UNE IMP
EHE CWI
★QMIPS★
INR UER
ZAR UTO

Quantitative
Modeling
In
Parallel
Systems

# Some Algebraic Considerations for Efficient Computations in Timed Petri Nets

Bruno Gaujal
INRIA Sophia-Antipolis
B.P. 93, 06902 Sophia Antipolis Cedex, France

D W2.T3.16b-2.v1

19 September 1995



**Distribution Level:** All Partners

**Approved by:** F. Baccelli

**Abstract** In this paper we show how structural properties of some classes of Timed Petri nets can be used to derive efficient algorithms to compute important quantities such as total number of firings, average number of tokens in places or stability conditions. This paper will appear in the proceedings of HICSS-29, Hawaii, Jan 96.
**Key words** Petri Nets, Parallel Simulation, Fixed Points, Fluid Models, Stability.

**The QMIPS Project:** The QMIPS project is a collaborative research project supported by the CEC as ESPRIT-BRA project no 7269. It is being carried out by the following organisations: CWI (Amsterdam), EHEI (University of Paris V), Imperial College (London), INRIA (Sophia-Antipolis), University of Erlangen, University of Newcastle, University of Torino and University of Zaragoza.

# 1 Introduction

The study of large discrete dynamic systems depends heavily on the model one uses. Several models have been introduced bringing along different techniques for the analysis of discrete systems. Queuing models have been studied since [15]. These models has yield a lot of results, however they are not really adapted for the study of very large systems. More recently, several other models have been proposed such as generalized semi-Markov processes ([11]) or Timed Petri nets ([18, 3]). These two models have more structure than classical queuing models and have been used to derive qualitative properties (like monotonicity or stability conditions) as well as quantitative results (like computations of the statistics of the system).

In this paper we focus on timed Petri nets. We show how structural considerations introduced in [4] where evolution equations of a Petri net are introduced and used to establish results about total number of events in system and stability conditions. For the class of *single input free choice nets* (introduced in [4]) we use these results to design effective computations. Therefore, our main interest in this paper is not to provide a multi purpose simulation algorithm (as in [8]) or to provide all kinds of statistics on the system under study but rather to focus on a specific class of systems (already quite interesting on a modeling point of view) and provide only some specific results such as

- the total number of firings,
- the average number of firings of the transitions,
- the verification of the stability of the system,

using new and fast parallel algorithms.

In the second section, we present more precisely the model we will be studying in the following, which is a class of free choice Petri nets. In Section 3, we build the evolution equations of the net. In Section 4, we use these equations to derive an efficient parallel algorithm to compute the total number of events taking place in the system. In Section 5 we construct a fluid model of a Petri net which is adapted for the computation of the average number of tokens in places. Section 6 provides an optimal representation of the system to solve the questions addressed in Section 5. Finally, we expose in Section 7 a way to compute the stability condition of a free choice net and Section 8 shows the results of simulations carried out under the PVM environment.

# 2 Definitions

A Petri net is a 4-tuple $(\mathcal{P}, \mathcal{T}, \mathcal{C}, \mathcal{M}_0)$ where $\mathcal{P}$ is the set of places of size $P$, $\mathcal{T}$ is the set of transitions of size $T$, $\mathcal{C}$ the set of arcs between places and transitions or between transitions and places ($\mathcal{C}$ is a subset of $\mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$). $\mathcal{M}_0$ is the initial marking in the places. We denote by ${}^\bullet t$ the set $\{p \in \mathcal{P} : (p,t) \in \mathcal{C}\}$ (i.e. the set of all input places of $t$). We define similarly the sets $t^\bullet$, ${}^\bullet p$, $p^\bullet$ as the set of output places of $t$, the set of input transitions of $p$ and the set of output transitions of $p$, respectively.

A *timed* Petri net is a Petri net with temporal data attached to transitions : $\Sigma = \{\sigma_t(n), t \in T, n \in \mathbb{N}\}$ where $\sigma_t(n)$ gives the duration of the $n$-th firing of transition $t$. If transition $t$ begins to fire for the $n$-th time at epoch $e$, this firing will end at epoch $e + \sigma_t(n)$; tokens are then taken out of input places and put in output places of $t$ according to the firing rule of the un-timed Petri net. For more details on this, see [19].

In the following, we will be interested in some subclasses of Petri nets.

**Definition 2.1 (marked graph)** *A marked graph (MG) is a Petri net $(\mathcal{P}, \mathcal{T}, \mathcal{C}, \mathcal{M}_0)$ such that $\forall p \in \mathcal{P}, |p^\bullet| = |{}^\bullet p| = 1$.*

**Definition 2.2 (State Machine)** *A State machine (SM) is a Petri Net $(\mathcal{P}, \mathcal{T}, \mathcal{C}, \mathcal{M}_0)$ such that $\forall t \in \mathcal{T}, |t^\bullet| = |{}^\bullet t| = 1$.*

**Definition 2.3 (free choice net)** *A free choice net (FCN) is a Petri net $(\mathcal{P}, \mathcal{T}, \mathcal{C}, \mathcal{M}_0)$ such that $\forall p \in \mathcal{P}, t_1, t_2 \in p^\bullet, t_1 \neq t_2 \Rightarrow {}^\bullet t_1 = {}^\bullet t_2 = \{p\}$.*

Note that the set of FCN contains both the sets of MG and SM.

## 2.1 Routing

When a place $p$ has several output transitions, the presence of a token in $p$ may create a conflict between its output transitions and one has to choose a policy to solve this conflict.

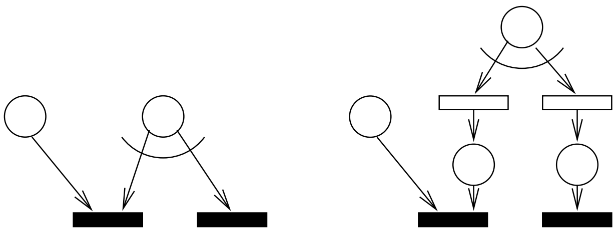
The most common policy used in the Petri net literature to solve the conflicts is the *race policy*. However we will be using the *routing policy* in the rest of the paper, which was introduced in [2].

For the race policy, the resolution of the conflict is based on the firing durations of the transitions : as soon as a token enters place $p$, all enabled transition potentially start their firing. The first transition to end actually fires and removes one token in each input place (including $p$). All the other transitions have their firing aborted and resume their former state. This policy has the advantage to preserve the Markovian behavior of the system when firings are exponentially distributed. However, it does not preserve qualitative properties of the system such as monotonicity (see [2]).

The routing policy is different. It solves the conflict by imposing a route to each token entering place $p$, regardless of the state of the system. Therefore a token may be routed to a transition which is not enabled while other transitions could have fired. In this case, the token has to wait for the enabling of its transition before being fired. More precisely, Place $p$ has a routing sequence $\nu^p : \mathbb{N} \rightarrow p^\bullet$, where $\nu^p(n)$ gives the transition $t \in p^\bullet$ to which the $n$-th token to enter place $p$ is routed to.

