Superposed Stochastic Automata: a class of Stochastic Petri nets amenable to parallel solution

Susanna Donatelli*

Dipartimento di Informatica, Università di Torino
corso Svizzera 185, 10149 Torino, Italy
email: susi@di.unito.it

Abstract

In [14] Plateau has introduced an efficient way for solving stochastic processes that are derived from the composition of stochastic automata by making extensive use of the Kronecker (tensor) algebra for matrices. In this paper we apply that efficient solution to a class of Stochastic Petri nets (SPN) [11] that has been called Superposed Stochastic Automata (SSA). The solution has been implemented both with sequential and parallel programs. SSA are a rather restricted subclass of SPN, but the extension to the general case doesn't appear to pose any theoretical problem

1 Introduction

It is well known that Stochastic Petri nets (SPNs) offer a powerful model for the modeling and performance evaluation of systems involving concurrency and non determinism (choices), the major application areas being that of parallel and distributed systems and of communication networks. The major drawback of using SPNs is the large state space that can be generated even by rather simple models.

There have been many attempts to simplify SPN solution, in particular trying to identify subclasses of Petri nets that have a solution of the "product form" type [10,9,8]. All these approaches try to solve the stochastic process associated with an SPN without generating the complete Markov chain. Till now these attempts have provided rather interesting insights on SPNs, but the actual applicability of these results to real problems is rather limited.

In [14,13] Plateau presents a solution method for network of Markovian stochastic automata that has two main advantages. The first is that there is no need to store the infinitesimal generator of the complete Markov chain, as it is expressed as Kronecker sum and product (see [3]) of the infinitesimal generators of the single automata and of a certain number of "correcting matrices": the solution process exploits the particular form of the Kronecker expression to avoid the computation of the infinitesimal generator of the global process. The second one is that the solution process is easily amenable to parallel implementation. The method has been implemented in a tool called PEPS [15] that takes the infinitesimal generator of the single automata and the correcting matrices given by the user, it performs a certain number of checks to control that the dependencies expressed by the network of automata are correct and computes (for the time being with a sequential algorithm) the steady state solution.

In this paper we bring the work described in [14] in the SPN environment. In particular we present a subclass of SPN that we call Superposed Stochastic Automata (SSA) from which we can automatically derive an expression of the type utilized by PEPS. By using SSA we loose some flexibility with respect to networks of stochastic automata, but we gain the advantages of being in a well known framework for which we have well developed tools and a clear semantics. Being in a clear, although rather restricted, framework as that defined by SSA we are able to automatically generate for an SSA a Kronecker expression of the type used by PEPS.

An SSA net is basically a set of stochastic state machines (automata) that interact through synchronization of the rendez-vous type (the classical synchronization in Petri nets). Given an SSA described in graphical form using the GreatSPN package [2], we automatically determine the Kronecker expression for the infinitesimal generator of the SSA and produce all matrices involved in the expression, so that the solution method is completely transparent to the user.

We have used SSA as a testing class of SPN to study the applicability of the efficient solution method based on Kronecker algebra in the field of stochastic Petri nets. The extension to the more general class of Superposed Stochastic Petri nets, where SPNs (in-

*This work has been partially supported by the CNR project "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo"

TH0386-3/91/0000/0054/$01.00 © 1991 IEEE

stead of state machines) interacts through transition synchronization, doesn't appear to bring in any major theoretical problem.

We know of only another attempt to exploit Kronecker algebra to solve complex stochastic Petri nets. In [1] Buchholz applies it to the study of hierarchical markovian models described through queueing networks and Petri nets, but the type of interactions considered is not of the synchronization type as submodels interacts through the exchange of tokens (asynchronous interaction).

This paper is organized as follows. Section 2 introduces the basic definitions of SSA and in Section 3 we derive the Kronecker expression of SSA systems. Section 4 applies the techniques presented in [3,14] to show how to efficiently compute the steady state solution of the Markov process associated to an SSA without having to explicitly compute (and store) its infinitesimal generator. We also discuss the complexity of a sequential implementation, while the major line of a parallel implementation are presented in Section 5 where we also present a few comparative results of sequential vs. parallel implementation. The extension to Superposed Stochastic Petri nets is discussed in Section 6 that concludes the paper.

2 Superposed Stochastic Automata

We introduce here the definition of "Superposed Stochastic Automata," (SSA) a subclass of stochastic Petri nets (SPN) the class of timed nets defined by Molloy in [12,11]. The name for the subclass come from the fact that SSA are the timed, stochastic counterpart of a class of Petri nets introduced by DeMichelis et al. in [4] and that is called "Superposed Automata" nets (SA). The distinctive feature of SSA nets is that they can be considered as a set of stochastic automata combined through a synchronization operator of the rendez-vous type. In Petri net terminology a SSA net is a set of SPN nets of the state machine type composed by superposition of transitions.

We first give a formal definition of a stochastic machine

Definition 1 *A stochastic state machine SSM is defined as the 4-tuple* $SSM = (P, T, F, \Lambda)$ *where*

- P *is the non empty set of places*
- T *is the non empty set of transitions*
- $F \subseteq P \times T \cup T \times P$ *with* $dom(F) \cup codom(F) = P \cup T$ *is the flow relation. It has to satisfy the following restriction:* $\forall t \in T :\mid {}^\bullet t \mid = \mid t^\bullet \mid = 1$
- $\Lambda : T \rightarrow I\!R^+$, *where* $\Lambda(t)$ *is the rate of the exponential probability distribution associated to transition* t

With respect to SPN nets, SSM have the additional restriction of not allowing any type of synchronization (all transitions are of the 1-in 1-out type), although they allow non determinism.

We now define Superposed stochastic automata net.

