1987

Reachability Graphs and Invariants for Stochastic High Level Petri Nets

Jing Dong
Dan C. Marinescu
Chuang Lin

Report Number: 87-686

https://docs.lib.purdue.edu/cstech/595

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
REACHABILITY GRAPHS AND INVARIANTS
FOR STOCHASTIC HIGH LEVEL PETRI NETS

Jing Dong
Dan C. Marinescu
Chuang Lin

CSD-TR-686
May 1987
REACHABILITY GRAPHS AND INVARIANTS FOR
STOCHASTIC HIGH LEVEL PETRI NETS

Jing Dong*
Dan C. Marinescu**
Chuang Lin***
Computer Science Department
Purdue University
West Lafayette, IN 47907

ABSTRACT

The paper investigates the properties of Stochastic High Level Petri nets. The research is motivated by the idea of using SHLPNs as high level descriptions of stochastic models. An algorithm for constructing the reachability graph is presented and the invariants of a Stochastic High Level Petri net are discussed.

1. INTRODUCTION

Stochastic High Level Petri nets are High Level Petri nets, [1], [4] augmented with exponentially distributed firing times. SHLPN are introduced in [6] while applications are discussed in [7] and [8]. The significant applications of SHLPNs are in the area of modeling and performance analysis of multiprocessor systems and communication protocols. The advantage of modeling homogeneous system using Stochastic High Level Petri nets is that the resulting models are simpler, more intuitive and have a smaller number of states.

The present paper attempts to define a linear relationship among the reachable markings in order to find all reachable markings of a Stochastic High Level Petri Net model. Place invariant properties of a SHLPN model are also discussed. The paper investigates the properties of the reachable markings of a SHLPN model and presents an algorithm for construction of the reachability graph by means of classifying all SHLPN reachable markings into either duplicate, equivalent, covering, dead markings or normal markings, and reducing the markings which are either duplicate, equivalent or covering.

The problem of constructing reachability trees for High-Level Petri nets has been investigated by Jensen [2]. There are explicit differences between the approach outlined by Jensen in [2] and [3] and ours. These differences are determined by the structure and properties of SHLPNs. First of all, the tokens flowing through a SHLPN have a variable number of attributes. The first two attributes of any token are its type and identity. Additional attributes may be acquired and/or lost. At the network description time, one has to specify the dynamics of the attribute list for each type of token.

* On leave of absence from Chengdu University of Science and Technology, China
** Work supported in part by ARO grant DAAG03-86-K-0106
*** Permanent address: State Planning Committee, Beijing, China
Another significant characteristics of SHLPNs is that the predicates associated with transitions may contain expressions involving the attributes of the tokens in the input places of the transition. The first problem occurs in connection with “unfolding” of a SHLPN. In case of Colored Petri net, the level of unfolding is determined by the cardinality of the color set and is the same for all places visited by tokens with the same color. In case of SHLPNs, the level of unfolding is a local property of each place and it is determined by the number of significant attributes and by the cardinality of the domain of each attribute. The problem of testing for equivalent markings is slightly more difficult in case of SHLPNs than in case of High Level Petri nets. In fact, each predicate associated with a transition induces a certain partitioning based upon the symmetry properties of the predicate. For example, the predicate associated with transition G in the example of the philosopher system (see Figure 3 of reference [6]) describes the condition that philosopher \( i \) uses forks \( i \) and \( i + 1 \) in order to eat. This predicate induces a symmetry of type rotation. Let us now consider the predicate associate with transition G in the model of a shared memory multiprocessor system (see the second example of reference [6], Figure 9). It expresses the condition that the common memory must be free in order to allow a certain processor to reference it. This predicate induces a symmetry of type permutation.

Another important distinction between our work and the one reported in [2] is that we are investigating the construction of the reachability graph rather than the reachability tree since we need to determine the transition rates among all states of the model. An important objective of our analysis is to determine the transition rates between the compounded states, the so called compounded transition rates.

The paper is organized as follows. Section 2 reviews the definition of SHLPNs. Section 3 introduces a method to find all reachable markings for the SHLPN model using the methods of linear algebra, and analyzes some invariant properties of the SHLPN model. In Section 4, we discuss the concepts of duplicate, equivalent, covering and dead markings, and give an algorithm to form the SHLPN reachability graph and for determining the compound transition rates.

2. STOCHASTIC HIGH LEVEL PETRI NETS

In order to solve the SHLPN reachability problem, it is necessary to review some basic concepts of the SHLPNs. Let us start with a formal definition.

1. A SHLPN model is a directed 3-tuple graph \((P, T, A)\) where
   - \(P\) is the set of places
   - \(T\) is the set of transitions
   - \(A\) is the set of arcs; \(A \subseteq (P \times T) \cup (T \times P)\)

2. A structure set \(\Sigma\) consisted of some types of individual tokens \((u_i)\) together with some operations \((o_i)\) and relations \((r_i)\), where \(i\) is token identity.

3. An arc labeled with \(n\)-attributes of token variables.

4. As an inscription on some transition, with a logical formula associated with the operations and relations of the structure \(\Sigma\) and variables existed at the surrounding arcs.

5. A marking associated with \(n\)-attributes of individual tokens.

6. A natural number \(K\) assigned to a place as an upper bound for the number of copies of the same token.
(7) **Firing rule.** Each element of $T$ represents a class of possible changes of markings. Such a change, called as transition firings, means removing tokens from a subset of places and adding them to other subset of places according to the expressions of labeling the arcs. A transition is enabled whenever, given an assignment of individual tokens to the variables which satisfy the predicate associated with the transition, all input places send enough copies of tokens, and the upper bound of all output places will not be exceeded by adding the corresponding copies of tokens. The state space of the system consists of the set of all markings involved the initial marking through such occurrences of firings.

In the SHLPN model, each place and each transition stands for a set of places and a set of transitions in the corresponding Stochastic Petri nets [6], [10]. An arc is labeled by the related token variables, called token attributes, i.e., its type, its identity, its environment, etc., and a token variable has a domain covering the set values of the attributes. The transition of the tokens is realized by the operation of the expressions of labeling arcs which are described according to the related token attributes.

As examples, we consider two SHLPN models from references [6] and [8]. Figure 1 shows the model of a multiprocessor system with 5 processors, 3 common memory modules and 2 busses. Figure 2 presents the model of a Multi-FLEX system with two types of synchronization conditions. The first one is a group synchronization in which a processor must wait for all other processors in its group to complete the execution in the local domain before migrating to the local shared domain. A group consists of three neighbor processors. The second synchronization condition is system synchronization. In order to execute in the global domain, a processor must wait for all other processors in the same system to complete the execution in the locally shared domain. The meaning of the places and the transitions shown in Figure 2 is described in detail in [8].