## 2.2 Modeling Questions

It is not difficult to see that any timed Petri Net with a routing policy can be transformed into an equivalent routed free choice net. This transformation is illustrated in Figure 1. Since routing places have a different semantic than usual in Petri nets, they are depicted in the figures by a place with a "fan" on its output arcs.



Routed Petri Net Equivalent free choice net

Figure 1: Transformation of a routing place into a free choice place.

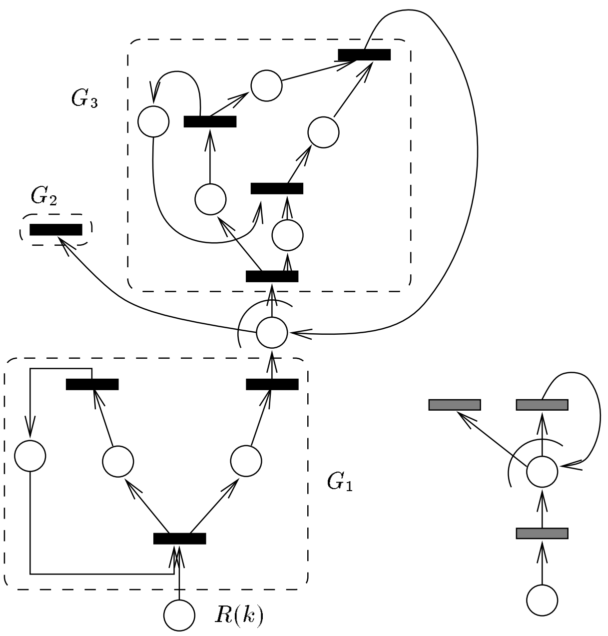
Now, within the class of free choice nets, the race policy and the routing policy coincide. In particular, the race policy can be implemented as a routing policy by choosing

$$\nu^p(n) = \mathrm{argmin}_{t\in p^\bullet}\sigma^t(n).$$

Another question that can be answered but in another context is the power of description of (non timed) Petri Nets with routing. By quasi-immediate reduction of the token flow model (introduced in [16]) to Petri nets with routing and using the fact that token flow models have the power of description of a Turing Machine (see [6]), we can say that Petri nets with routing also have the power of description of Turing machines.

## 2.3 Marked Graph Components

A place in a FCN is *serial* if it has only one input transition and one output transition. The other places are called *routing* places. The *marked graph components* (MGC) of a FCN $S$ are all the sub-nets of $S$ containing only serial places. The *routing* net $\mathcal{R}$ associated with $\mathcal{S}$ is a FCN induced from $\mathcal{S}$ where each MGC is aggregated into one transition, see Figure 2.




A SI-FCN and its MGC associated routing graph

Figure 2: Construction of the routing net of a FCN.

$F$ is a *single input* FCN (SI-FCN) if all MGC's contain a single transition which input place is not serial. This transition is called the input transition of the MGC it belongs to. $F$ is an *acyclic single input* FCN (ASI-FCN) if all its MGC are acyclic single input. An example of a single input acyclic net is given in Figure 3.

# 3 Evolution Equations

In the following, we shall consider the case when firing times are constant, positive, and all multiple of a common unit, which will be taken equal to 1 without loss of generality. We will denote by $K$ the (integer-valued) upper-bound on the firing times.

## 3.1 The (min,+) Algebra

The (min,+) algebra is used in graph theory [12] as well as in discrete event systems [3].

**Definition 3.1 ((min, +) algebra)** *The(min,+) algebra is the semi-field* $\mathbb{R}_{\min} = (\mathbb{R}^\star, \oplus, \otimes)$ *where* $\mathbb{R}^\star = \mathbb{R} \cup \{+\infty\}$*, the law* $\oplus$ *is the minimization, and the law* $\otimes$ *the usual addition.*

In this new algebra, the matrix product is defined of the following way :

$$\forall A \in \mathbb{R}_{\min}^{n\times m}, \forall B \in \mathbb{R}_{\min}^{m\times p} \;\; [A \otimes B]_{ij} = \bigoplus_{k=1}^{m} (A_{ik} \otimes B_{kj}).$$

## 3.2 Counters

The choice of state variables used to describe the system is essential. In the following we will use the counters of the Petri net rather than the markings as it is usually the case.

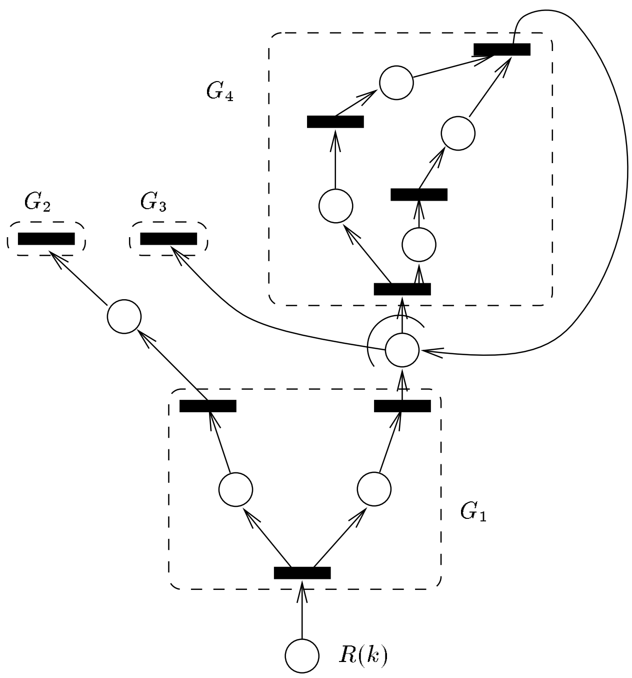



Figure 3: An ASI- FCN.

**Definition 3.2 (counters)** *The counters of a Petri net are the variables $X_i(k)$, $i \in \mathcal{T}$, $k \in \mathbb{N}$ and $X_i(k)$ is the number of times that transition $i$ has initiated a firing by time $k$.*

Any FCN is transformed into an equivalent canonical FCN where transitions belong to either one of the two following subsets of $\mathcal{T}$.

- $\mathcal{T}_m$ is the set of transitions $t$ (of size $T_m$) such that ${}^\bullet t$ contains only serial places.
- $\mathcal{T}_r$ is the set of transitions $t$ (of size $T_r$) such that ${}^\bullet t$ is a single place.

The construction of the canonical form of a free choice net is illustrated in Figure 4.

As a consequence, the counting vector $X(k)$ can be split into $X(k) = (Y(k), Z(k))$ where $Y(k)$ is the counting vector for transitions in $\mathcal{T}_m$ and $Z(k)$ the counting vector for transitions in $\mathcal{T}_r$.

