optimal simulation. Our main contribution is an algorithm generating a reachability graph of the optimal simulation for the class of systems which can be adequately modelled by Petri nets composed of finite state machines. Such a reachability graph is usually much smaller than the standard reachability graph of the net, yet both graphs essentially convey the same information about the system's behaviour.

---

2) Research partly supported by a grant from NSERC No. OGP 0036539.
3) Research supported by ESPRIT Basic Research Action 3148 (project DEMON).
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. We can identify at least three major streams of mathematical models within which concurrent systems can be tackled: (i) the automata-based models, e.g. Petri nets [Pet81,Rei85] and COSY [LSC81]; (ii) the algebraic models based on axiomatic approach, e.g. CSP [Ho85] and CCS [Mil80]; and (iii) the temporal logic and temporal logic related models, e.g. [MP83,PW84]. These different models have given rise to a number of interesting verification techniques, such as the algebraic transformations of CSP [BHR84] and CCS [HM85]; the temporal logic proof checkers of [Bro86,CG87,CES86]; and a variety of invariant methods applied to Petri nets [MS82], the origin of which can be traced back to the deadlock-trap analysis of the free-choice nets [Hac74].

The process of verification of dynamic properties of a concurrent system generally 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 further transition is enabled, and proving liveness requires showing that starting from any reachable state it is possible to execute any given action. Reasoning about the complete state-spaces 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. This is further worsened by the fact that the greater the degree of concurrency exhibited by the system is (the degree of concurrency is roughly the number of its sequential sub-systems), the quicker the state-space grows beyond any manageable size. To cope with the combinatorial explosion a number of sophisticated techniques have been invented. Perhaps the most attractive is induction, e.g. [Kel76], which employs invariants to prove that a property is true in all the states of the system. Although such invariants can sometimes be derived automatically [MS82], one usually has to formulate them without any external assistance, which severely limits the applicability of the invariant techniques. An alternative approach is one in which reasoning about the complete state-space is replaced by the analysis of its (reduced) representation [MR87]. Examples of an approach which is aimed at generating reduced state spaces of concurrent systems are [Jen87] and [Val89].

In [JK87,JK89 and JK89a] we investigated a possibility of defining a reachability relation on the system's histories which would be a 'small' subset of the complete reachability relation, and yet provide full information about the relevant dynamic properties of the system. We defined such a reduced reachability relation, called the optimal simulation, which enables the reasoning about many interesting properties of the system, and at the same time requires a minimal computational effort.

The optimal simulation has been defined in a general, trace-based, setting what makes it applicable to a number of models for concurrency, including Petri nets, and large subsets of CSP and CCS. This generality, however, had a negative side-effect in that it was not possible to come up with an efficient implementation of the optimal simulation in the general case.

In this paper we discuss the implementation aspects of the optimal simulation in more concrete framework, namely that of the Petri nets which can be decomposed onto finite state machines. The
nets in that class enjoy clear structural properties making possible an efficient algorithm generating a reachability graph of the optimal simulation.

The paper is organised as follows. In the next section we briefly outline the motivations and basic idea behind the optimal simulation. In Section 2 we introduce the state machine decomposable nets, and recall the main notions of [JK89a]. The third section contains a formal definition of reachability graphs. The two final sections present an algorithm generating a reachability graph of the optimal simulation.

Throughout the paper we use standard mathematical notation. In particular, if \( \sigma = A_1 \ldots A_k \) is a string and \( A \) is an element, then \( \sigma^* A \) denotes the string \( A_1 \ldots A_k A \).

1 Motivation

Execution paths generated by a Petri net can be represented by step sequences - each step being a finite set of transitions executed simultaneously. Consider the Petri net in Figure 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 deadlocked marking, \( n_1 = (a)(b,c) \), \( n_2 = (a)(b)(c) \) and \( n_3 = (a)(c)(b) \).

Suppose now that we were asked to find all the deadlocked markings of the net by inspecting as few step sequences as possible and by selecting possibly shortest sequences. An exhaustive search would include all three step sequences, \( n_1 \), \( n_2 \) and \( n_3 \). However, one may easily observe that all these lead to the same marking \( \{s,r\} \). One may also observe that \( n_1 \) is shorter a sequence than both \( n_2 \) and \( n_3 \). Hence an efficient search detecting the deadlocked markings of the net should include just one path, \( n_1 \).

![Figure 1.1](image_url)
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 some important 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 $\pi_1$ 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 as expressive as we would like it to be. To show this we take the net in Figure 1.2.

The maximally concurrent execution can find only one deadlocked marking of the net, $\{s\}$, by proceeding along the step sequence $\rho_1 = \{a,b\} \{d\}$. The other deadlocked marking, which might be reached through $\rho_2 = \{b\} \{c\}$ is left undetected.

In [JK89,JK98a] we have defined, by generalising the maximally concurrent execution, the optimal simulation which is both expressive and efficient way of executing the net for verification purposes. In this case of net in Figure 1.2, optimal simulation would consist of two step sequences, $\rho_1$ and $\rho_2$, together with their prefixes.