### 3. LINEAR ALGEBRA AND STOCHASTIC HIGH LEVEL PETRI NETS

The treatment of SHLPNs is greatly simplified and made more comprehensible by a linear algebraic representation. The problem of linear algebraic representation in net theory is discussed in depth in [5] and [9] and we follow the general treatment from [12] in connection with P/T nets.

#### 3.1. The Incidence Matrix of Place/Transition Nets With Tokens With Multiple Attributes.

Let us first review basic definitions related to $P/T$ nets. We follow the notations used by reference [12].

**Definition 3.1. Place/Transition Net**

A tuple $(S,T; F,K,M,W)$ is called a $P/T$ net iff:

(a) $(S,T; F)$ is a finite net with $S$ the set of places, $T$ the set of transitions, and $F$ the set of arcs.

(b) $K: S \rightarrow \mathbb{N} \cup \{\omega\}$ is the capacity (possibly unlimited) of each place. The maximum number of tokens a place may hold is at most its capacity, $K$.

(c) $W: F \rightarrow \mathbb{N}\{0\}$ is a weight associated with each arc of the net. The weight of an arc describes number of tokens which may be carried through that arc.
Definition 3.2. Marking and Follower Marking

Let N be a Place/Transition net.

(a) A marking M of N is a mapping \( M: S_N \rightarrow N \cup \{o\} \) iff \( M(s) \leq K_N(s) \) for all \( s \in S_N \).

(b) A transition \( t \in T_N \) is enabled by marking \( M \) iff

\[
\forall s \in *t: M(s) \geq W_N(s,t)
\]

and

\[
(2) \ s \in t^*: M(s) \leq K_N(s) - W_N(t,s)
\]

(c) If \( t \in T_N \) is enabled by marking \( M \), we call \( M' \) the follower marking of \( M \) iff for each \( s \in S_N \)

\[
M'(s) = \begin{cases} 
M(s) - W_N(s,t) & \text{iff } s \in *t\setminus t^* \\
M(s) + W_N(t,s) & \text{iff } s \in t^*\setminus *t \\
M(s) - W_N(s,t) + W_N(t,s) & \text{iff } s \in *t \cap t^* \\
M(s) & \text{otherwise}
\end{cases}
\]

The following notation is used:

- \( *t \) is the preset of \( t \), the set of input places for transition \( t \).
- \( t^* \) is the postset of \( t \), the set of output places for transition \( t \).
- \( N \) is the set of natural numbers
- \( Z \) is the set of integers.

Condition (b.1) means that the transition \( t \) is enabled iff any input place to it holds at least as many tokens as the weight of the incoming arc, while (b.2) requires that the number of tokens received from the input places should not exceed the available capacity of output places, which is the difference between their maximum capacity and their present occupancy.

If in a P/T net all transitions have the property that condition (b.2) is always satisfied when (b.1) is satisfied, then we call the net a "contact-free" net. We are primarily concerned with contact-free nets.

Condition (c) shows the effects of transition \( t \) firing from marking \( M \) to \( M' \), denoted as \( M[t > M'] \), upon the occupancy of all places in the net. The four equations correspond respectively to input, output, input and output (self-loops) and to places unconnected to transition \( t \).

Definition 3.3. The Incidence Matrix of a P/T net N

The Vector \( i: S \rightarrow Z \) is defined as
\[ L(s) = \begin{cases} W(t,s) & \text{iff } s \in t^*\setminus t \\ -W(s,t) & \text{iff } s \in t^*\setminus t \\ W(t,s) - W(s,t) & \text{iff } s \in t^* \cap t^* \\ 0 & \text{otherwise} \end{cases} \]

(b) The incidence matrix of net \( N, N:(S \times T) \rightarrow Z \) is defined as \( N(s,t) = L(s) \).

Each column of the incidence matrix is a vector \( L_j \) and its \( i \)-th component \( l_{i,j} \) identifies the impact of firing a transition \( t_j \) upon the place \( s_i \).

Let us now consider a \( P/T \) net \( N^a = (S^a, T^a, F^a; K, M, W) \) in which tokens in any places \( s \in S^a \) have a number of significant attributes, \( n_S: S^a \rightarrow N \).

Any token has at least two attributes, its type and its identity, and all attributes are considered significant attributes.

**Definition 3.4.**

The "Unfolded" \( P/T \) Net associated with \( N^a \) is

\[ N_u^a = (S_u^a, T_u^a, F_u^a; K, M, W) \]

such that

1. for every place \( s \in S^a, S_u^a \) has \( \mu_S \) places,
2. for every transition \( t \in T^a, T_u^a \) has \( \mu_S \) transitions,
3. for every arc \( f \in F^a, F_u^a \) has \( \mu_S \) arcs.

\( \mu_S \) is determined by the number of the significant attributes of tokens and by the domain of each attribute.

Clearly, a Stochastic High Level Petri net corresponds a \( P/T \) net \( N^a \), and from the Definition 3.4, each \( P/T \) net, \( N^a \), has an associated "unfolded" \( P/T \) net \( N_u^a \). Figure 3 and Figure 4 are the "unfolded" \( P/T \) nets of the multiprocessor system with SHLPN model shown in Figure 1, and the SHLPN model with group synchronization shown in Figure 2 respectively. It is remarkable that the "unfolded" \( P/T \) net \( N_u^a \) is an exact mapping of the original SHLPN model. The "unfolded" net has the same behavior as the original one. Therefore, we may analyze some characteristics of the SHLPN model, especially the reachability set, by employing the "unfolded" \( P/T \) net \( N_u^a \). An advantage of changing the SHLPN model into a "unfolded" \( P/T \) net \( N_u^a \) is that the transition of tokens is accomplished on the basis of the structure of the "unfolded" \( P/T \) net and the firing rule, instead of the operation of the expressions labeling arcs described by the related token attributes on the SHLPN model.