Definition 2 *We call Superposed Stochastic Automata net the tuple* $N = (P, T, F, \Pi, \Lambda)$ *where*

- P *is the non empty set of places*
- T *is the non empty set of transitions*
- $F \subseteq P \times T \cup T \times P$ *with* $dom(F) \cup codom(F) = P \cup T$ *is the flow relation. It has to satisfy the following restriction:* $\forall t \in T :\mid {}^\bullet t \mid = \mid t^\bullet \mid \geq 1$
- $\Pi = \{\Pi_i\}$ *is a partition of* P *such that* $\forall t \in T$ *and* $\forall i \in 1, \ldots, m : |\Pi_i \cap {}^\bullet t| = |\Pi_i \cap t^\bullet| \leq 1$
- $\Lambda : T \rightarrow I\!R^+$, *where* $\Lambda(t)$ *is the rate of the exponential probability distribution associated to transition* t

The condition on the partition Π identifies a set of stochastic state machines that compose the SSA net. SSA are a subclass of SPN nets in which only a certain type of synchronization is allowed, in particular each transition must have the same number of input and output arcs.

SSA transitions that have $\mid {}^\bullet t \mid = \mid t^\bullet \mid > 1$ are called *synchronization (or synchronized) transitions*, as they represent synchronization between subnets.

Another way to interpret SSA nets is to say that they are obtained from a set of SSMs by synchronization on transitions of equal rate.

We can now define SSM and SSA *systems* which can be obtained from SSM and SSA nets by assigning an appropriate initial marking.

Definition 3 *We call stochastic state machine system the tuple* $R = (P, T, F, \Pi, \Lambda, p)$ *where*

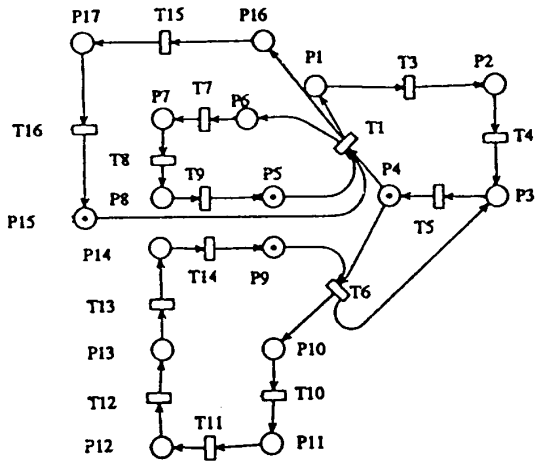
- (P, T, F, Π, Λ) *is a SSM net.*
- $p \in P$ *is the initial marking of the SSM*

The dynamics of SSMs is the same as that of SPNs. It is interesting to note that, due to the particular choice of the initial marking and to the requirement of the transitions to be 1-in 1-out, *exactly one place is marked in any reachable marking of an SSM*, so that we can specify the state of an SSM simply by the name of the marked place.

Definition 4 *We call Superposed Stochastic Automata system the tuple* $R = (P, T, F, \Pi, \Lambda, M_0)$ *where*

- $N = (P, T, F, \Pi, \Lambda)$ *is a SSA net*
- $M_0 \subseteq P$ *and* $\forall i \in [1, \ldots, m]\ |\Pi_i \cap M_0| = 1$

The requirement on the initial marking is such that *there is exactly one place marked in each element of the partition* in any reachable marking. Therefore an SSA system can be considered as the superposition on transitions of equal rate of a set of SSM systems.



$\Pi_0 = \{P1, P2, P3, P4\}$
$\Pi_1 = \{P5, P6, P7, P8\}$
$\Pi_2 = \{P9, P10, P11, P12, P13, P14\}$
$\Pi_3 = \{P15, P16, P17\}$

Figure 1: An SSA system

Figure 1 provides a pictorial representation of an SSA system where the subset of places is partitioned into four subsets. $T1$ is the synchronization transition of SSM systems identified by Π_0, Π_1, Π_3, and transition $T6$ is the synchronization between Π_0 and Π_2.

3 Infinitesimal generator of SSA systems

In this section we explain how to write the infinitesimal generator of an SSA system as Kronecker expression of the infinitesimal generators of the component automata.

It is well known in the literature [3] that there exists a strong relation between the matrix operator known as Kronecker sum and the composition of independent, continuous stochatic processes. In this section we exploit this property to write the infinitesimal generator of an SSA as the Kronecker expression of the infinitesimal generators of some independent processes plus some adequate matrices that play the role of "correction matrices.

3.1 Kronecker sum and product and the composition of stochastic processes

As far as this work is concerned we only consider matrix product and sum of matrices for square matrices (infinitesimal generator are square matrices by definition). Let $M(p)$ be the set of all $p \times p$ matrices of real values. We can then give the following definitions.

Definition 5 *Let $A \in M(n)$ and $B \in M(p)$; C is the Kronecker (tensor) product of A and B and we write $C = A \otimes B$ iff $C \in M(np)$ is defined by:*

$$C = \{c_{\bar{i}\bar{j}} : c_{\bar{i}\bar{j}} = a_{i_1 j_1} b_{i_2 j_2} \text{ with } \bar{i} = (i_1, i_2), \bar{j} = (j_1, j_2)\}$$

As a simple example consider the tensor product of two 2 x 2 matrices. We have

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$C = A \otimes B = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix}$$

If we interpret A and B as the state transition matrix of two discrete time Markov chains, it is immediate to recognize (see Davio in [3]) that C is the transition probabilities matrix of the process obtained as independent composition of the two original processes.

Let us now define the Kronecker (or tensor) sum of two matrices

Definition 6 *Let $A \in M(n)$ and $B \in M(p)$; D is the Kronecker (tensor) sum of A and B and we write $D = A \oplus B$ iff $D \in M(np)$ is defined by:*

$$D = A \oplus B = A \otimes Id_p + Id_n \otimes B$$

where Id_k is the $k \times k$ identity matrix.