In this paper we will focus on the problem of finding a reachability graph (i.e. a finite representation) of the optimal simulation. There are two reasons why generating such a reachability graph seems to be an important issue. Firstly, if we were able to generate the complete reachability graph of optimal simulation of the net, then it would be in general advantageous to use this graph for verification purposes rather than (usually much larger) graph which generates all the step se-

![Figure 1.2](image-url)
quences of the net. Secondly, if the net would generate reachability graph of the size we could not possibly handle, we might use the algorithms developed for the purpose of generating the reachability graph, to implement an efficient simulation technique based on the concept of optimal simulation.

2 SMD NETS AND THE OPTIMAL SIMULATION

In this paper we will consider the state machine decomposable (SMD) nets. An SMD net can be interpreted as a model of a non-sequential system composed of a number of sequential sub-systems. The SMD nets enjoy an important property: the independency relation on the transitions can be derived from the structural properties of the net, and depends only on the distribution of transition among the sequential sub-systems. Note that although the SMD nets have very simple structure, they can model a wide range of concurrent systems, and provide a semantical basis for more complex Petri net classes or other system models like COSY [LSC81], or substantial sub-models of CCS [Mil80] and CSP [Hoa85].

A finite-state machine is a triple $FSM = (S, T, F)$ such that $S$ and $T$ are non-empty disjoint finite sets of respectively places and transitions, and $F \subseteq S \times T \times T \times S$ is the flow relation such that for every $t \in T$, $\text{card}(\{s \mid (s, t, t) \in F\}) = \text{card}(\{s \mid (t, s) \in F\}) = 1$. Figure 2.1 shows two examples of finite state machines.

Note that $FSM_1 = \{(s_1, s_2, p); (a, b), (s_1, a), (s_1, b), (a, s_2), (b, p)\}$.

As usual, places are represented by circles, transitions are represented by boxes, and the flow relation is represented by arcs.

An SMD-net $N$ is an $n + 1$-tuple $(n \geq 1)$ $N = (FSM_1, ..., FSM_n, M_{init})$ such that each $FSM_i = (S_i, T_i, F_i)$ is a finite-state machine; and $M_{init} \subseteq S_1 \cup ... \cup S_n$ is the initial marking satisfying $\text{card}(M_{init} \cap S_i) = 1$ for all $i$. We will denote $S = S_1 \cup ... \cup S_n$, $T = T_1 \cup ... \cup T_n$, and $F = F_1 \cup ... \cup F_n$.

**Remark**

We could have defined $N$ as $N = (S, T, F, M_{init})$ for which there are $FSM_1, ..., FSM_n$ satisfying the above definition. Indeed, there would almost be no difference in the subsequent treatment bet-

---

![Figure 2.1: Finite state machines](image.png)
ween the two representations of $N$. However, the 'collapsed' representation $N=(S,T,F,M_{\text{init}})$ lacks the explicit representation of the net's structure needed to introduce the independency relation on which our whole approach is based. We also note that $N=(S,T,F,M_{\text{init}})$ may admit more than one decompositions onto the $FSM_i$'s. We do not care which one is taken as long as we have one. □

If we take $FSM_1$ and $FSM_2$ as in Figure 2.1 and add the initial marking $M_{\text{init}}=\{s_1,r_1\}$ then the resulting SMD net $N_1=(FSM_1,FSM_2,M_{\text{init}})$ can be represented as in Figure 2.2(a), where the marking is represented by placing tokens inside places. The collapsed version of $N_1$ is shown in Figure 2.2(b).

Let $N=(FSM_1,...,FSM_n,M_{\text{init}})$ be an SMD net fixed for the rest of this paper.
For every $t \in T$, let $\text{index}_t=\{i \mid t \in T_i\}$, and for $A \subseteq T$, let $\text{index}_A = \bigcup_{t \in A} \text{index}_t$.
For $x \in S \cup T$, $x^* = \{y \mid (x,y) \in F\}$ and $^*x = \{y \mid (y,x) \in F\}$.
The dot-notation extends in a natural way to sets of places and transitions.
A marking of $N$, $M \in \text{mar}$, is any subset $M$ of $S$ such that $\text{card}(M \cap S_i) = 1$ for all $i$.

![Figure 2.2: Explicit and collapsed representation of SMD net](image-url)
For the net of Figure 2.2(a) we have $\text{index}_a = \{1\}$ and $\text{index}_b = \{1, 2\}$. Also, $b^* = \{p\}$, $s_1^* = \emptyset$, and $\text{mar} = \{s_1, r_1\} \cup \{s_1, r_2\} \cup \{s_2, r_1\} \cup \{s_2, r_2\} \cup \{p\}$.

Let $\text{ind} \subseteq T \times T$ be a relation comprising all pairs of transitions $(a, b)$ such that $\text{index}_a \cap \text{index}_b = \emptyset$. If $(a, b) \in \text{ind}$ then $a$ and $b$ are interpreted as independent transitions, and only independent transitions can be executed simultaneously.