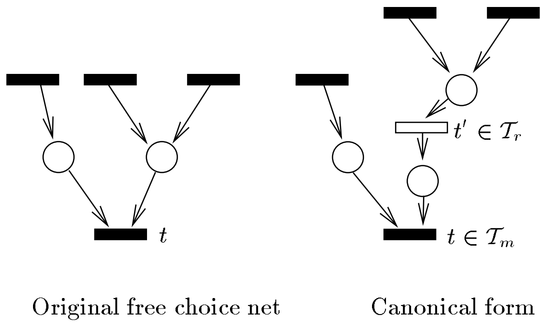



Figure 4: Canonical Form of a free choice net

### 3.3 Counter Equations

After the free choice net has been put into its canonical form, we can define the routing function $\Pi$ from the sequence $\nu$.

**Definition 3.3 (routing function)** *For each routing place $p$ and for every transition $t \in p^\bullet$, let $\Pi^t$ be the function :*

$$\Pi^t(n) = \sum_{i=1}^{n} 1_{\{\nu^{\bullet t}(i)=t\}}\,.$$

$\Pi^t(n)$ gives the number of tokens that are routed to transition $t$ after $n$ tokens have entered place ${}^\bullet t$.

Under the above assumptions, for all $k \in \mathbb{Z}$, the counting vectors $\{Y(k), Z(k)\}$ satisfy the following evolution equation:

$$Y(k) = 0, \quad Z(k) = 0, \quad \forall k < 0, \tag{1}$$

and, for $k \geqslant 0$,

$$Y(k) = \bigoplus_{l=1}^{K} \big(A_l \otimes Y(k-l) \oplus B_l \otimes Z(k-l)\big) \tag{2}$$

$$Z(k) = \Pi\left(\sum_{l=1}^{K} \big(P_l Z(k-l) + Q_l Y(k-l)\big) + R(k)\right) \tag{3}$$

The notations used in the recurrence equations are defined from the net structure as follows:

- The matrix $A_l$ on $\mathcal{T}_m \times \mathcal{T}_m$ is defined by $A_l(t,t') = u$ if the firing time of $t'(\in \mathcal{T}_m)$ is $l$ and there is a serial place between $t'(\in \mathcal{T}_m)$ and $t$, with $\mathcal{M}_0$-marking equal to $u$; $\infty$ otherwise. If there are more than one serial places between $t'$ and $t$, we take $c$ equal to the minimum of their $\mathcal{M}_0$-markings.
- The matrix $B_l$ on $\mathcal{T}_m \times \mathcal{T}_r$ is defined by $B_l(t,t') = u$ if the firing time of $t' \in \mathcal{T}_r$ is $l$ and if there is a serial place between $t'(\in \mathcal{T}_r)$ and $t$, with $\mathcal{M}_0$-marking equal to $u$; $\infty$ otherwise.
- The matrix $P_l$ on $\mathcal{T}_r \times \mathcal{T}_r$ is defined by $P_l(t,t') = 1$ if the firing time of $t'(\in \mathcal{T}_r)$ is $l$ and there is a place connecting $t'$ to $t$; 0 otherwise.
- The matrix $Q_l$ on $\mathcal{T}_r \times \mathcal{T}_m$ is defined by $Q_l(t,t') = 1$ if the firing time of $t'(\in \mathcal{T}_m)$ is $l$ and there is a place connecting $t'$ to $t$; 0 otherwise.
- $R_t(k)$ is the cumulated exogenous process in the single place ${}^\bullet t$ up to time $k$.
- For all vectors of integers $Z = (Z_1, \ldots, Z_q)$, where $q = |\mathcal{T}_r|$, $\Pi(Z)$ is the vector of integers:

$$\Pi(Z) = (\Pi_1(Z_1), \ldots, \Pi_q(Z_q)).$$

# 4 Total Number of Firing

Let $Y = Y(\infty)$ and $Z = Z(\infty)$ denote the vectors counting the total number of firings of the transitions. One can characterize the presence or the absence of deadlocks and related properties, like starvation, directly from $Y$ and $Z$: for instance, given an initial marking $\mathcal{M}_0$ such that $R(-1) = 0$, the system without exogenous input (i.e. $R(k) = R(-1) = 0$, $\forall k$) is deadlocked if and only if the $Z$ and $Y$ vectors associated with this $R(k)$ function are both equal to 0.

The integer-valued vectors $Z$ and $Y$ satisfy the system of equations

$$Y = A \otimes Y \oplus B \otimes Z \tag{4}$$

$$Z = \Pi\left(P \times Z + Q \times Y + R\right), \tag{5}$$

where

$$A = \bigoplus_{l=1}^{K} A_l, \ B = \bigoplus_{l=1}^{K} B_l, \ P = \sum_{l=1}^{K} P_l,$$

$$Q = \sum_{l=1}^{K} Q_l \text{ and } R = \lim_{k \to \infty} R(k).$$

Note that this system does not depend on the variables $\sigma_t$ anymore. All qualitative properties like starvation, liveness or deadlock are associated with the routing function $\Pi$, the topology, the initial marking and the exogenous $R$ vector only, and *not* with timing variables (neither firing times nor arrival epochs of exogenous tokens).

This property allows us to establish relations between the liveness of the underlying un-timed Petri net and the absence of deadlock or starvation in the timed version of the net. In particular, if the un-timed FCN is live, then the timed version never reaches a deadlock (this is not true for general Petri nets). Furthermore, if the un-timed FCN is live and if all the functions $\Pi^t$ are equitable (the function $\Pi^t$, $t \in \mathcal{T}$ is equitable if $\Pi^t(m)$ goes to infinity when $m$ goes to infinity) then, there will be no starvation in the timed version, that is, all the transitions will fire infinitely often. For more on this see [9].

**Lemma 4.1** *The total numbers of firings before deadlock is the minimal non-negative solution of the system of equations (4)-(5).*

**Proof** The operator

$$\begin{array}{rlll} \Phi: & \mathbb{N}^{|\mathcal{A}|} \times \mathbb{N}^{|\mathcal{B}|} & \to & \mathbb{N}^{|\mathcal{A}|} \times \mathbb{N}^{|\mathcal{B}|} \\ & (Y, Z) & \to & (A \otimes Y \oplus BZ, \ \Pi(PZ + QY + R)) \end{array}$$

is component-wise non-decreasing and integer-valued. Any finite non-negative solution $(S_1, S_2)$ of (4)-(5) is a finite fixed point of $\Phi$: $(S_1, S_2) = \Phi(S_1, S_2)$.

