# sPBC: A Markovian Extension of Finite Petri Box Calculus *


| Hermenegilda Macià | Valentín Valero | David de Frutos-Escrig |
|---|---|---|
| Dpto. Matemáticas | Dpto. Informática | Dpto. Sist. Inf. y Programación |
| Esc. Politécnica | Esc. Politécnica | Fac. Matemáticas |
| Univ. Castilla-La Mancha | Univ. Castilla-La Mancha | Univ. Complutense |
| Albacete, SPAIN 02071 | Albacete, SPAIN 02071 | Madrid, SPAIN 28040 |
| hmacia@pol-ab.uclm.es | valentin@info-ab.uclm.es | defrutos@eucmvx.sim.ucm.es |



## Abstract

*In this paper we present a stochastic extension of finite PBC, sPBC, in which actions have a stochastic delay associated, which is taken from a negative exponential distribution. We define for the language both an operational and a denotational semantics, using a class of labelled stochastic Petri net as supporting values. In this paper we restrict our attention to finite sPBC, i.e., we consider neither the iteration operator nor recursive definition of processes.*



**Keywords**: Stochastic Petri Nets, Stochastic Process Algebra, Performance Evaluation, PBC.


## 1 Introduction

The Petri Box Calculus (PBC) [7, 8, 9] is an algebraic model for the description of concurrent systems that separates the synchronization operator from the parallel one, in contrast to the way in which the synchronization is handled in classical process algebras, such as CCS [15], where the synchronization is embedded in the parallel operator. The main property of this algebra is that its operators are selected in such a way that it is relatively easy to define a denotational semantics based on a class of labelled Petri Net, called *boxes*. Two extensions of the language including time have been recently defined, namely tPBC [13] and TPBC [14]. Our contribution in this paper is the definition of a new extension of the basic model considering *multiactions* with stochastic delays, thus obtaining a stochastic (Markovian) extension of PBC called sPBC. These delays associated with multiactions are interpreted as the time that must elapse until the corresponding multiaction can be executed (computed from the instant in which it was activated), but the execution of multiactions takes no time.

Therefore, in sPBC each multiaction will have an associated random delay which follows a negative exponential distribution. Thus, a (basic) stochastic multiaction will be represented by a pair $<\alpha, r>$, where $\alpha$ represents a (classical) multiaction of PBC and $r \in \mathbb{R}^+$ is the parameter of the associated exponential distribution.

Accordingly, sPBC is a stochastic process algebra, similar to many other proposals that we may find in the literature, such as PEPA [11], TIPP [10] or EMPA [4]. In [16] we may already find a translation of PEPA into SPNs. Nevertheless, a main advantage of our approach inherited from those of plain PBC is that our translation is much more natural than those obtainable when starting from *standard* process algebras. This is because the operators in PBC were carefully selected, in particular the separation between synchronization and parallelism.

In order to define the denotational semantics of sPBC we have extended the plain boxes of PBC to obtain a stochastic version, on the basis of Stochastic Petri Nets [1, 2, 3]. Note the main consequence of the introduction of a continuous distribution (negative exponential) to measure delays of multiactions: the probability for two or more stochastic multiactions to be executed at the same time is zero; and the same is true for Stochastic Petri Nets: two or more transitions have zero probability to be executed in parallel (even if they are simultaneously enabled). As a consequence, we get a natural *interleaving semantics*, as opposed to the *true concurrency* exhibited by PBC. Even so, we still have a certain degree of parallelism at the low level, because we are dealing with multiactions (multisets of actions), and thus, a process evolves by executing a multiset of actions in a single step.


*This work has been supported by the CICYT project "Performance Evaluation of Distributed Systems", Ref. TIC2000-0701-C02-02.

In this paper, our goal has been to keep, as far as possible, both the syntax and the operational behaviour of the operators of PBC, but extending them in the adequate way, according to the stochastic interpretation. For instance, for the new multiactions generated by the application of the synchronization operator, we need to select a new parameter for the corresponding exponential distribution [1], and several criteria can be applied to do it. We may find in [12] a nicely detailed discussion of different alternatives for making this selection, and in this case we have concluded that a good option can be to take as the value of the parameter the least of the values involved in the synchronization, which at the intuitive level can be interpreted as taking the delay of the slowest of the values involved in the synchronization as the delay of the new multiaction (capturing the synchronization). Moreover, in case of conflict we adopt the *race policy*, i.e., whenever two or more stochastic multiactions are possible, the fastest one is executed, in the same way as this *race policy* governs the dynamic behavior of Stochastic Petri Nets.