We then define $\text{Ind}$ to be a set of all steps which are non-empty sets of mutually independent transitions, i.e., $\text{Ind} = \{A \subseteq T \mid A \neq \emptyset \land \forall a, b \in A. a = b \lor (a, b) \in \text{ind}\}$.

For the net of Figure 2.2(a) we have $\text{ind} = \{(a, c), (c, a), (a, c), (a, c), (a, c), (a, c)\}$ and $\text{Ind} = \{(a), (b), (c), (a, c), (a, c), (a, c)\}$.

A step sequence [SM81], $\sigma \in \text{Steps}$, is a string $\sigma = A_1 \ldots A_k$ such that $k \geq 0$, $A_i \in \text{Ind}$ (for $i = 1, \ldots, k$), and there are markings $M_0, M_1, \ldots, M_k \in \text{mar}$ satisfying the following.

1. $M_0 = M_{\text{init}}$.
2. For $i = 1, \ldots, k$, $A_i \subseteq M_{i-1}$ and $M_i = (M_{i-1} \cup A_i^*)$.

The empty step sequence $(k = 0)$ will be denoted by $\lambda$. We also denote $\text{mar}_\sigma = M_k$.

For the net of Figure 2.2(a) we have $\text{Steps} = \{\lambda, (a), (b), (c), (a, c), (a, c)\}$ and $\text{mar}_\lambda = \{s_1, r_1\}$ and $\text{mar}_{(a)} = \{s_2, r_2\}$.

The semantics of $N$ in terms of step sequences can be refined by identifying those step sequences which are essentially different realisations of the same underlying partial order history of the net.

Let $\equiv$ be the least equivalence relation on step sequences which contains all pairs of non-empty step sequences $(\sigma, \omega)$ such that $\sigma = A_1 \ldots A_k, A_k^+ \ldots A_l$ and $\omega = A_1 \ldots A_k, A_k^+ A_{k+1} A_{k+2}^+ \ldots A_l$, where $1 \leq k \leq l$, $A_k^+ \subseteq A_k$ and $A_k \cup A_k^+ = A_k$. The equivalence class of $\equiv$ containing step sequence $\sigma$ will be denoted by $[\sigma]$.

Henceforth an equivalence class $H$ of $\equiv$ will be called a history of $N$, $H \in \text{Hist}$.

For the net of Figure 2.2(a) we have $\{(a, c)\} = \{(a), (c), (a), (a, c)\}$ and $\text{Hist} = \{\lambda, [(a)], [(b)], [(c)], [(a, c)]\}$.

Step sequences belonging to a history $H$ can be regarded as different realisations of an underlying concurrent history which itself may be represented by a partially ordered set of event occurrences. This partial order is the intersection of all partial orders induced by the execution paths in that equivalence class. This is illustrated in Figure 2.3(b) for history $H = \{(a), (b), (c)\}$ of the SMD net of Figure 2.3(a). (Note that step sequence $\sigma = (a), (b), (c)$ 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 $a$. For more details the reader is referred to [Maz86].) Note that partial orders underlying the histories correspond to the processes [Rei85] of the SMD net. For the history of Figure 2.3(b) the corresponding process is shown in Figure 2.3(c).

Let for every $H \in \text{Hist}$, $\text{enabled}(H) = \{A \in \text{Ind} \mid \exists \sigma \in H. \sigma \cdot A \in \text{Steps}\}$.

**Proposition 2.1**

Let $H \in \text{Hist}$, $\sigma, \omega \in H$, and $A \in \text{enabled}(H)$.

1. $\text{mar}_\sigma = \text{mar}_\omega$.
2. $\sigma \cdot A \in \text{Steps}$.
3. $\sigma \cdot A = \omega \cdot A$. □
(a) Diagram showing step sequences and partial orders.

(b) Partial orders induced by step sequences belonging to history $H = \{ \{a\}, \{b\}, \{c\}, \{b\}, \{c\}/\{a\}, \{a,b\}/\{c\}, \{b\}/\{a,c\} \}$

(c) Diagram showing partial orders underlying history $H$.

Figure 2.3: Partial order underlying a history and the corresponding process.
The above proposition makes possible the following two definitions.

For every \( H \in Hist \) we denote by \( mar_H \) the marking satisfying \( mar_H = mar_{\alpha} \), for all \( \alpha \in H \).

For every \( H \in Hist \) and every \( A \in enabled(H) \), let \( H\{A\} \) be a history such that \( \forall \alpha \in H\{A\} \) for all \( \alpha \in H \).

**Proposition 2.2**

If \( H, J \in Hist \) and \( mar_H = mar_J \) then \( enabled(H) = enabled(J) \). □

In [JK89, JK89a] we introduced the notion of a *simulation* of the net which is a kind of reachability relation on the histories of the net representing possible modes of executing the net. In this paper we will deal only with two simulations, the *full and optimal* simulations.