It is possible to build the incidence matrix for a given SHLPN. For example, Figures 5 and 6 present the incidence matrix for the models in Figures 3 and 4 respectively. The incidence matrix \( N_u(s_i, t_j) \) of the "unfolded" \( P/T \) net \( N_u^a \) shows the change in the marking of \( s_i \) when \( t_j \) fires. It is obvious that the incidence matrix \( N_u(s_i, t_j) \) is entirely determined by the structure of the "unfolded" \( P/T \) net \( N_u^a \). We may consider the incidence matrix as an abstraction representation of the "unfolded" \( P/T \) net \( N_u^a \). On these grounds, we are able to discuss the behavior of the SHLPN model using its incidence matrix \( N_u(s_i, t_j) \).
3.2. The Incidence Matrix of a Stochastic High Level Petri net

Since a Stochastic High Level Petri net can be unfolded into a P/T net, the following discussion is carried out in the framework of a P/T model. Let us consider a marked P/T net, a P/T with an initial marking $M_1$ reflecting the initial distribution of tokens in the places of the net. An execution sequence

$$ES = (M_1, M_2, \ldots, M_j, t_1, t_2, \ldots, t_j)$$

is a sequence such that for all $i \in [2,j]$, the marking $M_i$ is a successor of $M_{i-1}$ when transition $t_i$ fires. We express this as:

$$M_{i-1} = M_i$$

A linear relationship exists between two successive markings of a marked sequence

$$M_i = M_{i-1} + \bar{t}$$

The previous relationship simply states that the successor marking $M_j$ has the same distribution of tokens as its ancestor, $M_{i-1}$, except for the tokens which have been removed from the input places and have been added to the output places as a result of firing of transition $t_i$.

The previous relation can be easily generalized as

$$M_i = M_{i-1} + \bar{N} \cdot \bar{t}$$

with

$\bar{N}$ the incidence matrix of the P/T net

$\bar{t}$ the characteristic vector of the firing sequence $t_1, t_2, \ldots, t_i, \ldots, t_j$. The characteristic vector has the elements corresponding to non-enabled transitions equal to zero and the one corresponding to the enabled transition equal to 1.

In case of a SHLPN with unfolded incidence matrix $N_u$, the previous relation becomes

$$M_i = M_{i-1} + N_u \cdot \bar{t}_u$$

As an example, consider Figure 3 and its incidence matrix shown in Figure 5. A marking will be represented concisely as the number of tokens in the following places of the "unfolded net": $(P, Q_i, Q_j, Q_k, A_i, A_j, A_k, M_j, M_k, B)$. The transitions of the model are represented in the characteristic vector in the following sequence: $(E_i, E_j, E_k, G_i, G_j, G_k, R_i, R_j, R_k)$. For the initial marking $M_1(5,000,000,111,2)$, transitions $E_i$, $E_j$ and $E_k$ are all enabled. Suppose $E_i$ fires, then $\bar{t}_u = (100000000)$. Thus, we have

$$M_2 = M_1 + N_u \cdot \bar{t}_u = (5,000,000,111,2) + (-1,100,000,000,0) = (4,100,000,111,2)$$
where

\[
N_u = \begin{bmatrix}
-1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 1
\end{bmatrix}
\]

The marking \( M_2 \) is reachable from \( M_1 \). The initial marking \( M_1 (5,000,000,111,2) \) corresponds to the system state when each of the 5 processors are in their "private memory" place, no processor is in the "queueing" and "accessing" place, the "common memories", \( M_i \), \( M_j \) and \( M_k \) as well as the two busses are free. The marking \( M_2 (4,100,000,111,2) \) corresponds to the case when the four processors are still in the "private memory" places, one processor in the "queueing" place \( Q_i \) is waiting for the "common memory" \( M_i \), no processor is in the "accessing" places, the three "common memories" \( M_i, M_j, M_k \) and the two busses are all free.

Similarly, let us consider the marking \( M_{15} (0,320,000,111,2) \). In this case the transitions \( G_i \) and \( G_j \) are enabled. If \( G_i \) fires, then,

\[
M_{36} = M_{15} + N_u \cdot \bar{t}_u = (0,320,000,111,2) + (0,-100,100,-100,-1) = (0,220,100,011,1)
\]

where \( N_u \) is as same as before, while \( \bar{t}_u = (000100000) \). \( M_{36} \) is reachable from \( M_{15} \). Clearly the procedure described above may be applied starting with the initial marking and it will eventually lead to the construction of all reachable markings of the net.

Let us now discuss briefly the relationship between the incidence matrix of an original SHLPN model and the incidence matrix of the "unfolded" SLHPN. As we can see from figures 5 and 6, each element \( n_{i,j} \) of the original incidence matrix of the SHLPN model is a scalar and the "unfolding" operation transforms it into a matrix \( [ n_{i,j} ] \).

Each element of the incidence matrix of the original SHLPN model corresponds to a submatrix of the incidence matrix of the "unfolded" model, with a size determined by the level of unfolding associated with transition \( t_j \) and with place \( s_i \). When the transition \( t_j \) and the place \( s_i \) exhibit the same level of "unfolding", then the elements of the incidence matrix of the "unfolded" SHLPN are diagonal matrices of the form \( [ n_{i,j} ] = n_{i,j} \cdot I \), with \( I \) the identity matrix.
3.3. Some Invariant Properties of a SHLPN Model

Various physical phenomena exhibit conservation laws. The same is true in net theory. In a net, if we detect a set of places which do not change their joint total token count when transitions are enabled, then such a set of places is called a place invariant. Knowledge about the set of place invariants of a net helps us to investigate some characteristics of the net. The important properties of an invariant are illustrated by the following lemma, [3]:

**Lemma 3.2:** Let $V$ be a weighted set of places, a vector in $[S^d_a \rightarrow Z]$. If $V \cdot \mathcal{N}_a = 0$, then $V \cdot M' = V \cdot M$ for all markings $M'$ and $M$, where $M'$ is reachable from $M$. Thus $V$ is said to be an invariant.

Proof:

$$V \cdot M' = V \cdot (M + \mathcal{N}_a \cdot \mathcal{I}_a) = V \cdot M + (V \cdot \mathcal{N}_a) \cdot \mathcal{I}_a = V \cdot M$$

Figures 5 and 6 show some invariants of the “unfolded” SHLPNs shown in Figure 3 and 4 respectively. It is necessary to point out that a place invariant is a row vector.

Let us now consider Figure 5. The model of the multiprocessor system has three invariants.

**Inv 1**  $\iff M(P) + \sum_{L=i,j,k} M(Q_L) + \sum_{L=i,j,k} M(A_L) = M_1(P) = 5$

This invariant represents the condition that the five processors are always distributed in the three groups of places, the “private memory” place $P$, the “queueing” group consisting of $Q_i$, $Q_j$ and $Q_k$ as well as the “accessing” group of places, $A_i$, $A_j$ and $A_k$. The actual distributions of the five tokens reflects the current state of the system.

**Inv 2**  $\iff \sum_{L=i,j,k} M(A_L) + \sum_{L=i,j,k} M(M_L) = \sum_{L=i,j,k} M_1(M_L) = 3$

This condition shows that the sum of common memory modules “free” and “busy” is equal to the total number of common memory modules.