Let's consider again the two matrices A and B of the example above: we can express their Kronecker (tensor) sum as follows:

$$A \otimes Id_2 = \begin{pmatrix} a_{11} & 0 & a_{12} & 0 \\ 0 & a_{11} & 0 & a_{12} \\ a_{21} & 0 & a_{22} & 0 \\ 0 & a_{21} & 0 & a_{22} \end{pmatrix}$$

$$Id_2 \otimes B = \begin{pmatrix} b_{11} & b_{12} & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 \\ 0 & 0 & b_{11} & b_{12} \\ 0 & 0 & b_{21} & b_{22} \end{pmatrix}$$

$$D = \begin{pmatrix} a_{11}+b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11}+b_{22} & 0 & a_{12} \\ a_{21} & 0 & a_{22}+b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{22}+b_{22} \end{pmatrix}$$

Again if we consider A and B as the infinitesimal generator of two, time continuous markovian processes, then D is the infinitesimal generator of the process obtained by independent composition of the two original one. It can be observed that in D all transition

rates among states that differ by more than a single component are set equal to zero, as it is the case when we are in a continuous-time environment.

From an applicative point of view we cannot take a big advantage from what has just been said, because usually the stochastic processes involved in an application are far from being independent.

Another aspect that deserves attention is that even if the infinitesimal generator of D is rewritten using only the infinitesimal generators of the components SSMs, nevertheless $D = A \oplus B$ has a number of elements equal to $|A| \times |B|$ and the same is true also for $C = A \otimes B$. The interesting point is that the solution of $\pi C = 0$, the characteristic equation of the Markov process defined by C, does not require to compute C, but the solution can be done in terms of A and B (see [3]). This is not particularly surprising since the solution of a stochastic process which is obtained as independent composition of two other processes can always be computed from the solution in isolation of the two processes.

3.2 Infinitesimal generator of an SSA system

What happens if we want to express the infinitesimal generator of a set of non independent stochastic processes in terms of the infinitesimal generator of the component processes? In [14] Plateau suggests (and applies) the idea of separating in each process the independent behaviour from the dependent one and to express the infinitesimal generator of the composed process as the sum of two parts: one is the composition of the independent behaviours of the component automata, and the other one takes into account all dependencies.

Let Σ be a SSA system whose associated partition Π individuates m SSM systems. Let Q_i be the infinitesimal generator of SSM_i and Q that of Σ. Our goal is to study the relationships among the single Q_i and Q. We shall therefore try to interpret the stochastic process associated to Σ as the composition, of course not independent, of the stochastic processes associates to the single SSM_i.

Before moving on to a more formal treatment of the problem, let us introduce a very simple example. Figure 2 depicts two simple SSMs, SSM_1 and SSM_2 that have a transition of equal rate μ. The SSA system Σ in Figure 3 is obtained from SSM_1 and SSM_2 by superposition on transition $t2$. The infinitesimal generators associated to SSM_1 and SSM_2 can be expressed as:

$$Q_1 = \begin{bmatrix} -\mu & \mu \\ \nu & -\nu \end{bmatrix} \quad Q_2 = \begin{bmatrix} -\mu & \mu \\ \lambda & -\lambda \end{bmatrix}$$

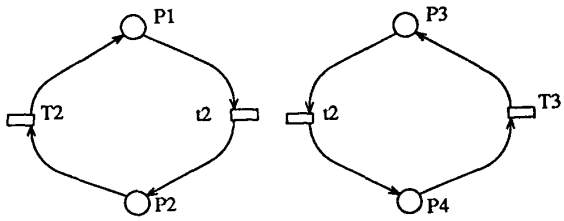


Figure 2: Component SSMs

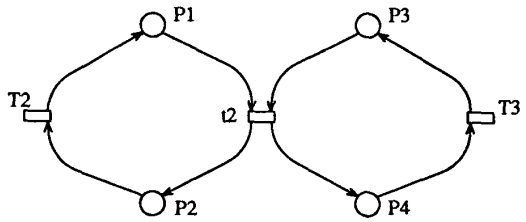


Figure 3: SSA system with only one synchronization transition and two SSMs

While the matrix Q of the SSA system is given by

$$Q = \begin{array}{c|cccc} & 1,3 & 1,4 & 2,3 & 2,4 \\ \hline 1,3 & -\mu & 0 & 0 & \mu \\ 1,4 & \lambda & -\lambda & 0 & 0 \\ 2,3 & \nu & 0 & -\nu & 0 \\ 2,4 & 0 & \nu & \lambda & -\lambda-\nu \end{array}$$

If we delete from Q_1 and Q_2 the elements that concern the non independent behaviour of the two SSMs, that is to say, if we delete from Q_1 and Q_2 the elements that represent the firing of the synchronized transition, we obtain the two following matrices Q_1' and Q_2'.

$$Q_1' = \begin{bmatrix} 0 & 0 \\ \nu & -\nu \end{bmatrix} \quad Q_2' = \begin{bmatrix} 0 & 0 \\ \lambda & \lambda \end{bmatrix}$$

The composition of the independent behaviour of SSM_1 and SSM_2 leads to

$$Q' = Q_1' \oplus Q_2' = \begin{array}{c|cccc} & 1,3 & 1,4 & 2,3 & 2,4 \\ \hline 1,3 & 0 & 0 & 0 & 0 \\ 1,4 & \lambda & -\lambda & 0 & 0 \\ 2,3 & \nu & 0 & -\nu & 0 \\ 2,4 & 0 & \nu & \lambda & -\lambda-\nu \end{array}$$

The resulting matrix Q' differs from Q only in the first row, because the first row corresponds to the state from which we can fire the synchronized transition.

We can rewrite Q as

$$Q = Q' + K$$

where K is an appropriate "correcting factor" (correcting matrices). If we want to express also the correcting factor as Kronecker expression of matrices which are related to the single components, then we can remark that

$$K = \mu \cdot \widehat{K} = \begin{array}{c|cccc} & 1,3 & 1,4 & 2,3 & 2,4 \\ \hline 1,3 & -\mu & 0 & 0 & \mu \\ 1,4 & 0 & 0 & 0 & 0 \\ 2,3 & 0 & 0 & 0 & 0 \\ 2,4 & 0 & 0 & 0 & 0 \end{array}$$

where $\widehat{K}$ can be rewritten as $-K_1 \otimes K_2 + K_3 \otimes K_4$. Please note the use of the Kronecker product, an operator which is usually associated to discrete processes. This is because the firing of a synchronized transition cause a change of state in all the SSMs involved in the synchronization. This type of situation is what was called "simoultaneous jump" in [15]. Finally, we can express K_1, K_2, K_3, K_4 as

$$K_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad K_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$K_3 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad K_4 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

The product $K_1 \otimes K_2$ is the correcting factor for the diagonal elements, while $K_3 \otimes K_4$ inserts the effects of the synchronized transition.