The full simulation, \( FULL \subseteq Hist \times Hist \), is defined in the following way.

\[
FULL = \{(G,H) \in Hist \times Hist | \exists A \in enabled(G). \ H = G\{A\} \}
\]

The full simulation 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 the relevant net's properties. A solution we came up with was the *optimal* simulation, \( OPT \). There are three reasons why \( OPT \) is 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, we think that \( FULL \) and \( OPT \) essentially capture the same behavioural properties of the net. For more details on this the reader is referred to [JK89a].

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 execution paths. For instance, each deadlocked marking will be generated through the shortest step sequence leading to it.

Moreover, there is no other simulation which would satisfy (1)-(3) above.

Let \( \zeta \subseteq Ind \times Ind \) comprise all pairs of steps \((A, B)\) such that for every \( b \in B \) there is \( a \in A \) satisfying \((a, b) \in \zeta \). The relation \( \zeta \) may be interpreted as forward dependency on steps.

A step sequence \( \sigma = A_1 \ldots A_k \) is canonical, \( \sigma \in Steps_{can} \), if \( k \geq 0 \) and \((A_{i-1}, A_i) \in \zeta \) for \( i = 2, \ldots, k \).

Intuitively, in canonical step sequence the execution of transitions is never delayed, i.e. no transition can be moved from \( A_i \) to \( A_{i-1} \).

The definition of \( OPT \) rests on the following result.

**Theorem 2.3** [CF69, JLD86]

Every history contains exactly one canonical step sequence. □

The canonical step sequence contained in a history \( H \) will be denoted by \( can(H) \). We also introduce an auxiliary reachability relation, \( CAN \).

\[
CAN = \{(G,H) \in Hist \times Hist | \exists A \in enabled(G). can(H) = can(G) \{A\} \}
\]

Clearly, \( CAN \subseteq FULL \).

For every \( H \in Hist \), let \( maxenabled(H) \) comprise all steps in \( enabled(H) \) which are maximal w.r.t. set inclusion, i.e. \( maxenabled(H) = \{A \in enabled(H) | \forall B \in enabled(H). A \subseteq B \Rightarrow A = B \} \).

Let \( Hist_{maxend} \subseteq Hist \) comprise histories whose canonical step sequences end with a maximal step,
\[ \text{Hist}_{\text{maxend}} = \{ A_1 \ldots A_k \mid k \geq 1 \land A_1 \ldots A_k \in \text{Steps}_{\text{can}} \land A_k \in \text{maxenabled}(A_1 \ldots A_{k-1}) \} \]

Note that \( \text{Hist}_{\text{maxend}} \) is in general a proper subset of \( \text{Hist} \). For example, if we take the net of Figure 2.2(a) then \( \{ a \} \in \text{Hist} \cdot \text{Hist}_{\text{maxend}} \) since \( \{a,c\} \in \text{enabled}(\lambda) \).

Finally, the optimal simulation, \( \text{OPT} \), is defined as the smallest subset of \( \text{CAN} \) such that for every \( H \in \text{Hist}_{\text{maxend}} \) there are \( H_1, \ldots, H_m \) satisfying: \( H_1 = (\lambda), H_m = H \), and \( (H_i, H_{i+1}) \in \text{OPT} \), for \( i = 1, \ldots, m-1 \). We also denote \( \text{Hist}_{\text{opt}} = \{ \lambda \} \cup \{ G \mid (G,H) \in \text{OPT} \} \cup \{ H \mid (G,H) \in \text{OPT} \} \).

For the net of Figure 2.2(a) we have the following:

\[
\begin{align*}
\text{Steps}_{\text{can}} &= \{ \lambda, A, B, C, D \} \\
\text{CAN} &= \{ (\lambda, A), (\lambda, B), (\lambda, C), (\lambda, D) \} \\
\text{maxenabled}(\lambda) &= \{ B, D \} \\
\text{Hist}_{\text{maxend}} &= \{ B, D \} \\
\text{OPT} &= \{ (\lambda, B), (\lambda, D) \}
\end{align*}
\]

where \( A = \{ a \}, B = \{ b \}, C = \{ c \} \) and \( D = \{ a, c \} \).

In this paper we investigate the implementation aspects of the optimal simulation. In [JK93a] we demonstrated that, under certain 'finiteness' assumptions which are satisfied for the SMD nets, there always exists a finite reachability graph of \( \text{OPT} \). In the rest of this paper we will outline the idea behind an efficient algorithm generating relatively small reachability graph of the optimal simulation.

### 3 Reachability Graphs

A reachability graph is a labelled graph \( \text{RG} = (V, \text{Arcs}, \lambda) \), where \( V \) is a non-empty set of vertices with \( \lambda \in V \) being the initial vertex, and \( \text{Arcs} \subseteq V \times \text{Ind} \times V \) is the set of labelled arcs, such that the following hold.