**Inv 3**  $\iff \sum_{L=i,j,k} M(A_L) + M(B) = M_1(B) = 2$

This condition indicates that the sum of “free busses”, tokens in the place $B$ and the “busy busses”, tokens in any of the place in the group $A$ is constant and equal to the total number of busses, two. In other words each processor in the “accessing” place $A_L$ ($L = i,j,k$) must take one bus, and at most 2 processors are in the “accessing” states.

The place invariant properties of a net enable us to understand the behavior of the system.
4. AN ALGORITHM FOR CONSTRUCTING THE REACHABILITY GRAPH OF A STOCHASTIC HIGH LEVEL PETRI NET

A reachability graph can be associated with a Petri net such that a node of the graph corresponds to a reachable marking and an arc of the graph corresponds to a transition of the Petri net. In case of a High Level Petri net with an initial marking, all its reachable markings can be organized in a tree structure which will contain all possible sequences of transition firing. In general the reachability tree is infinite. In order to make it useful it is necessary to find ways of reducing it to a finite size. The basic idea is to reduce the size of the tree by defining classes of markings and by including in the tree only one node for each class. The reference [2] presents an algorithm for constructing reachability trees for High-Level Petri nets based upon these ideas.

In Section 3, we have discussed the problem of finding all reachable markings of a SHLPN model by means of linear algebraic techniques and we have analyzed some properties of the SHLPN model. In this section, we present the problem of grouping the reachable markings into classes of either duplicate, equivalent, dead, covering or normal markings and develop an algorithm for constructing the reachability graph by means of reduction of duplicate, equivalent and covering markings.

4.1 Reachability Trees for Predicate/Transition Nets

The reachability tree represents the set of all reachable markings of a P/T net. In the reachability tree an arc is labeled by the transition which connects a marking with its successor.

As an example consider the marked P/T net from [11] which is shown in Figure 7. A marking will be shown as a vector with each component representing the number of tokens in places, $P_1$, $P_2$, and $P_3$ respectively. If the initial marking is $(1,0,0)$ the $t_1$ and $t_2$ transition are enabled. When they fire two new markings, $(1,1,0)$ and $(0,1,1)$ are produced. The new markings are added to the partial tree shown in Figure 8a, the initial marking $(1,0,0)$ being the root of the tree. Let us now consider the new marking $(1,1,0)$. Its successors are the markings $(1,2,0)$ and $(0,2,1)$ obtained when the transitions $t_1$ and $t_2$ fire. In a similar way starting from marking $(0,1,1)$ its successor is $(0,0,1)$ obtained when $t_3$ fires. In this way we form the tree shown in Figure 8b. The procedure is repeated again for the nodes which are the leaves of the new tree and new nodes are added to the tree presented in Figure 8c. If the procedure is repeated every reachable marking will eventually be produced.

In order to make the tree be a useful analysis tool, it is necessary to find the means to limit the tree to a finite size. Examining the characteristics of markings it may be noticed that the reduction to a finite tree can be realized by limiting the new markings produced at each step.

Let us now present a taxonomy of the markings. Based upon this taxonomy a reduction of the reachability tree will be attempted.

**Dead Markings.** Markings in which no transitions are enabled. Notice that the marking $(0,0,1)$ in Figures 8b and 8c is dead. No new markings can be produced from the marking. The node in the tree corresponding to a dead marking is called dead node.

**Duplicate Markings.** Markings which have previously appeared in the tree are called duplicate markings. For example $(0,2,1)t_3 > (0,1,1)$ from Figure 8c, the marking $(0,1,1)$ has been previously produced by firing $t_2$ for the initial marking $(1,0,0)$. The $(0,1,1)$ marking produced by firing $t_3$ for marking $(0,2,1)$ is then a duplicate of the marking $(0,1,1)$ produced by firing transition $t_2$ for the initial marking.
The duplicate markings lead to duplicate nodes in the tree. Only the first occurrence at such node is allowed to develop its successors in the tree.

**Covering Markings.** Let us consider two markings $M$ and $M'$ which belong to the same execution sequence and $M'$ is a successor of $M$. Let us assume that the marking $M'$ has at least as many "identical" tokens in every place as $M$. It follows that any transition which can fire in $M$ can also fire in $M'$. The marking $M'$ is said to cover the marking $M$.

In the tree a class of covering markings is represented by only one node.

We sketch now the algorithm used to construct the reachability tree of a P/T net. Each marking is processed in sequence and it is classified as either dead, duplicate, covering or normal. Let us consider a marking $M_x$, which reflects the disposition of the tokens in all places of the net. We call the node associated with this marking $x$. Then the following steps have to be considered:

1. If there already exists another node $y$ in the tree such that $M_x = M_y$, then the node $x$ is a duplicate node.
2. If no transitions are enabled for the marking $M_x$, then the node $x$ is a dead node.
3. For all transitions $t_j \in T$ which are enabled in $M_x$, create a new node $z$ associated with the marking $M_z$. The marking $M_z$ reflects the new disposition of tokens obtained as a result of firing of the transition $t_j$. Then for some place $p$:
   a. If $M_z(p) = \omega$, then $M_z(p) = \omega$.
   b. If there exists a node $y$ on the path from the root node to $x$ with $M_y(p) < M_z(p)$, then $M_z(p) = \omega$.
   c. Otherwise $M_z(p) = M_z(p)$.

In the third step, an arc labeled $t_j$ is created from node $x$ to node $z$. And node $z$ becomes the new node, then repeat the above 3 steps until all reachable markings have been classified as dead, duplicate, covering or normal markings. Thus the algorithm halts.

4.2 An Algorithm for the Reachability Graph of a Marked SHLPN

A marked SHLPN is a Stochastic High Level Petri net with an initial marking. Stochastic High Level Petri nets are extensions of High-Level Petri Nets more precisely they are High-Level Petri Net augmented with exponentially distributed transition rates associated with all the transitions of the net. In turn High-Level Petri Nets are extensions of regular P/T nets. It follows that the set of reachable markings of the SHLPN model might be infinite as in the case of P/T nets. Therefore it is necessary to reduce the reachable markings to limit the size of the state space of the SHLPN model using the techniques described earlier.

The reachability graph has the markings of the SHLPN as nodes and the transitions of the SHLPN as arcs. Each arc has associated with it the transition rate between the corresponding (compounded) markings of the net. Conceptually the algorithm must first construct the set of all reachable markings and then identify the duplicate markings, the equivalent markings and the covering markings and reduce them in such a way that the graph will contain only one node for each group of markings.