Synchronization of two automata on a single transition. Let SSM_1 and SSM_2 be two SSMs with respectively n and m states and infinitesimal generators Q_1 (a $n \times n$ matrix) and Q_2 (a $m \times m$ matrix). Let Q (a $(n \cdot m) \times (n \cdot m)$ matrix) be the infinitesimal generator of the SSA obtained from SSM_1 and SSM_2 by superposition on the transition t of rate μ. Let us call i_1 e j_1 (i_2 e j_2) the two states of SSM_1 (SSM_2) such that $i_1[t > j_1$ ($i_2[t > j_2$), where we use the usual notation $i[t > j$ to say that t fires in i and its firing produces the new state j. We can build from Q_1 the matrix Q_1' where all state transitions out of state i_1 due to t are inhibited (please note that this would require a modification also in the diagonal elements). We can write

$$Q_1' = Q_1 - \mu E_{i_1 j_1}$$

where $E_{i_1 j_1}$ is the $n \times n$ matrix defined by

$$E_{i_1 j_1}(l,k) = \begin{cases} 1 & \text{if } l = i_1 \text{ and } k = j_1 \\ -1 & \text{if } l = k = i_1 \\ 0 & \text{otherwise} \end{cases}$$

We can build also Q_2 in a similar way. $Q_1' \oplus Q_2'$ is the independent part of the expression. We now have to determine the correcting factor that allows the firing of t when SSM_1 is in state i_1 and SSM_2 is in state i_2. We have therefore to add to $Q_1' \oplus Q_2'$ a $(n \cdot m) \times (n \cdot m)$ matrix where the only non null values are

$$-\mu((i_1,i_2),(i_1,i_2)) = \mu((i_1,i_2),(j_1,j_2)) = \mu$$

Hence $Q = Q_1 \oplus Q_2 - \mu(K_1 \otimes K_2) + \mu(K_3 \otimes K_4)$, where K_1 is the $n \times n$ matrix where the only non null element is that of index (i_1, i_1), K_3 is the same but for the states (i_1, j_1), while K_2 and K_4 are $m \times m$ matrices where the only non null elements are (i_2, i_2) and (i_2, j_2).

Synchronization of N automata on a single transition. In this case there are N SSMs, $SSM_i, i = 0, \ldots, N-1$ of associated matrices Q_i. Let us indicate with (h_k, j_k) the two states of SSM_i that have $h_k[t > j_k)$, when SSM_i is considered in isolation. If m_i is the set of places (and therefore of states) of SSM_i, then Q_i is a $m_i \times m_i$ matrix . From Q_i we can build $Q_i' = Q_i - \mu \cdot E_{h_i,j_i}$ and [1] $Q' = \Pi_{\oplus} Q_i'$ which differ from Q only in the firing of transition t, which is enabled in state $(h_0, h_2, \ldots, h_{N-1})$ with $(h_0, h_2, \ldots, h_{N-1})[t > (j_0, j_2, \ldots, j_{N-1})$. This addition requires an adjustment in the diagonal element $((h_0, h_2, \ldots, h_{N-1}),(h_0, h_2, \ldots, h_{N-1}))$. Finally

$$Q = \Pi_{\oplus_{i=0}^{N-1}} Q_i' - \mu \Pi_{\otimes_{i=0}^{N-1}} K_i + \mu \Pi_{\otimes_{i=0}^{N-1}} K_i'$$

where K_i is the matrix identically equal to zero, but for the element $K_i(h_i, h_i) = 1$, while K_i' is the matrix whose only non zero element is $K_i'(h_i, j_i) = 1$. Note that $E_{h_i,j_i} = K_i' - K_i$.

It is now clear that, for a given synchronization transition, it is necessary to add two correcting factors, and each correcting factor is the Kronecker product of N matrices.

Synchronization of N automata on two transitions. We now consider the case of N automata SSM_i that are synchronized on two transitions t_1 and t_2 of associated rates μ_1 and μ_2.

We indicate with $h_i[t_1 > h_i'$ the state transition caused by t_1 in SSM_i, if any, and $j_i[t_2 > j_i'$ is the state transition caused by t_2 in SSM_i, if any.

We can group the N automata as follows:

G_1: automata that synchronize only on t_1

G_2: automata that synchronize only on t_2

G_3: automata that synchronize on both t_1 e t_2

[1] We indicate with $\Pi_{\oplus_{i=0}^{N-1}} Q_i$ the successive N applications of operator $\oplus$.

The three groups have empty intersections.

Following the usual approach, we build Q'_i from Q_i as follows, depending on what is the group of SSM_i.

$$\begin{aligned} SSM_i \in \mathrm{G}_1 &\Rightarrow Q'_i = Q_i - \mu_1 E_{h_i,h'_i} \\ SSM_i \in \mathrm{G}_2 &\Rightarrow Q'_i = Q_i - \mu_2 E_{j_i,j'_i} \\ SSM_i \in \mathrm{G}_3 &\Rightarrow Q'_i = Q_i - \mu_1 E_{h_i,h'_i} - \mu_2 E_{j_i,j'_i} \end{aligned}$$

We can rewrite the above expression as $Q'_i = Q_i - \mu_1 E_{(\ {}^\bullet t_1, t_1^\bullet)_i} - \mu_2 E_{(\ {}^\bullet t_2, t_2^\bullet)_i}$ where $(\ {}^\bullet t_1, t_1^\bullet)_i = (\ {}^\bullet t_1 \cap \mathrm{G}_i, t_1^\bullet \cap \mathrm{G}_i)$. The correcting factors are of the type:

$$\begin{aligned} Q \;=\; & \Pi_{\oplus_{i=0}^{N-1}} Q'_i \\ & -\mu_1 \Pi_{\otimes_{i=0}^{N-1}} K_i(t_1) - \mu_2 \Pi_{\otimes_{i=0}^{N-1}} K_i(t_2) \\ & +\mu_1 \Pi_{\otimes_{i=0}^{N-1}} K'_i(t_1) + \mu_2 \Pi_{\otimes_{i=0}^{N-1}} K'_i(t_2) \end{aligned}$$

where

$K_i(t_1)$ is a matrix of all zeros, but for element $(\ {}^\bullet t_1 \cap SSM_i,\ {}^\bullet t_1 \cap SSM_i)$ which is equal to one. If ${}^\bullet t_1 \cap SSM_i = \emptyset$ then K_i is the identity matrix.