Let us now define the sequence of vectors $(\alpha(k), \beta(k))_{k \in \mathbb{N}}$ in $\mathbb{N}^{|\mathcal{A}|} \times \mathbb{N}^{|\mathcal{B}|}$ by $(\alpha(0), \beta(0)) = (0, 0)$ and

$$(\alpha(k+1), \beta(k+1)) = \Phi(\alpha(k), \beta(k)).$$

This sequence is component-wise non-decreasing and so, it has a limit when $k$ goes to infinity denoted $(\alpha, \beta)$. The vector $(\alpha, \beta)$ is a fixed point of $\Phi$ (actually, whenever $(\alpha, \beta)$ is finite, the limit vector is reached in a finite number of steps).

A straightforward induction on $n$ shows that for any other non-negative fixed point $(S_1, S_2)$ of $\Phi$, $(\alpha, \beta) \leqslant (S_1, S_2)$, where the order relation is component-wise, which proves the first part of the lemma.

If the net reaches a deadlock, the total firing vector $X = (Y, Z)$ is finite and non-negative and this vector is a solution of (4)-(5).

Conversely, let us assume that this system admits a finite solution. Then $(\alpha, \beta)$ is necessarily finite. Consider the sequences $(Y'(k), Z'(k))$ defined by

$(Y'(k), Z'(k)) = (0, 0)$, for $k < 0$ and

$$\begin{array}{rcl} Y'(k) & = & \bigoplus_{l=1}^{M} (A_l \otimes Y'(k-l) \oplus B_l \otimes Z'(k-l)) \\ Z'(k) & = & \Pi \left( \sum_{l=1}^{M} (P_l \times Z'(k-l) + Q_l \times Y'(k-l)) + R \right) \end{array}$$

for $k \geqslant 0$. We get $(Y(k), Z(k)) \leqslant (Y'(k), Z'(k))$ by induction. Similarly, we verify by induction that for all $k \in \mathbb{N}$, that $(Y'(k-1), Z'(k-1)) \leqslant (\alpha(k), \beta(k))$ component-wise because $Y'(k)$ and $Z'(k)$ are increasing sequences. Therefore, $(Y(\infty), Z(\infty)) \leqslant (\alpha, \beta)$ which is finite, and finally, $(Y(\infty), Z(\infty)) = (\alpha, \beta)$. $\square$

## 4.1 Parallel Computation

The previous section provides another way of computing the total number of firings in the case when the system reaches a deadlock. Indeed, we can iterate the operator $\Phi$ on a null vector until convergence to a fixed point which will be the total number of firings. This computation has several advantages. First, the convergence to the fixed point is faster when iterating function $\Phi$ than when iterating equations 2 and 3. Indeed, the previous section shows that $(\alpha(k), \beta(k))$ , the result of the $k$-th iteration of function $\Phi$ is larger than $X(k), Y(k)$, the result of the $k$-th iteration of equations 2 and 3. Moreover, this computation can be carried out in parallel on each component of vector $X$ by computing synchronously for all $i$. If $\Phi = (\Phi_1, \cdots, \Phi_T)$, then $X_i(n) = \phi_i(X(n-1))$.

Finally, this can be further improved by removing all the synchronizations after each step of the computation.

Let us denote by $\tilde{X}_i(n)$ the result of the $n$-th iteration of the function $\Phi_i$. If the iterations of all the functions $\Phi_i$ are not synchronous, then, $\tilde{X}_i(n)$ may be different from $X_i(n)$. If at a given instant $\theta$, each $\Phi_i$ has been iterated $n_i$ times with $n_1 \leqslant n_2 \leqslant \cdots \leqslant n_T$ (up to a re-numbering of the transitions), then we certainly have $\tilde{X}_1(n_1) \geqslant X_1(n_1)$ and $\tilde{X}_T(n_T) \leqslant X_T(n_T)$ by monotonicity of all the functions $\Phi_i$. However, note that $n_1$ is an upper bound of the number of iterations $s$ performed by the synchronous algorithm by time $\theta$. Therefore, we have

$$\forall i \leqslant T, \ \tilde{X}_i(n_i) \geqslant X_i(n_1) \geqslant X_i(s). \tag{6}$$

Note that for all $i$, the sequence $\tilde{X}_i(n)$ is in $\mathbb{N}$, non-decreasing and is smaller than $X_i$, the $i$-th component of the fixed point of $\Phi$. When for all $i$, $\tilde{X}_i(n_i+1) = \tilde{X}_i(n_i)$, then $(\tilde{X}_i(n_i))_{i \leqslant T}$ is a fixed point of $\Phi$ smaller than $X$ and therefore is equal to $X$. Therefore, this method also converges towards the solution in a fixed number of steps. The stopping test is when for all $i$, $\tilde{X}_i(n_i+1) = \tilde{X}_i(n_i)$ (which can be done asynchronously). Equation (6) says that the asynchronous version converges faster in time than the synchronous version (even if it requires more iterations).

On a practical point of view, this algorithm with no synchronization can be implemented using only non blocking communications on a message passing parallel architecture. This is more efficient than the synchronous computation because it allows overlap between computations and communications.

# 5 Average Number of Firings

The operator $\Pi$ is not associative in general. This makes several efficient time parallel methods unusable to simulate the system. Space parallel algorithms have been used in [5] with good efficiency but it is of no doubt that a time parallel method would be interesting when very long simulated times are required.

## 5.1 Fluid Model

In some particular cases the evolution can be described by associative operations as for the *fluid model* associated with the system. We recall that the evolution equation of a FCN is given by Equations (2)-(3).

We assume that the function $\Pi$ is ergodic (*i.e.* for all $t \in \mathcal{T}$ the limit $\mathbf{E}\Pi_t(n)/n = f_i$ exists. Note that this assumption is verified in most practical cases. For example $\Pi$ can be a periodic process or an ergodic random process (including i.i.d. routing as a special case).

Now, we can define a diagonal matrix $F$ as the asymptotic routing rate of function $\Pi$ by $F_{i,i} = f_i$. We consider the "fluid" model associated with a FCN where the discrete state variables $X(n) = (Y(n), (Z(n)) \in \mathbb{N}$ are replaced by the quantities $(U(n), V(n)) \in \mathbb{R}$ denoting the quantity of fluid which has passed through the transitions by time $n$. We replace the routing function $\Pi$ by the asymptotic rate matrix $F$ and the evolution equation of the fluid model is

$$U(k) \quad = \quad \bigoplus_{l=1}^{K} (A_l \otimes U(k-l) \oplus B_l \otimes V(k-l)) \tag{7}$$

$$V(k) \quad = \quad F\left(\sum_{l=1}^{K}(P_l V(k-l) + Q_l U(k-l)) + R(k)\right) \tag{8}$$

A similar model has been studied in [7] to interpret these evolution equations as dynamic programming equations of a stochastic decision process. Note also that the evolution of this fluid model is still in discrete time, unlike in most of fluid approximations of discrete time system. Here, only the "tokens" are transformed into "volume of fluid".