There is also a reachability tree associated with the graph. The reachability tree provides less information about the system than the reachability graph since a number of transitions are missing. The tree cannot be used for constructing the transition matrix of the system, hence for
finding the steady-state solution of the model. Nevertheless the tree is useful in order to deter-
mine the firing sequence from the original marking to any other marking in the reachability set
and can be used for the transient analysis of the system. Moreover the reachability tree is more
readable than the reachability graph and can be used to illustrate some properties of the SHLPN
model.

Since we intend to use a Stochastic High Level Petri Net as a high level description of a
stochastic model, one of the results produced by our algorithm should be in the form of a state
table (see for example Table 2 in [7]). Each entry in the table corresponds to a state, it describes
the set of previous states and the transition rates from each previous state to the current state as
well as the set of post-states and the transition rates from the current state to each of the post-
states. The algorithm should have built-in provisions to actually construct the reachability graph
and the reachability tree, and to display them whenever it is required.

The terms "node of the graph", "marking of the net", and "state of the system" are used
with essentially the same meaning in the following.

The algorithm divides the set of the nodes of the graph into two subsets: processed
nodes, PROC_N, and unprocessed nodes, UNPROC_N. Initially the first set is empty and the
second one contains only one node, namely the node corresponding to the initial marking. The
set of unprocessed nodes is maintained as an ordered list of nodes. The unprocessed nodes are
entered in this list as they are generated by the algorithm and they are processed on a first-
come-first-serve basis. This provision guarantees that all the nodes in the reachability tree
found at level \( i \) are processed before any node of the tree located at level \( i + 1 \). After finishing
the processing of a node, the node is deleted from the second set and added to the set of pro-
cessed nodes. The algorithm terminates when the set of unprocessed nodes becomes empty.
When the algorithm terminates, the first set contains the reachability set of the net with all the
grouping operations already performed.

A very important characteristic of the algorithm is that the actions corresponding to group-
ing of the equivalent markings and determining the compound transition rates are done in the
same time. Rather than constructing the entire state space and then reducing it like other Mar-
kov chain solvers do, in case of the SHLPN model we identify equivalent states as they are gener-
erated. Since the equivalent markings are generally produced by the transitions belonging to the
same group they are easier to identify than in the case of a general Markov chain. We say that
two transitions belong to the same group when they have been obtained as a result of an
"unfolding" operation performed on some transition of the original SHLPN model.

This property of the algorithm is extremely important for two basic resons. First of all it
does reduce the complexity of the computations associated with determining the equivalence
conditions and thus the complexity of the entire algorithm. Even more important is the fact that
the state aggregation technique used is entirely under the control of the investigator of the
model which has to specify the conditions for aggregation at the model definition time. The
aggregation of states carried out in order to reduce the complexity of a model, always leads to
an approximate solution since the system with compound states does provide less information
about the system than the original model. The approach taken in case SHLPN modeling is that
the aggregation should take place only when the information lost due to the aggregation of
states is not relevant for particular measure of performance investigated and for this reason the
aggregation conditions are part of the model definition.

The system maintains internally an "unfolded" version of the SHLPN net. In this version
each SHLPN transition is represented by a set of transitions. The size of this set is determined
by the number of "significant attributes" of the tokens in the input places of the transition and
the range of each of attribute.
To process a node of the graph, called in the following the current node, the following sequence of actions need to be carried out:

- a. Determine the set of all transitions enabled by the marking associated with the current node. This operation may involve the evaluations of the predicates associated with some transitions. If no transitions are enabled classify the current node as a dead node and add it to the set of processed nodes. The dead node corresponds to an absorbing state in the Markov chain describing the system.

- b. Fire each transition of the set and determine the successor node and the transition rate. The transitions belonging to the same group must be fired in sequence in order to simplify the algorithm used for detecting equivalent nodes.

- c. For each successor node determine whether it corresponds to a duplicate marking. In case of a duplicate marking update the corresponding entries of the current node and the entry corresponding to the next node. To update the entry corresponding to the current node means to add a new entity in the next state section of the state table while to update the entry for the next node means to add a new entity in the previous state section of the state table. Each entity is a pair of the form <state, transition rate>. Every time such an update is made one has to check if a previous entity between the same states exists due to the same transition. If this is the case only the transition rate needs to be updated.

- d. If the successor node is not a duplicate node it must be checked if it is equivalent to an existing node. If it is equivalent then the transition rate of the existing equivalent node must be updated.

- e. If the successor node is neither a duplicate node nor an equivalent node in respect to nodes already processed, then add the successor node to the set of unprocessed nodes.

Clearly the most difficult problem we are faced with is the problem of testing for the equivalence condition. In order to be equivalent two markings must satisfy the following necessary conditions:

- EQ1. The distribution of tokens of the same type in all places of the original SHLPN model must be identical.

- EQ2. The distribution of the tokens individualized by one or more attributes, in the "unfolded net" must be either identical or an identical distribution could be obtained for the two markings as a result of a transformation applied to the attributes. The transformation is specified at the model definition time.

- EQ3. If one or more invariance condition over a set of places of the "unfolded net" are specified at the model definition time they have to be checked and the states are equivalent only if the conditions are satisfied.

To clarify these ideas let us examine the model of the multiprocessor system from [6]. The original SHLPN model is shown in Figure 1, the "unfolded net" is presented in Figure 3, its incidence matrix is shown in Figure 5 and the state transition table is shown in Table 1. The reachability tree is shown in Figure 9. In this case the invariance condition is expressed as: the distribution of the sum of the tokens in the "queuing" place, Q and in the "accessing place", A must be the same in the "unfolded net", when there is at least one token in A. The transformation is a permutation on the third attribute of the tokens of type p, which is used to specify the identity of the common memory to be accessed next.

Let us consider first the marking 10 (see Table 1). The distribution of tokens in marking 10 is: (1, 220, 000, 111, 2). Five transition are enabled, \( E_i, E_j, E_k, G_i \) and \( G_j \). When \( E_i \) fires the marking 15 is reached and the distribution of tokens in this marking is (0, 320, 000, 111, 2). When transition \( E_j \) fires a new marking say 15' is obtained, (0, 230, 000, 111, 2). The two
marking are equivalent. Indeed, condition EQ1 is obviously satisfied, and EQ3 is also satisfied since there is no token in place $A$. EQ2 is also true since 230 is obtained as a permutation of 320. Intuitively we accept this equivalence since marking 15 corresponds to the case when two processors which want to access one common memory and three processors which want to access another common memory are all queued in place $Q$ and it is irrelevant whether the group of two processors will access memory $i$ and the group of three processors will access memory $j$ or viceversa. An entirely similar argument can be carried out for the pair of transitions $G_i$ and $G_j$, to show that they are also leading to equivalent markings. Now, when transition $E_k$ fires a new marking say 13 is obtained, $(0, 221, 000, 111, 2)$. This marking is not equivalent to 15 since the condition EQ2 is not satisfied.