1. If \((p, A, q) \) and \((p, A, r) \) are arcs then \( q = r \).
2. If \((v_1, A_1, v_2), (v_2, A_2, v_3), \ldots, (v_k, A_k, v_{k+1}) \) are arcs such that \( v_1 = \lambda \), then \( A_1 \ldots A_k \in \text{Steps} \).

The reachability graph \( \text{RG} \) generates a relation on the histories of \( N \), \( \text{Rel}(\text{RG}) \subseteq \text{Hist} \times \text{Hist} \), which comprises all pairs of histories \((G, H)\) for which there are arcs \((v_1, A_1, v_2), (v_2, A_2, v_3), \ldots, (v_k, A_k, v_{k+1})\) such that \( v_1 = \lambda \), \( G = [A_1 \ldots A_{k-1}] \), and \( H = [A_1 \ldots A_k] \).

Clearly, \( \text{Rel}(\text{RG}) \subseteq \text{FULL} \).

A reachability graph representing the full semantics, \( \text{RG}_{\text{FULL}} \), can be defined in the following way.

1. \( V = \{ \lambda H \mid H \in \text{Hist} \} \).
2. \( \text{Arcs} = \{ (\lambda H, A, \lambda H[A]) \mid H \in \text{Hist} \land A \in \text{enabled}(H) \} \).
3. \( \lambda = \lambda \).

It is not difficult, using Proposition 2.1 and 2.2, to show that \( \text{RG}_{\text{FULL}} \) is indeed a reachability graph of the full simulation, i.e. \( \text{Rel}(\text{RG}_{\text{FULL}}) = \text{FULL} \).

At this point it is not obvious what might be a finite reachability graph of the optimal simulation. The reason is that Proposition 2.2 does not hold for \( \text{OPT} \). That is, even if \( G, H \in \text{Hist}_{\text{opt}} \) satisfy \( \lambda = \lambda \) and \( (H, H[A]) \in \text{OPT} \) then it does not necessarily follow that \((G, G[A]) \in \text{OPT} \). Take, for example, the SMD net \( N_2 = (\text{FSM}_1, \text{FSM}_2, [p_1, q_1]) \) shown in Figure 3.1. Let \( H = [b] \) and \( G = [c] \). We have \((\lambda, G) \in \text{OPT} \), \((\lambda, H) \in \text{OPT} \), and \( \lambda = \lambda \). On the other hand, \((H, H[b]) \in \text{OPT} \) while
Figure 3.1

$(G,G-[\{b\}]) \in \text{OPT}$ since $(G,G-[\{b\}]) \in \text{CAN}$. Consequently, the construction used to derive $RG_{\text{FULL}}$ in which histories leading to the same marking were assigned the same node of the graph would not work. In Section 5 we will show how to strengthen the condition $\text{mar}_G = \text{mar}_H$ in order to guarantee $(H,H-[A]) \in \text{OPT} \iff (G,G-[A]) \in \text{OPT}$.

4 A CHARACTERISATION OF STEPS IN THE OPTIMAL SIMULATION

In this section we prove a technical result giving a characterisation of steps involved in optimal simulation which will be used in the subsequent construction of the reachability graph.

Let $\mathcal{V} \subseteq \text{CAN}$ comprise all pairs of histories $(G,H)$ such that there is a non-empty set $I \subseteq \{1,\ldots,n\}$ satisfying the following conditions (below $A \in \text{Ind}$ satisfies $H = G[A]$, and $S_I$ denotes $\bigcup_{i \in I} S_i$).

\begin{enumerate}
    \item \label{cond1} $\text{index}_A \cap I = \emptyset$.
    \item \label{cond2} There is $(t) \in \text{enabled}(G)$ such that $I \subseteq \text{mar}_G \cap S_I$.
    \item \label{cond3} If $u \in T$ is such that $\text{index}_u \cap I \neq \emptyset$ and $u \cap S_I \subseteq \text{mar}_G$, then $\text{index}_u \subseteq I$.
\end{enumerate}

The conditions (4.1)-(4.3) are illustrated in Figure 4.1. The idea behind $\mathcal{V}$ is that for all continuations of $H$ in $\text{CAN}$, the tokens in nets $FSM_i$, for $i \in I$, are 'frozen'. Hence, by (4.2), no continuation of $H$ in $\text{CAN}$ can yield a step which is maximal. To prove this formally, we need an auxiliary lemma.

**Lemma 4.1**

Let $(G,H) \in \mathcal{V}$.

1. $H \in \text{Hist}_{\text{maxend}}$.
2. If $(H,J) \in \text{CAN}$ then $(H,J) \in \mathcal{V}$.

**Proof**

1. Follows from (4.1) and (4.2).
2. Let $I \subseteq \{1,\ldots,n\}$ and $A$ be such that (4.1)-(4.3) hold for $(G,H)$, and let $B$ be such that $J = H-[B]$.

We will show that $I$ and $B$ satisfy (4.1)-(4.3) for $(H,J)$.