## 5.2 Relations Between the Fluid and the Discrete Model

In general the throughput in the fluid model is an upper bound of throughput in the discrete model. More precisely, we have the following lemma.

**Lemma 5.1** *Assume that the routing sequences $\{\nu^p(i)\}$ are i.i.d. For all $k \in N$, $(U(k), V(k)) \geqslant (\mathbf{E}[Y(k)], \mathbf{E}[Z(k)])$, where $\mathbf{E}[.]$ denotes the expectation.*

**Proof** This inequality holds by induction on $k$. First note that $\mathbf{E}Y(0) = U(0)$ and $\mathbf{E}Z(0) = V(0)$. Taking expectation on both side of Equation (2) and using the concavity of the min operator yields

$$\begin{aligned}
\mathbf{E}[Y(k)] &\leqslant \bigoplus_{l=1}^{K}\Big(A_l \otimes \mathbf{E}[Y(k-l)] \oplus B_l \otimes \mathbf{E}[V(k-l)]\Big) \\
&\leqslant \bigoplus_{l=1}^{K}\Big(A_l \otimes U(k-l) \oplus B_l \otimes V(k-l)\Big) \\
&= U(k).
\end{aligned}$$

As for the comparison of $V(k)$ and $\mathbf{E}[Z(k)]$, we take expectation in Equation (3) for transition $t$. If we set $\mathbf{P}_t = \mathbf{P}\left[\nu^{\bullet t}(1) = t\right]$, we obtain after some computations,

$$\begin{aligned}
&\mathbf{E}[Z_t(k)] \\
&= \mathbf{P}_t \times \left(\sum_{l=1}^{M}(P_l \mathbf{E}\left[Z(k-l)\right] + Q_l \mathbf{E}\left[Y(k-l)\right]) + R(k)\right)_t \\
&\leqslant \mathbf{P}_t \times \left(\sum_{l=1}^{M}(P_l V(k-l) + Q_l U(k-l) + R(k)\right)_t .
\end{aligned}$$

To finish the proof, note that when $\{\nu^p(i)\}$ are i.i.d. sequences, then $\mathbf{P}\left[\nu^{\bullet t}(1) = t\right] = F(t,t)$. This implies that

$$\begin{aligned}
&\mathbf{P}_t \times \left(\sum_{l=1}^{M}(P_l V(k-l) + Q_l U(k-l) + R(k)\right)_t \\
=\;& F\left(\sum_{l=1}^{M}(P_l V(k-l) + Q_l U(k-l) + R(k)\right)_t \\
=\;& V_t(k).
\end{aligned}$$

□

However, the fluid and the discrete model can be very different indeed. One can exhibit examples where the fluid model is "live" ($V(k)$ is not bounded in $k$) and the discrete model is in a deadlock (all $Z(k)$ have a finite limit in $k$). However, in the case of acyclic single input systems, new relations between the fluid model and the discrete model can be given. In particular, for i.i.d. routed network, the fluid model gives the same average number of events (see Lemma 5.2).

In the following, we will assume that the free choice net under consideration is an acyclic single input FCN, and we will denote by $t^{(i)}$ the unique transition of $\mathcal{T}_r$ which is an input of a MGC $G_i$.

If $F$ is a ASI-FCN, the equations for the number of events can be put in a simplified form.

Let $t$ be an input transition of a MGC $G$ and $t'$ be an arbitrary transitions in $G$. If $\Lambda(t,t')$ is the set of all the paths in $G$ from transition $t$ to transition $t'$. By acyclicity of each MGC, all these paths are finite. Then, for each path $\lambda$ in $\Lambda(t,t')$, we define

$$\begin{aligned} w(\lambda) &= \sum_{t\in\lambda} \sigma_t \\ \mu(\lambda) &= \sum_{p\in\lambda} \mathcal{M}_0(p). \end{aligned}$$

Now we can define $M$:

$$M = \max_{t,t'\in\mathcal{T}} \max_{\lambda\in\Lambda(t,t')} w(\lambda).$$

In words, $M$ is the maximum of the sums of the timings on all the paths contained in MGCs.

Now, the counting vector verifies

$$\begin{aligned} Y(k) &= \bigoplus_{l=1}^{M} (C_l \otimes Z(k-l)) \\ Z(k) &= \Pi\left(\sum_{l=1}^{M} (P_l Z(k-l) + Q_l(C_l \otimes Z(k-l))) + R(k)\right). \end{aligned}$$

Where the matrices $C_l$ trace the firings counters from the input transition to each transition in a MGC. More precisely, $C_l$ of size $T_m \times T_r$ verifies : $C_l(t,t') = u$ if

$$u = \min_{\lambda\in\Lambda(t',t),w(\lambda)=l} \mu(\lambda).$$

The sequences $\{P_l\}$ and $\{Q_l\}$ are extended to take into account the time space extension from $K$ to $M$ by matrices full of 0.

Since each MGC is single input, the matrices $C_l$ have at most one finite element per line. Therefore, we can define $C'_l, P'_l$ and $R''$ by $C_l \otimes Z(k-l) = C'_l \times Z(k-l) + R''(l)$ and $P'_l$ by $P'_l = P_l + Q_l \times C'_l$. In this case, the counting vector verifies the equations:

$$Y(k) = \bigoplus_{l=1}^{M} (C_l \otimes Z(k-l)) \tag{9}$$

$$Z(k) = \Pi\left(\sum_{l=1}^{M} (P'_l Z(k-l)) + R'(k)\right), \tag{10}$$

with $R'(k) = R(k) + R''(k)$. Similarly, the fluid model verifies:

$$U(k) = \bigoplus_{l=1}^{M} (C_l \otimes V(k-l)) \tag{11}$$

$$V(k) = F\left(\sum_{l=1}^{M} (P'_l V(k-l)) + R'(k)\right). \tag{12}$$

**Lemma 5.2** *Let $S$ be an acyclic single input free choice net. Assume that the routing sequences $\{\nu^p(i)\}$ are i.i.d. in $i$ and mutually independent in $p$. Then we get $\mathbf{E}[Z(k)] = V(k)$.*

**Proof** The proof holds by induction on $k$. First note that $\mathbf{E}[Z(0)] = V(0) = 0$. Taking expectations on both sides of (10), we obtain (on line $t$)

$$\begin{aligned}
\mathbf{E}\left[Z_t(k)\right] &= \mathbf{E}\left[\sum_{i=1}^{\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R'(k)\right)_t} \mathbf{1}_{\nu^{\bullet t}(i)=t}\right] \\
&= \sum_{i=1}^{\infty}\mathbf{E}\left[\mathbf{1}_{\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R'(k)\right)_t \geqslant i}\ \mathbf{1}_{\nu^{\bullet t}(i)=t}\right] \\
&= \sum_{i=1}^{\infty}\mathbf{E}\left[\mathbf{1}_{\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R(k)\right)_t > i-1}\ \mathbf{1}_{\nu^{\bullet t}(i)=t}\right].
\end{aligned}$$