Let us now consider the marking 18, $(3, 010, 100, 011, 1)$. The following transitions are enabled: $E_i$, $E_k$, $E_j$, $G_j$, and $R_i$. We concentrate only on the first two transitions. When $E_k$ fires we obtain marking 20, $(2, 011, 100, 011, 1)$ while when $E_i$ fires we obtain marking 21, $(2, 110, 100, 011, 1)$. The two markings are not equivalent since they violate the condition EQ3.

We now discuss briefly the problem of determining the compounded transition rates which are the transition rates among compounded states. In general the transition rates are marking dependent and the rates specified next to a transition in the SHLPN graph are used for reference purpose only. The actual rates are determined by two factors:

- The level of unfolding of the original transition, or the cardinality of the transition group.
- The number of tokens with identical properties in all input places of the unfolded transitions.

We continue with the example used previously. In case of marking 10, $(1, 220, 000, 111, 2)$ and its followers, the equivalent markings 15 and 15' we see that there is only one token in the input place when either $E_i$ or $E_j$ fire hence the total rate is: $1 \times (\lambda_1 + \lambda_1) = 2 \times \lambda_1$. Now let us examine what happens when transitions $G_i$ and $G_j$ fire. Two new markings, marking 25, $(1, 120, 100, 011, 1)$ and marking 25', $(1, 210, 010, 101, 1)$ are obtained. It is easy to show that the two markings are equivalent. The resulting transition rate is now $2 \times \lambda_1 + 2 \times \lambda_1 = 4 \times \lambda_1$.

Now we describe the algorithm to form the SHLPN reachability graph. Let $NEW_N(M, l)$ be a node function which produces a new node, where $M$ is marking associated with the node, and $l$ is node label. $NEW_I(I, info)$ is a function used to append new information, $info$, to an existing node with label $I$. Let $NEW_M(M, t)$ be a marking function. This function produces all marking which can be reached from $M$ for all transitions $t \in T$ enabled in marking $M$. Let $NEW_A(n_1, n_2, t)$ be an arc function. The invocation of the arc function creates a new arc which from source node $n_1$ to the destination node $n_2$, where $t$ is the sequence of transition firings.

The algorithm to form the reachability graph of the SHLPN with the initial marking $M_1$ is

UNPROC_N: $= \{NEW_N(M_1, \text{normal})\};$ PROC_N$:= \emptyset$

REPEAT UNTIL UNPROC_N $= \emptyset$

BEGIN

GET_NEXT_NODE $x \in UNPROC_N$

IF $M_x = M_y$ for some node $y \in PROC_N$

THEN $NEW_I(I_x, \text{"duplicate to } y\text{"})$

ELSE IF $M_x \sim M_y$ for some node $Y \in PROC_N$

THEN $NEW_I(I_x, \text{"equivalent to } y\text{"})$

ELSE IF NO transitions are enabled in $M_x$ for all $t \in T$

THEN $NEW_I(I_x, \text{"dead node"})$

END
FOR all transitions enabled in $M_x$ for $t \in T$ DO
BEGIN
\text{M := NEW}_M(M_x,t); I := normal
FOR ALL ancestors $z$ with $M > M_x$ DO
BEGIN
FOR ALL $p \in P, m(p) > M_x(p)$ DO
BEGIN
\text{M(p) := }\emptyset
\text{NEW}_I(l, "covering z")
END
IF $M \sim M_0$, for some successor from $M_x$ and $M \sim M$, for some $u \in \text{PROC}_N$ THEN DO
BEGIN
\text{NEW}_I(l, "equivalent to u")
a := \text{NEW}_A(x,u,t)
r := \text{NEW}_R(M_x,t)
END
ELSE IF $M = M_0$, for some node $u \in \text{PROC}_N$ THEN DO
BEGIN
\text{NEW}_I(l, "duplicate to u")
a := \text{NEW}_A(x,u,t)
r := \text{NEW}_R(M_x,t)
END
ELSE IF $M \sim M$, for some node $u \in \text{PROC}_N$ THEN DO
BEGIN
\text{NEW}_I(l, "equivalent to u")
a := \text{NEW}_A(x,u,t)
r := \text{NEW}_R(M_x,t)
END
ELSE IF $M \sim M$, for some successor $v$ from $M_x$ THEN DO
BEGIN
\text{NEW}_I(l, "equivalent to v")
a := \text{NEW}_A(x,v,t)
r := \text{NEW}_R(M_x,t)
END
ELSE DO
BEGIN
\text{v := NEW}_N(M,I)
\text{UNPROC}_N := \text{UNPROC}_N \cup \{v\}
a := \text{NEW}_A(x,v,t)
r := \text{NEW}_R(M_x,t)
END
END
\text{UNPROC}_N := \text{UNPROC}_N \setminus \{x\}; \text{PROC}_N := \text{PROC}_N \cup \{x\}
END
5. Conclusions

The paper investigates some of the algorithms necessary in order to design a tool for the performance analysis of concurrent systems based upon Stochastic High Level Petri nets models. The main idea is to use Stochastic High Level Petri nets as high level descriptions of stochastic models.

Some important properties of SHLPN models are discussed first and the concept of "unfolding" a SHLPN is introduced. The properties of the incidence matrix of a SHLPN model are analyzed and the invariants of a model are discussed.

An algorithm for constructing the reachability graph of the system is presented and its remarkable features are discussed.

References

PLACE AND TRANSITION SIGNIFICANCE

P is the "private memory" place. When the place P holds tokens, the corresponding processors are active in their own memory.

Q is the "queuing" place. When the place Q holds tokens, the corresponding processors are queued for common memory.

A is the "accessing" place. When the place A holds tokens, the corresponding processors are accessing common memory.

M is the "idle common memory" place. When the place M holds tokens, the corresponding common memories are idle.

B is the "available bus" place. When the place B holds tokens, the corresponding busses are available.

E is the "end of activity" transition. When it fires, a processor ends its activity in its private memory.

G is the "getting common memory" transition. The transition is enabled when the common memory module and a bus are free. The transition firing time is related to $1/\lambda_2$.