$K_i(t_2)$ is the same as above, but with $(\ {}^\bullet t_2 \cap SSM_i,\ {}^\bullet t_2 \cap SSM_i)$.

$K'_i(t_1)$ is a matrix of all zeros, but for element $(\ {}^\bullet t_1 \cap SSM_i, t_1^\bullet \cap SSM_i)$, which is equal to one. If ${}^\bullet t_1 \cap SSM_i = \emptyset$ then K'_i is the identity matrix.

$K'_i(t_1)$ is the same as above, but with $(\ {}^\bullet t_2 \cap SSM_i, t_2^\bullet \cap SSM_i)$.

Identities matrices have been introduced to take into account those automata that do not have the synchronization.

Synchronization of N automata on an arbitrary number of transitions. We are now ready to deal with the most general case of N SSMs systems that are synchronized on an arbitrary set of transitions. Let $TS \subseteq T$ be the set of synchronization transitions. A generic transition t is a synchronization transition iff $|\ {}^\bullet t| > 1$, i.e. $TS = \{t \in T : |\ {}^\bullet t| > 1\}$ (we remind the reader that, by definition of SSA, $|\ {}^\bullet t| = |t^\bullet|$). We say that an automata partecipates to the t synchronization iff $|\ {}^\bullet t \cap SSM_i| = 1$.

We have therefore the following formula that, given an SSA systems, computes its infinitesimal generator Q as Kronecker expressions of matrices of the $m_i \times m_i$ type, where m_i is the number of states of SSM_i (that also coincides with the number of places of the SSM).

$$\begin{aligned} Q \;=\; & \Pi_{\oplus_{i=0}^{N-1}} Q'_i \\ & - \sum_{t \in TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K_i(t) \\ & + \sum_{t \in TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K'_i(t) \end{aligned} \qquad (1)$$

where $Q'_i = Q_i - \sum_{t \in TS} \mu_t E_{(\ {}^\bullet t, t^\bullet)}$. The correcting factors K and K' are defined by

$$K_i(t) = \begin{cases} \text{identity} & \text{if } \{\ {}^\bullet t \cap SSM_i\} = \Phi \\ \text{all zeros but} & \\ (\ {}^\bullet t \cap S_i,\ {}^\bullet t \cap S_i) & \text{if } |\ {}^\bullet t \cap S_i| = 1 \end{cases}$$

$$K'_i(t) = \begin{cases} \text{identity} & \text{if } \{\ {}^\bullet t \cap S_i\} = \Phi \\ \text{all zeros but} & \\ (\ {}^\bullet t \cap S_i, t^\bullet \cap S_i) & \text{if } |\ {}^\bullet t \cap S_i| = 1 \end{cases}$$

Where we have used S_i for SSM_i for typesetting reasons. We have finally obtained an expression for the infinitesimal generator of an SSA system in terms of Kronecker sum and product of simple matrices. In particular if the SSA system under study is composed of N automata and TS synchronization transitions, then the number of matrices involved in the expression is $N + 2N|TS|$.

It is important to observe that the expression of Q is only a more compact way of writing the infinitesimal generator of an SSA system, but actually the number of elements of Q is still $Nm \times Nm$.

We have mentioned in the previous section that the solution of an equation of the type $\pi C = 0$ where C is the Kronecker product of a certain number of smaller matrices, does not require to compute C. This doesn't hold true any longer, at least it is not so immediate. In the next section we shall deal with the problem of finding a solution of $\pi Q = 0$ that does not require the computation and the storage of Q.

4 Solution of equation $\pi Q = 0$

In the previous section we have derived an expression for the infinitesimal generator of an SSA system. If the SSA system is composed of N SSMs, and if each SSM has m states, then the expression derived involves matrices of the $(m \times m)$ type, and many of them are very sparse. If we compute the expression of Q, then the resulting matrix is of size $(Nm \times Nm)$. In order to take advantage of the particular structure of the expression it is necessary to find an appropriate solution method that does not require to develop the Kronecker products and sums

The solution of the characteristic equation of the Markov process $\pi Q = 0$ is typically done numerically, and there are two basic techniques: *direct* method and *iterative* method. If we want to work on the expression without computing Q, then we cannot apply direct methods, that typically requires to work with the complete infinitesimal generator. We have hence to use an iterative method, like, for example, the power methods, that is the one that is actually used in our implementation. At each iteration we compute the n-th approximation of the probability vector π as:

$$\pi^n = \pi^{n-1} + \frac{1}{\omega(1+\varepsilon)} \pi^{n-1} Q$$

where ω is the maximum of the diagonal elements and ε is an arbitrarily small constant.

In order for the power method to converge we need to have a single unitary eigenvalue and a sufficient condition for a stochastic matrix to have a unique unitary eigenvalue is that the matrix is irreducible. This condition corresponds to the requirement that the Markov process be ergodic. But *this is not the case for the matrix expressed by Equation 1*, even if the process associated to the SSA is actually ergodic.

Observe infact that we consider as possible states of an SSA system all possible combinations of the states of the single SSMs, but not all states are possible. For example consider the SSA system of Figure 1. All states that are obtained as combinations of the states of the single automata are actually reachable from the initial marking of the system. If we now modify the SSA in Figure 1 so as to superpose transition $T15$ with transition $T7$, then all states where place $P6$ is marked and $P16$ is not are not reachable any longer from the initial marking.