We first observe that from (4.1) it follows that

\begin{equation}
\text{mar}_G \cap S_I = \text{mar}_H \cap S_I.
\end{equation}

Hence (4.2) and (4.3) hold for $(H,J)$. 

11
Figure 4.1: u is a kind of transition which is excluded by (4.3)

Suppose \( b \in B \) and \( \text{index}_b \cap I = \emptyset \). From (4.4) and (4.3) it follows that \( \text{index}_b \subseteq I \). This, however, contradicts (4.1) and \((A,B) \notin \zeta\). Hence (4.1) holds for \((H,J)\). \(\square\)

For every \(H \in \text{Hist} \), let \(\Gamma_H\) be the set of all \(A \in \text{Ind} \) such that \((H,H(A)) \notin \text{CAN} \cup \forall\).

**Theorem 4.2**

If \((H,H(A)) \notin \text{OPT}\) then \(A \in \Gamma_H\).

**Proof**

From Lemma 4.1 it follows that if \((H,H(A)) \notin \forall\) then \((H[A]) \notin \text{Hist}^{\text{maxend}}\) and, furthermore, there are no \(A_1, \ldots, A_I\) such that

\[
(H[A], H[A][A_1]) \notin \text{CAN}
\]

\[
(H[A[A_1], H[A][A_1][A_2]) \notin \text{CAN}
\]

\[
\ldots
\]

\[
(H[A][A_1], \ldots, [A_{I-1}], H[A][A_1][A_2], \ldots, [A_{I-1}] [A_I]) \notin \text{CAN}
\]

and \((H[A][A_1] [A_2], \ldots, [A_{I-1}] [A_I]) \notin \text{Hist}^{\text{maxend}}\).

Hence \((H,H(A)) \notin \text{OPT} \), a contradiction. \(\square\)

Consequently, in generating \(\text{OPT}\) instead of looking at \(\text{enabled}(H)\) as the potential next steps for a history \(H \in \text{Hist}^{\text{opt}}\), we can restrict ourselves to the (usually much smaller) set \(\Gamma_H\).

It is not difficult to check whether \(A \in \text{enabled}(H)\) belongs to \(\Gamma_H\).

Suppose \(\text{can}(H) = A_1 \ldots A_k\) and \(A \in \text{enabled}(H)\).

We first check whether \((H,H(A)) \notin \text{CAN}\), by taking \(L = \text{index}_{A_1}\) and checking if \(L \cap \text{index}_a = \emptyset\) for all \(a \in A\). All these can be done in \(\text{card}(A_k) + \text{card}(A)\) steps.

To check whether \((H,H(A)) \notin \forall\) we proceed as follows.
One can show that \((H_H(A)) \in \mathcal{V}\) if and only if the final \(I\) satisfies (4.1)-(4.3). The whole process takes no more than \(n\) steps as the sequence of the \(I\)'s generated during the execution of the loop is a strictly decreasing one.

5 Reachability Graph of the Optimal Simulation

The set \(\Gamma_H\) can be used to identify those histories involved in the optimal simulation which have identical continuations in \(OPT\).

Let \(\sim \subseteq Hist \times Hist\) be defined by \(G \sim H\) \(\iff (mar_G = mar_H \land \Gamma_G = \Gamma_H)\).

Theorem 5.1

If \(G, H \in Hist_{opt}\) and \(G \sim H\) then for all \(A \in Ind\) the following hold.

1. \((G_G(A)) \in OPT \iff (H_H(A)) \in OPT\).

2. If \((G, G(A)) \in OPT\) then \(G(A) \sim H(A)\).

Proof

(1) Suppose \((G_G(A)) \in OPT\). By Theorem 4.2, \(A \in \Gamma_G\). Hence, by \(\Gamma_G = \Gamma_H\), \(A \in \Gamma_H\).

Thus \((H_H(A)) \in CAN\).

From \((G, G(A)) \in OPT\) it follows that \(G(A) \in Hist_{maxend}\), or there are \(A_1, \ldots, A_L\) such that

\((G(A), G(A)[A_1]) \in CAN\n
\((G(A)[A_1], G(A)[A_1][A_2]) \in CAN\n
\vdots\n
\((G(A)[A_1]{\ldots}[A_L-1], G(A)[A_1]{\ldots}[A_L-1][A_L]) \in CAN\)

and \(G(A)[A_1]{\ldots}[A_L-1][A_L] \in Hist_{maxend}\).

Thus from \((H, H(A)) \in CAN\) it follows that

\((H(A), H(A)[A_1]) \in CAN\n
\((H(A)[A_1], H(A)[A_1][A_2]) \in CAN\n
\vdots\n
\((H(A)[A_1]{\ldots}[A_L-1], H(A)[A_1]{\ldots}[A_L-1][A_L]) \in CAN\)

and \(H(A)[A_1]{\ldots}[A_L-1][A_L] \in Hist_{maxend}\).

Thus \((H_H(A)) \in OPT\).