R is the "releasing common memory" transition. After accessing common memory, the processor releases the common memory and the bus and returns to the private memory execution.

Figure 1. The SHLPN model of the multiprocessor system [6].
PLACE AND TRANSITION SIGNIFICANCE

$P_1$ is the "private memory" place. When this place holds tokens, the corresponding processors are active in their own memory.

$P_2$ is the "group synchronization" place. When this place holds tokens, the corresponding processors are waiting for synchronization with their neighbors.

$P_3$ is the "common queuing" place. When this place holds tokens, the corresponding processors are queued for common memory.

$P_4$ is the "common memory" place. When this place holds a token, the corresponding processor is accessing common memory.

$P_5$ is the "common bus" place. When this place holds a token, the common bus is free.

$P_6$ is the "sub-system synchronization" place. When this place holds tokens, the corresponding processors are waiting for synchronization with all others.

$P_7$ is the "global queuing" place. When this place holds tokens, the corresponding processors are queued for global memory.

$P_8$ is the "global memory" place. When this place holds a token, the corresponding processor is accessing global memory.

$P_9$ is the "global bus" place. When this place holds a token, the global bus is free.

$T_1$ is the "end of private activity" transition. When it fires, a processor ends its activity in its private memory.

$T_2$ is the "group synchronization" transition. When a group of processors all finish their private activities, the transition can fire.

$T_3$ is the "getting common memory" transition. The transition is enabled when the common bus is free and there is a processor in the common queue.

$T_4$ is the "end of common activity" transition. When it fires, a processor ends its activity in common memory.

$T_5$ is the "sub-system synchronization" transition. When all processors of the sub-system finish their common activities, this transition can fire.

$T_6$ is the "getting global memory" transition. The transition is enabled when the common bus and the global bus are free and there is a processor in the global queue.

$T_7$ is the "end of global memory" transition. When it fires, a processor ends its activity in global memory.

Figure 2. Modeling of a Multi-Flex System with Processor Synchronization Constraints [8]
Figure 3. The "Unfolded" P/T net of the multiprocessor system with SHLPN.
Figure 4. The "Unfolded" P/T net of the SHLPN model with group synchronization.
Figure 5. The incidence matrix and initial marking as well as invariant vectors of Figure 3.
| \(T_{1i} \) | \(T_{1j} \) | \(T_{1k} \) | \(T_{2i} \) | \(T_{2j} \) | \(T_{2k} \) | \(T_{3i} \) | \(T_{3j} \) | \(T_{3k} \) | \(T_{4i} \) | \(T_{4j} \) | \(T_{4k} \) | \(T_{5i} \) | \(T_{5j} \) | \(T_{5k} \) | \(T_{6i} \) | \(T_{6j} \) | \(T_{6k} \) | \(T_{7i} \) | \(T_{7j} \) | \(T_{7k} \) | \(M_1 \) | Invariants \(Inv_1\) \(Inv_2\) \(Inv_3\) |
| \(P_{1j} \) | -1 | | | | | | | | | | | | | | | | | | | | | |
| \(P_{1j} \) | -1 | | | | | | | | | | | | | | | | | | | | | |
| \(P_{1k} \) | -1 | | | | | | | | | | | | | | | | | | | | | |
| \(P_{2i} \) | 1 | -3 | | | | | | | | | | | | | | | | | | | | |
| \(P_{2j} \) | 1 | -3 | | | | | | | | | | | | | | | | | | | | |
| \(P_{2k} \) | 1 | -3 | | | | | | | | | | | | | | | | | | | | |
| \(P_{3i} \) | 3 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{3j} \) | 3 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{3k} \) | 3 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{4i} \) | 1 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{4j} \) | 1 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{4k} \) | 1 | -1 | | | | | | | | | | | | | | | | | | | | |
| \(P_{6} \) | 1 | 1 | 1 | -9 | | | | | | | | | | | | | | | | | | | |
| \(P_{7i} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{7j} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{7k} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{8i} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{8j} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{8k} \) | | | | | | | | | | | | | | | | | | | | | |
| \(P_{9} \) | | | | | | | | | | | | | | | | | | | | | |

Figure 6. The incidence matrix and initial marking as well as invariant vectors of Figure 4.
Figure 7. A marked P/T net.

Figure 8a. The first step in building a reachability tree.
Figure 8b. The second step in building a reachability tree.