But since the events $\{\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R(k)\right)_t \leqslant i-1\}$ and $\{\nu^{\bullet t}(i) = t\}$ are independent, we set $\mathbf{P}_t = \mathbf{P}\left[\nu^{\bullet t}(1) = t\right]$ and we get

$$\begin{aligned}
\mathbf{E}\left[Z_t(k)\right] = \mathbf{P}_t \times \sum_{i=1}^{\infty}\mathbf{P}\left[\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R(k)\right)_t > i-1\right] \\
= \mathbf{P}_t \times \mathbf{E}\left[\left(\sum_{l=1}^{M}(P_l' Z(k-l))+R(k)\right)_t\right] \\
= \mathbf{P}_t \times \left(\sum_{l=1}^{M}(P_l'\mathbf{E}\left[Z(k-l)\right])+R(k)\right)_t \\
= \mathbf{P}_t \times \left(\sum_{l=1}^{M}(P_l' V(k-l))+R(k)\right)_t.
\end{aligned}$$

The last equality is obtained by induction. Finally the i.i.d. assumption and the law of large numbers imply that $F(t,t) = \mathbf{P}\left[\nu^{\bullet t}(1) = t\right] = \mathbf{P}_t$.

$$\mathbf{E}\left[Z_t(k)\right] = F\left(\sum_{l=1}^{M}(P_l' V(k-l))+R'(k)\right)_t = V_t(k). \tag{13}$$

This last equation establishes the result.
□

## 5.3 Time Parallel Simulation of the Fluid Model

The time parallel techniques used for marked graphs ([1]) can be used to simulate the system. Equation (12) is a linear equation of order $M$ and using standard state extension techniques, it can be transformed into an equation of order one of the form:

$$\tilde{V}(n) = \tilde{F}(\tilde{P}\tilde{V}(n-1)+\tilde{R}(n), \tag{14}$$

with

$$\tilde{V}(n) = \left(V(n)^t, V(n-1)^t, \cdots, V(n-M)^t\right)^t,$$

$$\tilde{F} = \begin{pmatrix} F & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & I & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{pmatrix}, \quad \tilde{P} = \begin{pmatrix} P_1' & P_2' & \cdots & P_M' \\ I & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} \end{pmatrix},$$

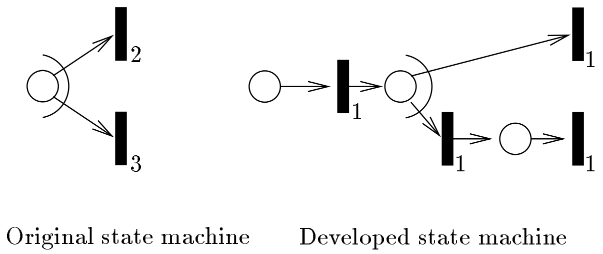



Figure 5: Transformation of a SM into a SM with unit firing times

$$\tilde{R}(n) \quad = \quad \left(R'(n)^t, 0, \cdots, 0\right)^t,$$

where $()^t$ denotes the transpose function, I is the identity matrix of size $T_r \times T_r$, $\mathbf{0}$ is the null matrix of size $T_r \times T_r$ and 0 is the null vector of size $T_r \times 1$.

Equation (14) corresponds to the evolution of a system $\tilde{S}$ which is a state machine with $MT_r$ transitions with timings all equal to one. When we develop this equation, we obtain

$$\tilde{V}(n) = (\tilde{F}\tilde{P})^n \tilde{V}(0) + \sum_{k=0}^{n-1} (\tilde{F}\tilde{P})^k \tilde{F}\tilde{R}(k), \tag{15}$$

which can be computed by applying two successive parallel prefix scans, one to get the powers of matrices: $(FW)^k,\ k \leqslant n$ and one to get the partial sums, $\sum_{i=0}^{k} (FW)^i \tilde{F}\tilde{R}(i),\ k \leqslant n$.

The complexity of this algorithm when neglecting the communication cost (PRAM model) with $Pr$ processors is $O\left(|\tilde{F}|^3 \frac{n}{Pr} + log(n)\right)$.

At this point, one can note that the complexity of the simulation depends heavily on the size of the system: $|\tilde{F}| = MT_r$. In the next section we show how to reduce this size and we describe the minimal representation of Equation (12).

# 6 Minimal Representation of State Machines

## 6.1 Development of a Place

Note that state machines (SM) constitute a subclass of ASI-FCN with all its MGC reduced to transitions: $\mathcal{T}_m = \emptyset$ and $\mathcal{T}_r = \mathcal{T}$.

Any state machine $S$ can be *developed* into an equivalent state machine $S_d$ where each transition has timing 1. This transformation is given in Figure 5.

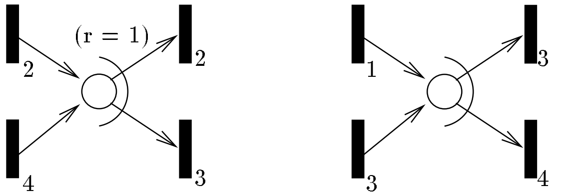
Note that after such a transformation, the evolution equation of $S_d$ is

$$V_d(n) = F_d(P_d V_d(n-1) + R_d(n)), \tag{16}$$

which size is

$$\Gamma \stackrel{\text{def}}{=} \sum_{p \in \mathcal{P}} \max_{t \in p^\bullet} \sigma(t) - 1.$$

After the development of the places, the size of the matrix $F_d$ involved in Equation (16) is $\Gamma + T$ and the system is directly of order one. Therefore, the complexity of the temporal simulation of the system $S_d$ is $O((\Gamma + T)(n/Pr + log(n)))$, which compares advantageously with the complexity of the simulation of the system $\tilde{S}$.




Original timings After retiming with $r = 1$

Figure 6: Illustration of the retiming of a state machine.

## 6.2 Retiming

Since the size of the system now depends on the maximum of the firing times on all output transitions of the routing places, one can further reduce the size of the system by changing those timings without altering the behavior of the system. This technique is called a *retiming* of the system and was first introduced in [17] to optimize digital circuits.

Here, a retiming can also be viewed as the dual of a firing. A new representation of the system is obtained by moving the firing times of the transitions around places the same way tokens are moved around transitions during a firing. A *retiming* is a function $r$ from $\mathcal{P}$ (the set of the places) to $\mathbb{N}$.

Let $p \in \mathcal{P}$. After retiming $r$, For all $t \in {}^\bullet p, \sigma'_t = \sigma_t - r(p)$ and for all $t \in p^\bullet, \sigma'_t = \sigma_t + r(p)$.