(2) By (1), \(H(A) \in Hist\). Clearly, \(mar_H(A) = mar_H(A)\). Furthermore, \(\Gamma_G(a) = \Gamma_H(a)\) follows from \(can(G(A)) = can(H(A)) \iff A\).

We now may define a reachability graph of the optimal semantics, \(RG_{OPT}\), in the following way.

1. \(V = \{\text{\(mar_H(H_H)\)} \mid H \in Hist_{opt}\}\).

2. \(Arcs = \{(\text{\(mar_H(H_H)\)}, A, \text{\(mar_H(A), \Gamma_H(A)\)}) \mid (H, H(A)) \in OPT\}\).

3. \(v_{\text{init}} = (\text{\(M_{\text{init}}, \Gamma_H(A)\)})\).
Theorem 5.2

$RG_{OPT}$ is a finite reachability graph of the optimal simulation.

Proof

$Rel(RG_{OPT}) = OPT$ follows from Theorem 5.1.

The finiteness of $RG_{OPT}$ follows from the finiteness of the net. \qed

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

Our algorithm generating $RG_{OPT}$ will follow in principle the same pattern. There are, however, two essential differences. First, checking whether a specific node exists requires also the comparison of the sets $\Gamma_H$. Second, a newly generated arc cannot be accepted as belonging to $RG_{OPT}$ before another arc, labelled with a maximal step, is found which can be reached from the new arc. Consequently, we first generate an auxiliary graph $G_{aux}$ and then prune the arcs which do not lead to maximal steps, obtaining a graph isomorphic to $RG_{OPT}$.

We first show how to generate the auxiliary graph $G_{aux}$.

Let $G_{aux} = (V, Arcs)$, where $V \subseteq \text{Steps}_{can} \times 2^\text{Ind}$ and $\text{Arcs} \subseteq V \times \text{Ind} \times V$, be a directed graph generated in the following way (below for every $v = (\sigma, \psi) \in V$ we denote $\sigma(v) = \sigma$ and $\psi(v) = \psi$).

\begin{align*}
V & := \{(\lambda, \Gamma_{\text{start}})\}; \\
Arcs & := \emptyset; \\
\text{WHILE not all } v \in V \text{ have } \psi(v) \text{ empty DO} \\
& \hspace{1em} \text{BEGIN} \\
& \hspace{2em} \text{Find any } v \in V \text{ and } A \in \psi(v) \\
& \hspace{2em} \text{IF there is } w \in V \text{ such that } \{\sigma(w)\} \sim \{\sigma(v) \cdot A\} \\
& \hspace{2em} \text{THEN Arcs} := \text{Arcs} \cup \{(v, A, w)\} \\
& \hspace{2em} \text{ELSE} \\
& \hspace{3em} \text{BEGIN} \\
& \hspace{4em} w := (\sigma(v) \cdot A, \Gamma_{\{\sigma(v) \cdot A\}}) \\
& \hspace{4em} V := V \cup \{w\} \\
& \hspace{4em} \text{Arcs} := \text{Arcs} \cup \{(v, A, w)\} \\
& \hspace{3em} \text{END} \\
& \hspace{2em} \psi(v) := \psi(v) \setminus \{A\} \\
& \hspace{1em} \text{END}
\end{align*}
The resulting graph \( G = (V, \text{Arcs}) \) contains a graph isomorphic to \( R G_{\text{OPT}} \). To obtain it we proceed as follows.

Let \( \text{Arcs}_o \) be the set of all \((v,A,w) \in \text{Arcs}\) such that there are arcs
\[(v_1, A_1, v_2), (v_2, A_2, v_3), ..., (v_l, A_l, v_{l+1}), ..., (v_{l+k}, A_{l+k}, v_{l+k+1})\]

satisfying the following
1. \( v_1 = (\lambda, \Gamma[l]) \)
2. \( k \geq 1 \)
3. \( (v_l, A_l, v_{l+1}) = (v, A, w) \)
4. \( A_{l+k} \in \text{maxenabled}[A_1, ..., A_{l+k-1}] \).

Then \( R G_{\text{opt}} = (V_o, \text{Arcs}_o, v_{\text{init}}) \), where \( v_{\text{init}} = (\lambda, \emptyset) \) and \( V_o \) is the set comprising \( v_{\text{init}} \) and all the endpoints of the arcs in \( \text{Arcs}_o \).

It is not difficult to see that \( R G_{\text{opt}} \) is isomorphic to \( R G_{\text{OPT}} \). Hence \( \text{Rel}(R G_{\text{opt}}) = \text{OPT} \).