Figure 8c. The third step in building a reachability tree.
<table>
<thead>
<tr>
<th>Marking</th>
<th>Place index</th>
<th>Transition and Post marking</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$ $Q_i$ $Q_j$ $Q_k$ $A_i$ $A_j$ $A_k$ $M_i$ $M_j$ $M_k$ $B$</td>
<td>$E_i \rightarrow 2$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5 0 0 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 3;E_i \rightarrow 4;G_i \rightarrow 17$</td>
</tr>
<tr>
<td>2</td>
<td>4 1 0 0 0 0 0 1 1 1 2</td>
<td>$E_k \rightarrow 5;E_i \rightarrow 6;G_i \rightarrow 18$</td>
</tr>
<tr>
<td>3</td>
<td>3 1 0 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 6;E_i \rightarrow 7;G_i \rightarrow 19$</td>
</tr>
<tr>
<td>4</td>
<td>3 2 0 0 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 8;G_i \rightarrow 20$</td>
</tr>
<tr>
<td>5</td>
<td>2 1 1 1 0 0 0 1 1 1 2</td>
<td>$E_k \rightarrow 8;E_i \rightarrow 9;E_j \rightarrow 10;G_i \rightarrow 21;G_j \rightarrow 22$</td>
</tr>
<tr>
<td>6</td>
<td>2 2 1 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 9;E_i \rightarrow 11;G_i \rightarrow 23$</td>
</tr>
<tr>
<td>7</td>
<td>2 3 0 0 0 0 0 1 1 1 2</td>
<td>$E_k \rightarrow 12;E_i \rightarrow 13;G_i \rightarrow 24;G_j \rightarrow 29$</td>
</tr>
<tr>
<td>8</td>
<td>1 2 1 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 12;E_i \rightarrow 14;E_j \rightarrow 15;G_i \rightarrow 26;G_j \rightarrow 27$</td>
</tr>
<tr>
<td>9</td>
<td>1 3 1 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 13;E_i \rightarrow 15;G_i \rightarrow 25$</td>
</tr>
<tr>
<td>10</td>
<td>1 2 2 0 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 14;E_i \rightarrow 16;G_i \rightarrow 28$</td>
</tr>
<tr>
<td>11</td>
<td>1 4 0 0 0 0 0 1 1 1 2</td>
<td>$G_i \rightarrow 30;G_j \rightarrow 31$</td>
</tr>
<tr>
<td>12</td>
<td>0 3 1 1 0 0 0 1 1 1 2</td>
<td>$G_k \rightarrow 32;G_i \rightarrow 33$</td>
</tr>
<tr>
<td>13</td>
<td>0 2 2 1 0 0 0 1 1 1 2</td>
<td>$G_j \rightarrow 34;G_i \rightarrow 35$</td>
</tr>
<tr>
<td>14</td>
<td>0 3 2 0 0 0 0 1 1 1 2</td>
<td>$G_i \rightarrow 36;G_j \rightarrow 37$</td>
</tr>
<tr>
<td>15</td>
<td>0 5 0 0 0 0 0 1 1 1 2</td>
<td>$G_i \rightarrow 38$</td>
</tr>
<tr>
<td>16</td>
<td>1 3 0 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 22$</td>
</tr>
<tr>
<td>17</td>
<td>2 1 0 0 0 0 0 1 1 1 2</td>
<td>$E_j \rightarrow 21;E_i \rightarrow 39;R_i \rightarrow 2;E_j \rightarrow 22$</td>
</tr>
<tr>
<td>18</td>
<td>3 0 1 1 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 2;E_j \rightarrow 21;E_i \rightarrow 23$</td>
</tr>
<tr>
<td>19</td>
<td>2 0 1 1 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 24;G_j \rightarrow 40;R_i \rightarrow 3$</td>
</tr>
<tr>
<td>20</td>
<td>3 1 0 0 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 3;E_k \rightarrow 24;E_i \rightarrow 26;G_j \rightarrow 41;E_j \rightarrow 25$</td>
</tr>
<tr>
<td>21</td>
<td>2 1 1 0 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 4;E_j \rightarrow 25;E_i \rightarrow 27;E_k \rightarrow 29;G_i \rightarrow 41$</td>
</tr>
<tr>
<td>22</td>
<td>1 1 0 0 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 4;E_j \rightarrow 26;E_i \rightarrow 28$</td>
</tr>
<tr>
<td>23</td>
<td>3 1 0 0 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 5;E_i \rightarrow 30;E_j \rightarrow 33;G_j \rightarrow 42$</td>
</tr>
<tr>
<td>24</td>
<td>1 2 0 0 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 5;G_i \rightarrow 43;E_k \rightarrow 33;E_i \rightarrow 37$</td>
</tr>
<tr>
<td>25</td>
<td>1 1 1 1 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 6;E_k \rightarrow 30;E_i \rightarrow 35;E_j \rightarrow 36;G_j \rightarrow 44$</td>
</tr>
<tr>
<td>26</td>
<td>1 1 1 0 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 7;E_k \rightarrow 31;E_i \rightarrow 34;E_j \rightarrow 37;G_i \rightarrow 44$</td>
</tr>
<tr>
<td>27</td>
<td>1 2 1 1 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 7;E_j \rightarrow 35;E_i \rightarrow 38$</td>
</tr>
<tr>
<td>28</td>
<td>1 2 0 1 0 0 0 1 1 1 2</td>
<td>$E_i \rightarrow 31;G_i \rightarrow 42$</td>
</tr>
<tr>
<td>29</td>
<td>0 2 1 1 1 0 0 0 1 1 1 2</td>
<td>$R_i \rightarrow 8;G_j \rightarrow 46$</td>
</tr>
<tr>
<td>Mark</td>
<td>Value</td>
<td>Next</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>31</td>
<td>03010101011</td>
<td>$G_i \to 46; G_k \to 47; R_j \to 9$</td>
</tr>
<tr>
<td>32</td>
<td>02200011101</td>
<td>$R_k \to 10; G_i \to 48$</td>
</tr>
<tr>
<td>33</td>
<td>01211000111</td>
<td>$G_k \to 48; G_j \to 49; R_i \to 8$</td>
</tr>
<tr>
<td>34</td>
<td>04000101011</td>
<td>$R_j \to 11; G_i \to 50$</td>
</tr>
<tr>
<td>35</td>
<td>03101000111</td>
<td>$R_j \to 9; G_j \to 50$</td>
</tr>
<tr>
<td>36</td>
<td>02201000111</td>
<td>$R_i \to 10; G_j \to 51$</td>
</tr>
<tr>
<td>37</td>
<td>03100101011</td>
<td>$R_j \to 9; G_i \to 51$</td>
</tr>
<tr>
<td>38</td>
<td>04001000111</td>
<td>$R_i \to 11$</td>
</tr>
<tr>
<td>39</td>
<td>30001100010</td>
<td>$R_j \to 17; E_k \to 40; E_i \to 41$</td>
</tr>
<tr>
<td>40</td>
<td>200111100010</td>
<td>$E_i \to 42; E_k \to 45; R_i \to 18; R_j \to 18$</td>
</tr>
<tr>
<td>41</td>
<td>21001100010</td>
<td>$E_k \to 42; E_j \to 43; E_i \to 44; R_i \to 18; R_j \to 19$</td>
</tr>
<tr>
<td>42</td>
<td>111011100010</td>
<td>$E_i \to 45; E_j \to 46; E_i \to 20; E_j \to 21$</td>
</tr>
<tr>
<td>43</td>
<td>111011000010</td>
<td>$R_j \to 21; E_k \to 49; E_i \to 51$</td>
</tr>
<tr>
<td>44</td>
<td>12001100010</td>
<td>$R_i \to 22; R_j \to 23; E_k \to 46; E_i \to 50; E_j \to 51$</td>
</tr>
<tr>
<td>45</td>
<td>100211100010</td>
<td>$R_i \to 22; E_k \to 47; E_i \to 48$</td>
</tr>
<tr>
<td>46</td>
<td>020111100010</td>
<td>$R_i \to 29; R_j \to 26$</td>
</tr>
<tr>
<td>47</td>
<td>030001111000</td>
<td>$R_k \to 27$</td>
</tr>
<tr>
<td>48</td>
<td>012010101000</td>
<td>$R_k \to 25; R_i \to 29$</td>
</tr>
<tr>
<td>49</td>
<td>01111100010</td>
<td>$R_j \to 24$</td>
</tr>
<tr>
<td>50</td>
<td>030011100010</td>
<td>$R_i \to 27; R_i \to 28$</td>
</tr>
<tr>
<td>51</td>
<td>02101100010</td>
<td>$R_i \to 25; R_j \to 26$</td>
</tr>
</tbody>
</table>

Table 1. Reachable markings for the multiprocessor system with SHLPN.
Figure 9. The reachability tree of the model in Figure 1.