The key observation that allows to use the power method even in this case is that, starting from an initial value of π_0 in which we assign probability 1 to the state that corrisponds to the initial marking, and probability 0 to all other states, then by computing at each iteration the formula $\pi^n = \pi^{n-1} + \frac{1}{\omega(1+\varepsilon)}\pi^{n-1}Q$, we never put to non zero the probability of states which are not reachable from the initial marking. This is because the matrix Q contains only the rate transitions among reachable states.

We also have to note that the choice of the initial value of π influence the speed of convergence of the method in a negative sense (the mass probability is all accumulated in a single point and it may require a great number of iterations to distribute it among all reachable states). Better methods are actually under study to provide a more convenient initialization for the probability vector. As an example consider the case where each SSM is synchronized with only another SSM: in this case all states that are obtained as combination of the local states are actually reachable, and we can therefore assign equal probability to all states.

We now show how the expression for Q can be rewritten in order to allow the computation of the iterative equation also without explicitly computing Q. This part is mainly a rewriting in our context of the results obtained by Davio in [3] which have been applied to the composition of non independent stochastic processes by Plateau in [14] and [13].

The iterative equation with Q substituted by its expression has the form:

$$\pi^k = \pi^{k-1} + \frac{1}{\omega(1+\varepsilon)}\pi^{k-1}(\Pi_{\oplus_{i=0}^{N-1}} Q_i' + \quad (2)$$

$$- \sum_{t\epsilon TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K_i(t) + \sum_{t\epsilon TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K_i'(t))$$

The Kronecker sum of Equation (2) can be rewritten as ordinary matrix sum of Kronecker products. As we shall see next, Kronecker products are easier to deal with. We obtain:

$$\begin{aligned}\Pi_{\oplus_{i=0}^{N-1}} Q_i' = {} & Q_0' \otimes Id_1 \otimes \cdots \otimes Id_{N-1} + \\ & Id_0 \otimes Q_1' \otimes Id_2 \otimes \cdots \otimes Id_{N-1} + \ldots \\ & Id_0 \otimes Id_1 \otimes Q_2' \otimes Id_3 \cdots \otimes Id_{N-1} + \\ & Id_0 \otimes \cdots \otimes Id_{N-2} \otimes Q_{N-1}'\end{aligned}$$

That is to say a sum with N elements. Each element is the Kronecker product of N matrices and $N-1$ of these matrices are identity matrices.

Equation (2) can therefore be rewritten as

$$\begin{aligned}\pi^k = {} & \pi^{k-1} + \tfrac{1}{\omega(1+\varepsilon)}\pi^{k-1}(\textstyle\sum_{i=0}^{k-1} Id_{p_0} \otimes \cdots \otimes \\ & Id_{p_{i-1}} \otimes Q_i' \otimes Id_{p_{i+1}} \cdots \otimes Id_{p_{k-1}} + \\ & - \textstyle\sum_{t\epsilon TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K_i(t) + \\ & \textstyle\sum_{t\epsilon TS} \mu_t \Pi_{\otimes_{i=0}^{N-1}} K_i'(t))\end{aligned} \quad (3)$$

In [3] it is shown that it is possible to rewrite the Kronecker product of N matrices M_i each of the type $(p_i \times p_i), 0 \leq i < N$ as follows

$$\begin{aligned}\otimes_{i=0}^{N-1} M_i = {} & \Pi_{i=0}^{N-1} Id_{p_0} \otimes \cdots \otimes Id_{p_{i-1}} \otimes \\ & M_i \otimes Id_{p_{i+1}} \cdots \otimes Id_{p_{N-1}} \\ = {} & \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes M_i)\end{aligned}$$

where $(q_i = \Pi_{j=0, j\neq i}^{N-1} p_j)$ and S_{p_i,q_i} is the matrix that realizes a perfect shuffle permutation of the (p_i, q_i) type. Equation (3) can hence be transformed into

$$\begin{aligned}\pi^k = {} & \pi^{k-1} + \tfrac{1}{\omega(1+\varepsilon)}\pi^{k-1}(\textstyle\sum_{j=0}^{N-1} \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes A_{ij}) \\ & - \textstyle\sum_{t\epsilon TS} \mu_t \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes K_i(t)) + \\ & + \textstyle\sum_{t\epsilon TS} \mu_t \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes K_i'(t)))\end{aligned} \quad (4)$$

where

$$A_{ij} = \begin{cases} Q_i & \text{if } i = j \\ Id_{p_j} & \text{otherwise} \end{cases}$$

By distributing π^{k-1} over the terms of the sum we obtain

$$\begin{aligned}\pi^k = {} & \pi^{k-1} + \tfrac{1}{\omega(1+\varepsilon)}(\textstyle\sum_{j=0}^{N-1} \pi^{k-1} \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes A_{ij}) \\ & - \textstyle\sum_{t\epsilon TS} \mu_t(\pi^{k-1} \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes K_i(t))) + \\ & + \textstyle\sum_{t\epsilon TS} \mu_t(\pi^{k-1} \Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes K_i'(t))))\end{aligned} \quad (5)$$

By taking advantages of the following equalities:

$$\begin{aligned}&\Pi_{i=0}^{N-1} S_{p_i,q_i}(Id_{q_i} \otimes A_i) = \\ &\quad \Pi_{i=0}^{N-1} S_{p_0 \cdots p_i, p_{i+1} \cdots p_{N-1}}(Id_{q_i} \otimes A_i) S_{p_{i+1} \cdots p_{N-1}, p_0 \cdots p_i}\end{aligned}$$

and

$$S_{p_{i+1}\cdots p_{N-1},p_0\cdots p_i} = S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}}$$

where S^T indicates the inverse of the permutation realized by S. We finally get to

$$\pi^k = \pi^{k-1} + \frac{1}{\omega(1+\varepsilon)}\Big(\sum_{j=0}^{N-1}(\pi^{k-1} \tag{6}$$

$$\Pi_{i=0}^{N-1} S_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} (Id_{q_j}\otimes A_{ij}) S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}}) +$$

$$-\sum_{t\epsilon TS}\mu_t(\pi^{k-1}\Pi_{i=0}^{N-1} S_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} (Id_{q_i}\otimes K_i(t)) S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}}) +$$

$$+\sum_{t\epsilon TS}\mu_t(\pi^{k-1}\Pi_{i=0}^{N-1} S_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} (Id_{q_i}\otimes K'_i(t)) S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}})) \tag{7}$$