In Figure 6, the retiming procedure is illustrated with an example where the retiming value is 1. All input transitions have a timing decreased by one and all output transitions have a timing increased by one.

In the following, we will only consider *valid* retimings, *i.e.* retimings leading to positive timings on all the transitions .

**Lemma 6.1** *Let us assume that $S^1 = (\mathcal{P}, \mathcal{T}, \mathcal{M}, \Sigma^1, R^1)$ is a strongly connected state machine. We construct another state machine $S^2 = (\mathcal{P}, \mathcal{T}, \mathcal{M}, \Sigma^2, R^2)$ with the same topology and $\Sigma^2 = r(\Sigma^1)$, for a valid retiming $r$ $(\sigma_i^2 = \sigma_i^1 + r({}^\bullet i) - r_(i^\bullet))$. If we choose $R^2(n)$ in an appropriate manner, then there exists constants $C_i$ such that*

$$V_i^2(n) = V^1(n + C_i), \ \forall n \in \mathbb{N}, \ i \in \mathcal{T}.$$

**Proof** In the following, we shall assume that we retime only one place of the SM, the result for a general retiming follows by retiming each place one by one. We call $p$ the place which is retimed. Therefore, we get:

$$\sigma_t^2 = \begin{cases} \sigma_t^1 - 1 & \text{if } t \in {}^\bullet p \\ \sigma_t^1 + 1 & \text{if } t \in p^\bullet \\ \sigma_t^1 & \text{otherwise.} \end{cases}$$

We also restrict ourselves to an elementary input process with only one exogenous token. The behavior with the total input process will be the superposition of all the elementary input processes.

We introduce the daters of the system: $\mathcal{D}_t(n)$ is the epoch of the beginning of the $n$-th firing of transition $t$.

We have to distinguish two cases. If the input token is in place $p$, then we have:

$$\mathcal{D}_t^2(1) = \begin{cases} \mathcal{D}_t^1(1) & \text{if } t \in p^\bullet \\ \mathcal{D}_t^1(1) + 1 & \text{otherwise.} \end{cases}$$

From this point on, for all the transitions, we have: $\mathcal{D}_t^2(n+1) - \mathcal{D}_t^2(n) = \mathcal{D}_t^1(n+1) - \mathcal{D}_t^1(n)$. This implies that $V_t^2(k) = V_t^1(k)$ for $t \in p^\bullet$ and $V_t^2(k) = V_t^1(k+1)$ otherwise.

If the input token is not in place $p$ but in place $q \neq p$, then

$$\mathcal{D}_t^2(1) = \begin{cases} \mathcal{D}_t^1(1) - 1 & \text{if } t \in p^\bullet \\ \mathcal{D}_t^1(1) & \text{otherwise.} \end{cases}$$

From this point on, for all the transitions, we have: $\mathcal{D}_t^2(n+1) - \mathcal{D}_t^2(n) = \mathcal{D}_t^1(n+1) - \mathcal{D}_t^1(n)$. This implies that $V_t^2(k) = V_t^2(k-1)$ for $t \in p^\bullet$ and $V_t^2(k) = V_t^1(k)$ otherwise.

This ends the proof for elementary input processes. When we superpose two input processes, the counter vector is the sum of the counter vectors for each elementary input process. To keep the relation between the original system and the retimed system, we need the constants $C_t$ to be the same for both elementary input processes. In order to obtain this result, we need to modify the input process of the retimed system. The *delay* between two input tokens is the time elapsed between the instant an arbitrary transition fire for the first time the first token and the instant when it fires the second token for the first time. Note that this difference is the same for all transitions.

By retiming, this difference can change but if we delay the arrival epoch of the latest token ( which is done by a modification of the function $R^2(n)$, then we can impose this difference to be the same in the original system as well as in the retimed system. Now, it should be clear that, when the two input process are superposed, we still get the same relation: $V_t^2(k) = V_t^2(k + C_t)$. $\square$

In other words, the lemma says that we can transform a state machine into an equivalent state machine by retiming.

To obtain the best representation of a given state machine, we make two transformations: first we apply a retiming of the state machine, then we develop the new state machine.

We say that a retiming is *optimal* if the retimed system (after development) is of minimal size among all possible retimed systems.

The optimal retiming can be computed in polynomial time in the size of the system, in $O(|\mathcal{T}||\mathcal{P}|\log^2|\mathcal{P}|)$. This results comes from the analogy with retiming of synchronous circuitry ([17]).

Therefore a refined time parallel simulation of the fluid model goes like this:

1. apply optimal retiming,
2. develop the retimed system,
3. simulate by time parallelism,
4. reconstruct the result in the original system by applying Lemma 6.1.

**Markov Chain** The evolution of a SM with Bernoulli routing can also be modeled by a discrete time semi-Markov process. This process can be transformed into a Markov chain by extending the state space using the development procedure described in Section 6.1. Another use of the retiming procedure is that it gives a Markov chain of minimal size that can be used to analyze the system.

# 7 Stability Criterion

The stability of a stochastic system can be viewed as the fact that the markings in all the places in the system remain bounded in probability. More precisely, if $\mathcal{M}(t)$ is the marking at time $t$, then $\mathcal{M}(t)$ converges in law to a finite stationary process.

In [4], a necessary and quasi-sufficient stability condition for SI-FCN is given. If we assume that at time zero, the net is in a deadlock (no transition can fire) and if a finite number of arrivals create a finite number of firings of all the transitions almost surely, then the stability condition can be put under the form $\rho < 1$ where $\rho$ is given by:

$$\rho = \lambda\gamma(0),$$

where

- $\lambda$ is the intensity of the input process
- and
$$\gamma(0) = \max_{G \in \mathrm{MGC}} \mu(G)\mathbf{E}[X_G^{[0]}],$$

where $\mu(G)$ is the cycle time of the MGC $G$ and $X_G^{[0]}$ is the total number of firing of the input transition of $G$ when the input process is reduced to a single token entering the system at time 0.

The computation of $\mu(G)$ can be carried out using an algorithm of Karp ([14]) and the computation of $\mathbf{E}[X_G^{[0]}]$ can be done by using the fluid equation. Let $t$ be the input transition of $G$, then $\mathbf{E}[X_t^{[0]}]$ is the $t$-th component of the solution of the linear system:

$$X = F\left(P'X + R^{[0]}\right), \tag{17}$$

Where $R^{[0]}$ is the input process with only one token in the system. Note that since $R^{[0]}$ is a finite input process, this system has a finite solution by assumption. The minimal solution of the system (17) gives $\mathbf{E}[X^{[0]}]$.

### 7.1 Complexity Result

This stability condition can be checked by solving the linear system (17) ($O(T^3)$) using a classical Gauss algorithm and by computing the cycle time of each MGC. This can be done in ($O(\Gamma^3)$) using the algorithm of Karp, [14].