To illustrate the process of generating \( G_{\text{aux}} \) we take the net of Figure 3.1. In Figure 5.1 we show a possible sequence of steps of the algorithm generating \( G_{\text{aux}} \), where \( A = \{a\} \), \( B = \{b\} \), \( C = \{c\} \) and \( D = \{b, c\} \). We note that in this case \( G_{\text{aux}} = R G_{\text{opt}} \).
Figure 5.1: Generating the graph $G_{aux}$
As another example we consider the net $N_k$ ($k \geq 2$) of Figure 5.2(a). The graph $G_{aux}$ is shown in Figure 5.2(a), where $A = \{a\}$, $B_i = \{b_i\}$, $C_i = \{c_i\}$ and $D_i = \{b_i, c_i\}$. Also in this case $G_{aux} = RG_{opt}$, so no unnecessary arcs and nodes are generated. The reachability graph of the optimal simulation for $N_k$ has $3k$ vertices and $3k + 5$ arcs which compares favourably with the reachability graph of $RG_{FULL}$ which has $k^2$ vertices and $3k^2 + 1$ arcs.

**Concluding Remarks**

In this paper we outlined the idea behind an algorithm generating reachability graph of optimal simulation - a way of executing a system directly generalising the maximally concurrent execution [JK89]. Together with 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 reachability graph of the full execution mode, $RG_{FULL}$. Furthermore, as our experience has shown, the greater the degree of concurrency the system exhibits, the more benefits can be gained by switching from $RG_{FULL}$ to $RG_{OPT}$. There are, however, situations where $RG_{OPT}$ is bigger than $RG_{FULL}$. For example, the reachability graph $RG_{OPT}$ generated in Figure 5.1 is bigger than $RG_{FULL}$. However, the situation is less paradoxical than it might look at the first glance. Let $MAX$ be the maximally concurrent simulation [JK96, JK89], and let $RG_{MAX}$ be its reachability graph. (Formally, $MAX$ is the minimal subset of $FULL$ such that $\{[\lambda], [A] | A \in maxenabled([\lambda])\} \subseteq MAX$, and if $(G, H) \in MAX$ and $A \in maxenabled(H)$ then $(H, H \cap [A]) \in MAX$.) It is not difficult to see that $RG_{MAX} \subseteq RG_{FULL}$ and

![Diagram](image.png)

*Figure 5.2: Reachability graph of the optimal simulation*
$\text{RG}_{\text{MAX}} \subseteq \text{RG}_{\text{OPT}}$, as well as $\text{MAX} \subseteq \text{OPT} \subseteq \text{FULL}$. Furthermore, the relation $\text{OPT}$ is a minimal subset of $\text{FULL}$ containing the same behavioural information as $\text{FULL}$, and intuitively $\text{OPT}$ is only 'slightly' bigger than $\text{MAX}$. As the result, the reachability graph of $\text{OPT}$, $\text{RG}_{\text{OPT}}$, is only 'slightly' bigger than the reachability graph of $\text{MAX}$, $\text{RG}_{\text{MAX}}$. On the other hand, $\text{RG}_{\text{MAX}} \subseteq \text{RG}_{\text{FULL}}$, and the difference between $\text{RG}_{\text{MAX}}$ and $\text{RG}_{\text{FULL}}$ strongly depends on the degree of concurrency exhibited by the net. If the net contains only a few concurrent transitions (as that of Figure 3.1) then the difference between $\text{RG}_{\text{MAX}}$ and $\text{RG}_{\text{FULL}}$ is rather small, and in such a case $\text{RG}_{\text{OPT}}$ might be bigger than $\text{RG}_{\text{FULL}}$. When the net contains many concurrent transitions then the difference between $\text{RG}_{\text{MAX}}$ and $\text{RG}_{\text{FULL}}$ increases dramatically, while $\text{RG}_{\text{OPT}}$ is still only 'slightly' bigger than $\text{RG}_{\text{MAX}}$. If we make the net 'more concurrent' by replacing transitions $b$ and $c$ by two 'transitions cycles', $b_1, b_2, ..., b_k$ and $c_1, c_2, ..., c_k$ (similar to those in Figure 5.2(a)) the situation will change radically as $\text{RG}_{\text{OPT}}$ will have $O(k)$ vertices and arcs, whereas $\text{RG}_{\text{FULL}}$ will have $O(k^2)$ vertices and arcs. Note also that if $\text{RG}_{\text{FULL}}$ is loop-free (i.e. $\text{FULL}$ is finite) then we always have $\text{RG}_{\text{OPT}} \subseteq \text{RG}_{\text{FULL}}$.

Although in this paper we consider only nets which can be decomposed into finite state machines, our approach can be extended to other kinds of nets. The only condition we need is that the dependency relation defined on transitions is constant. Under such an assumption we may use some of the results of [Ja84] to define sequential subcomponents (represented by finite state machines in this paper) and then proceed similarly as in this paper. In fact, if the behaviour of a concurrent system can be adequately modelled in terms of traces of [Maz86], then the approach presented above can always be applied.

The approach presented in this paper bears certain resemblance to the stubborn set method introduced by A. Valmari [Val89]. Although Valmari's approach is based on the sequential mode of execution, it is possible that some of his ideas might improve our results, and vice versa. We intend to investigate the relationship between the two methods for generating reduced state spaces in near future.

References