Where the matrix multiplication by $S^T_{p,q}$ is at no cost if indices are appropriately stored. The most costly operation is the computation of an expression like that in the first term of the sum:

$$\pi^{k-1}\Pi_{j=0}^{N-1} S_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} (Id_{q_j}\otimes A_{ij}) S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} \tag{8}$$

To efficiently compute all these products we can take advantage of the left associative law of the ordinary matrix multiplication, and we can therefore compute the following sequence of expressions:

$$\begin{aligned}
\pi_0^k &= (((\pi^{k-1} S_{p_0,p_1\cdots p_{N-1}})(Id_{q_0}\otimes A_0)) S^T_{p_0,p_1\cdots p_{N-1}}) \\
\pi_1^k &= (((\pi_0^k S_{p_0p_1,p_2\cdots p_{N-1}})(Id_{q_1}\otimes A_1)) S^T_{p_0p_1,p_2\cdots p_{N-1}}) \\
&\vdots \\
\pi_{N-1}^k &= (((\pi_{N-2}^k S_{p_0\cdots p_{N-2},p_{N-1}})(Id_{q_{N-1}}\otimes A_{N-1}) \\
&\qquad S_{p_0\cdots p_{N-2},p_{N-1}}
\end{aligned}$$

and the computation of π_{N-1}^k gives the final value for (8).

By taking advantage of the special properties of the Kronecker product we can therefore solve the characteristic equation of an SSA system by simply using a vector of $\Pi_{j=0}^{N-1} p_i$ elements (number of (potential) states of the SSA), and matrices of size $(p_i \times p_i)$ $(0 \le i < N)$ without being forced to explicitly compute and store the infinitesimal generator of the SSA, which has a (potential) size of $(\Pi_{j=0}^{N-1} p_i \times \Pi_{j=0}^{N-1} p_i)$.

We have therefore found a way of computing each iterative step in such a way as to save storage. But what is the computational complexity of the proposed procedure? The complexity of the computation of Equation 7 is of the order

$$N^2\alpha + 2|TS|N\alpha = O(N^2\alpha)$$

where

$$\alpha = (\Pi_{i=0}^{N-1} p_i)\Big(\sum_{i=0}^{N-1} p_i\Big) \tag{9}$$

and the term α expresses the complexity of the inner computation of Equation 7. Infact in the product $PS_{\sigma_i}(Id_{q_i}\otimes A_i)S^T_{\sigma_i}$ the cost due to permutations is of the order $\Pi_{i=0}^{N-1} p_i$ and to execute the product $P_{\sigma_i}(Id_{q_i}\otimes A_i)$, where $P_{\sigma_i} = PS_{\sigma_i}$ it is necessary to execute q_i ($q_i = \Pi_{j=0,j\neq i}^{N-1} p_i$) products of order p_i.

5 Parallel solution

The expression 7 lends itself to parallel evaluation. We have developed a parallel solution algorithm that has been implemented on a transputers machine (a *MEIKO Computing Surface*) under the CS_Tool programming environment.

We can observe that the three summations in Equation 7 are independent: their evaluation is not influenced by the order, moreover they work on different data (matrices). The independence of data is an important parameter that we have to consider, because in a transputers based machine there is no shared memory: if we partition data among processes (and consequently among transputers) we can avoid data duplications.

A second observation is that each addendum of the three summations are independent (for the same above mentioned reasons) each one from the others.

It's important to observe that the parallel architecture and the tool which have been used have influenced the choices that we have taken during the algorithm development. In particular every transputer has four links to connect it to the others processing elements and to avoid rerouting (made automatically by the CS_Tool Run Time Environment) a good rule is to give to the set of transputers the structure of a ternary tree.

Using CS_Tool, only a transputer of the transputers set allocated to the parallel application can directly access the file system. This observation suggest to limit file system accesses to one of the sequential processes that compose the parallel program.

In the parallel program we have three types of processes. Processes of type A compute (at each step of the iterative algorithm) a matrix product of the following form

$$\pi^{k-1}\Pi_{i=0}^{N-1} S_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}} (Id_{q_i}\otimes M) S^T_{p_0\cdots p_i,p_{i+1}\cdots p_{N-1}}$$

where M is a $A_{i,j}$ matrix or a correction matrix. The number of these processes is $2|TS| + N$. Processes of type B act instead as results collectors: they receive partial values of π^k from first type processes and add them to obtain the value of the relative summation in the k-th step. Finally, there is a single process of a

third type that coordinates the iterative algorithm. At every step, it computes the steady state vector from the partial steady state vectors that it receives from the process of type B; it checks the algorithm convergency to end the parallel program. This process is the only one that access the file system.

At each step, the computation starts when processes receive the steady state vector obtained in the previous step by the "father" process. The steady state vector is computed from the process of the third type.

The parallel algorithm computes each summation term in parallel so that its complexity is of the order of $O(N\alpha)$ and therefore the expected speed up is $\approx N$, we therefore expect a speed-up which is proportional to the number of SSMs that compose the SSA system.

Table 1 shows a comparison among the execution time of sequential implementation on a single transputer and its parallel conterpart on Tr transputers. Each SSA system analyzed is described in terms of the number of places $|P|$, of transitions $|T|$, of synchronized transitions $|TS|$, and of number of states $|RS|$.

We provide here only results of parallel versus sequential implementation of our solution, while it should also be necessary to compare these results from that coming from the standard SPN packages. Unfortunately, the SPN package that is available here, which is GreatSPN [2] runs on a SUN workstation and therefore the comparison is unfeasible, both because it is difficult to estimate the relative speed of the two machines and because it is difficult to interpret the execution time information available under the SUN operating system. We plan to re-implement the solution part of the GreatSPN package under our transputer machine, so as to be able to test also the speed of the sequential solution proposed with respect to more classical techniques.