We can also note that the computation of $\mu(G)$ can be carried out independently (in parallel) for each MGC.

## 8 Experiments

To implement the time parallel algorithm described in Section 5.3, we have adapted a parallel program originally written to compute $A_1 \bullet A_2 \bullet \cdots \bullet A_i$, $i \leqslant n$ where $\bullet$ is an associative operation on the $A_i$'s and $n$ is large, see [10]. Our program uses the PVM passing message protocol. It was run on a farm of a dozen of Sun workstations ranging from SS2 to SS20. The main feature of the program is that it uses a dynamic workload balancing protocol taking into account the load coming from other users of the processors, see [10]. The main problems to measure the quality of the algorithm are the processor sharing with other users and the fact that the speed of the processors ranges from 1 to 3. To avoid the first problem we have run the program during periods of very low activity of the workstations. As for the heterogeneous processors, they result in a non regular curve for the efficiency (see Figure 8. We have computed the running time of the program using from one to nine processors to simulate a system of moderate size (around twenty transitions) for 100,000 units of time. Figures 7 and 8 respectively depict the speed and the efficiency of the program when the number of processors grows. The speed goes from around 500 seconds to 50 seconds and the efficiency stays high. The fact that the speed curve is not always decreasing comes from the fact that the exogenous load may have changed during the simulation.

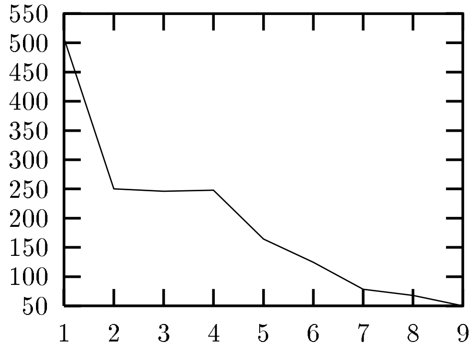



Figure 7: Total time of simulation when the number of processors increases.

Finally, although the results show a good efficiency, one has to take into account the fact that the processors are different. The drop noticed with 4 processors comes from the fact that the fourth processors added to the list was slow (SS2) respectively to the others. The fact that the final efficiency is higher than one can also be explained by the fact that the last one added to the list is the fastest of all.

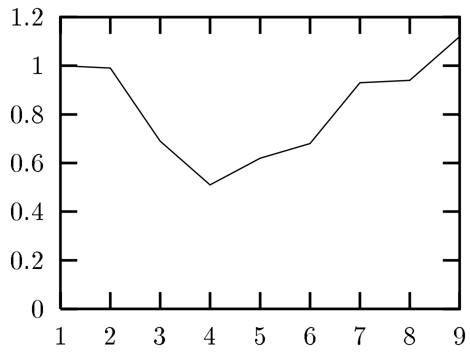



Figure 8: Efficiency versus number of processors.

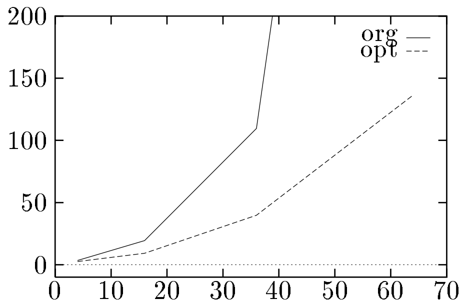



Figure 9: Simulation of the original system versus the optimal retimed system

To show the interest of the minimal representation, we have run this program on a set of state machines of increasing size (ranging from 4 to 64 transitions with twice as many places) to test the efficiency of retiming. This was done over a dozen of workstations. The state machines we have used to carry out the series of test are dual of the marked graph job-shop models (see for example [13] for a presentation of the marked graph job-shop model). In Figure 9, Curve "org" shows the time it takes to simulate the original system for 10,000 units of time. Curve "opt" shows the duration of the simulation after optimal retiming of the system.

# 9 Conclusion

Algebraic simulation may not be as multipurpose as classical discrete event simulation. However, for specific uses, which are presented here, it provides efficient parallelizations which are not possible under the classical approach. For instance, the computation of fixed points and the use of unbounded parallelism provided by time parallel techniques are usually difficult to use within the event oriented approach. Finally, the optimization of the representation of the system is also specific to the equational approach and opens new fields of investigation for the future.

## References


[1] F. Baccelli and M. Canales. Parallel simulation of stochastic Petri nets using recurrence equations. *ACM Transactions on Modeling and Computer Simulation*, 3(1):20–41, 1993.

[2] F. Baccelli, G. Cohen, and B. Gaujal. Recursive equations and basic properties of timed Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 1(4), 1992.

[3] F. Baccelli, G. Cohen, J.P. Quadrat, and J.G. Olsder. *Synchronization and Linearity*. Wiley, 1992.

[4] F. Baccelli, S. Foss, and B. Gaujal. Structural, temporal and stochastic properties of unbounded free choice Petri nets. Technical Report 2411, INRIA, 1994.

[5] F. Baccelli, N. Furmento, and B. Gaujal. Parallel and distributed simulation of free choice Petri nets. In *PADS*, 1995.

[6] J.T. Buck. *Scheduling dynamic dataflow graphs with bounded memory using the token flow model*. PhD thesis, University of California in Berkeley, September 1993.

[7] G. Cohen, S. Gaubert, and J.P. Quadrat. Algebraic system analysis of timed Petri nets. In *Idempotency*, Cambridge University Press, 1995. To appear.

[8] A. Fersha. *Handbook of parallel and distributed computing*, chapter Parallel and distributed simulation of discrete event systems. McGraw-Hill, 1995.

[9] B. Gaujal. *Parallélisme et simulation des systèmes à événements discrets*. PhD thesis, University of Nice-Sophia Antipolis, June 1994.

[10] B. Gaujal, A. Jean-Marie, and S. Mazeau. Strategies for load balancing in distributed computations with associative operations. In *QMIPS*, London, 1994.

[11] P. Glasserman and D.D. Yao. *Monotone structure in discrete-event systems*. Wiley, 1994.

[12] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, Paris, 1979.

[13] H.P. Hillion and J.M. Proth. Performance evaluation of job shop systems using timed event graphs. *IEEE Transactions on Automatic Control*, 34:3–9, 1985.

[14] R.M. Karp. A characterization of the minimal cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[15] L. Kleinrock. *Queueing systems, Vol 1: theory*. J.Wiley & Sons, New York, NY, 1975.

[16] E. Lee. Consistency in data-flow graphs. *IEEE Transactions on Parallel and Distributed systems*, 2(2), April 1991.

[17] C.E. Leiserson and J.B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.

[18] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance models of multiprocessor systems*. MIT Press, Cambridge, Mass., 1986.

[19] T. Murata. Petri nets: Properties, anlysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, 1989.