6 Conclusions

In this paper we have presented the application of a new solution method introduced by Plateau in [14] that exploits Kronecker algebra as nicely described in [3] to compute the steady state solution of SSA, a subclass of SPN. The use of this rather restricted class was justified by the need of keeping all formulas as simple as possible in order to arrive in a reasonable amount of time to have also a parallel implementation of the solution method.

Nevertheless, although SSA appear to be as rather restrictive on the kind of systems that can be modelled, they have a rather wide range of applicability. In particular the untimed counterpart of SSA, the class of net known as Superposed Automata (see [4]) from which SSA are derived, has been used to model distributed systems, especially in the office organization environment [5].

The extension to Superposed Stochastic Petri Nets (SSPN) doesn't appear to bring any major theoretical problem. SSPN can be defined as the class of nets obtained by a set of SPN systems that interact through transition(s) synchronization. It is therefore a very powerful model, perfectly suitable to describe systems that interect by message exchanges implemented in a rendez-vous style and other.

The automatic generation of the Kronecker expression of an SSPN doesn't appear to pose major problems. The main difference is that while in SSA a transition can be enabled in a single state, so that we have one correcting factor per synchronized transition, in SSPN a synchronized transition can be enabled in more then one state (and typically this is the case) so that we have to provide more correcting factors. It is clear that if there are a large number of correcting factors then part of, or all, the advantages introduced by the solution method proposed can be lost, although the advantages of the parallel implementation are still retained. It should also be noted that this is true also for SSA: if SSMs are very dependent one from the others, i.e., if they performs a lot of synchronizations, then the advantages of the method can be lost. The point here is that with SSPN it can be easier to describe systems that, even if they interact "graphically" on a single transition, the level of interactions among stochastic processes can be very high, if there are many different markings in which that transition is enabled.

We plan in the near future to extend our parallel implementation also to SSPN and to apply SSPN to the modelling of parallel CSP-like programs, as described in [7]. A more long term goal would be to extend the class of nets to the kind of hierarchical interactions described in [1].

Acknowledgements

This work is part of the author's phD dissertation [6] which highly profited from the supervision of Prof. Gianfranco Balbo. The parallel implementation was done in collaboration with Valerio Malenchino, whose contribution is highly appreciated. Finally, the author is grateful to the anonymous referees for their detailed comments on the draft of this paper.

References

[1] P. Buchholz. Numerical solution methods based on structured descriptions of Markovian models. In *Proc. 5th Int. Conf. Modeling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy, February 1991.

[2] Giovanni Chiola. *GreatSPN* 1.5 software architecture. In *Proc. 5th Int. Conf. Modeling Techniques*

SSM	$\|P\|$	$\|T\|$	$\|TS\|$	$\|RS\|$	Seq.	Paral.	Tr.	Speed Up
2	4	3	1	4	0.002064	0.002080	8	0.992307
2	14	13	1	48	0.027393	0.012503	8	2.190914
3	6	4	3	8	0.006991	0.002502	13	2.794164
4	11	7	4	17	0.052246	0.010891	16	4.797171
3	7	4	2	6	0.008830	0.003476	11	2.540276
4	17	15	2	288	0.262457	0.069659	12	3.767739
4	26	30	3	287	2.773557	0.508684	13	5.452416
4	28	28	6	228	5.806063	1.142755	13	5.080759

Table 1: Parallel versus sequential implementation

and Tools for Computer Performance Evaluation, Torino, Italy, February 1991.

[3] M. Davio. Kronacker products and shuffle algebra. *IEEE Transactions on Computers*, 30(2):1099–1109, 1981.

[4] F. De Cindio, G. De Michelis, L. Pomello, and C. Simone. Superposed automata nets. In C. Girault and W. Reisig, editors, *Application and Theory of Petri Nets*. IFB 52, New York and London, 1982.

[5] F. De Cindio, G. De Michelis, and C. Simone. GAMERU, a language for the analysis and design of human communication pragmatics within organizational systems. In G. Rozenberg, editor, *Advances on Petri Nets '87*, number 266 in LNCS. Springer Verlag, 1987.

[6] S. Donatelli. *L'uso delle reti di Petri per la valutazione e la validazione di sistemi di grandi dimensioni.* PhD thesis, Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149, Torino, Italy, February 1990. (in italian).

[7] S. Donatelli, G. Franceschinis, A. Mazzeo, N. Mazzocca, and M. Ribaudo. Characterizing parallel programs behaviour. Technical report, 1991. (submitted for publication).

[8] G. Florin and S. Natkin. Matrix product form solution for closed synchronized queueing networks. In *Proc. 3rd Intern. Workshop on Petri Nets and Performance Models*, pages 29–39, Kyoto, Japan, December 1989. IEEE-CS Press.

[9] W. Henderson and P.G. Taylor. Aggregation methods in exact performance analysis of stochastic Petri nets. In *Proc. 3rd Intern. Workshop on Petri Nets and Performance Models*, pages 12–18, Kyoto, Japan, December 1989. IEEE-CS Press.

[10] A.A. Lazar and T.G. Robertazzi. Markovian Petri net protocols with product form solution. In *Proc. of IEEE INFOCOM'87*, San Francisco, CA, USA, March 1987. Also in *Performance Evaluation*, Vol.12, 1991, pp.67–77.

[11] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transaction on Computers*, 31(9):913–917, September 1982.

[12] M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models.* PhD thesis, UCLA, Los Angeles, CA, 1981. Ph.D. Thesis.

[13] B. Plateau. *Repartition, parallelisme et des elements de leur valutation.* PhD thesis, (in french), Paris, France, 1981.

[14] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. 1985 SIGMETRICS Conference*, pages 147–154, Austin, TX, USA, August 1985. ACM.

[15] B. Plateau. Peps: a package for solving complex Markov models of parallel systems. In R. Puigjaner and D. Poiter, editors, *Modeling techniques and tools for computer performance evaluation*, pages 291–306. Plenum Press, New York and London, 1990.