With the use of negative exponential distribution we gain some additional noteworthy advantages, such as the consequences that derive from the memoryless property of this distribution. One important conclusion we may draw is that we do not need to distinguish between several possible memory policies *(resampling, enabling memory* or *age memory)*, which simplifies the formal treatment of the model.

These decisions allow us to define a neat translation of sPBC-expressions into a class of labelled Stochastic Petri Net, called s-boxes. Finally, in this paper we present a direct procedure to construct a CTMC (Continue Time Markov Chain) from a given sPBC-expression, although it is clear that this translation can be also obtained from the corresponding SPNs.

The paper is structured as follows: in Section 2 we present the syntax of finite sPBC-expressions, and a brief description of the language. The operational semantics for sPBC is defined in Section 3, and in Section 4 we present the denotational semantics of *s-boxes*. Finally, Section 5 contains some conclusions and the future work.

# 2 Syntax of Finite sPBC

In this paper we only deal with finite processes, since the definition of the denotational semantics for the recursion and iteration operators requires a more sophisticated formal treatment, as occurs in PBC.

[1]We restrict ourselves to the negative exponential distribution in order to keep the Markovian character of the model.

## 2.1 Notation

Henceforth we will use the following notation:

- $\mathcal{A}$ will be a countable set of action names. For every $a \in \mathcal{A}$, we have the corresponding *conjugate* action, $\widehat{a} \in \mathcal{A}$, such that $a \neq \widehat{a}$ and $\widehat{\widehat{a}} = a$, as in CCS [15]. Letters $a, b, \widehat{a}, ...$ will be used to denote the elements of $\mathcal{A}$.
- $\mathcal{L} = \mathcal{B}(\mathcal{A})$, will represent the set of finite multisets of elements in $\mathcal{A}$ (*multiactions*). Given $\alpha \in \mathcal{L}$, by $\alpha(a)$ we will indicate the number of instances of $a$ in $\alpha$. In particular, we consider that $\emptyset \in \mathcal{L}$, which is used to specify the execution of a multiaction that does not contain any visible action names (internal action).
- The alphabet of a multiaction $\alpha \in \mathcal{L}$ is defined by:

$$A(\alpha) =_{def} \{a \in \mathcal{A} \,|\, \alpha(a) > 0\}$$

- Relabelling functions $f : \mathcal{A} \rightarrow \mathcal{A}$, which preserve conjugates, i.e.: $\forall a \in \mathcal{A},\ f(\widehat{a}) = \widehat{f(a)}$.
- We define the set of stochastic multiactions by $\mathcal{SL} =_{def} \{< \alpha, r > \,|\, \alpha \in \mathcal{L} \text{ and } r \in \mathbb{R}^+\}$. We allow the same multiaction $\alpha \in \mathcal{L}$ to have different stochastic rates in the same specification.
- We define the alphabet of a stochastic multiaction as the alphabet of the corresponding multiaction: $A(< \alpha, r >) =_{def} A(\alpha)$. This definition can be easily generalized to elements of $\mathcal{B}(\mathcal{SL})$: given $\gamma = \{< \alpha_1, r_1 >, \ldots, < \alpha_n, r_n >\} \in \mathcal{B}(\mathcal{SL})$, we define $A(\gamma) =_{def} \cup_{< \alpha_i, r_i > \in \gamma} A(< \alpha_i, r_i >)$.

Next we introduce some operations on multiactions:

- **Addition**
  - *of multiactions*: $\alpha + \beta =_{def} \delta$, where $\delta(a) =_{def} \alpha(a) + \beta(a)$
  - *of multisets of stochastic multiactions*: $\{< \alpha_1, r_1 >, \ldots, < \alpha_n, r_n >\} + \{< \beta_1, s_1 >, \ldots, < \beta_m, s_m >\} =_{def} \{< \alpha_1, r_1 >, \ldots, < \alpha_n, r_n >, < \beta_1, s_1 >, \ldots, < \beta_m, s_m >\}$
- **Synchronization**
  - *of multiactions*: $\alpha \oplus_a \beta =_{def} \gamma$ where:
    $$\gamma(b) = \begin{cases} \alpha(b) + \beta(b) - 1 & \text{if } b = a \,\vee\, b = \widehat{a} \\ \alpha(b) + \beta(b) & \text{otherwise} \end{cases}$$
    which is only applicable when $a \in \alpha$ and $\widehat{a} \in \beta$, or $\widehat{a} \in \alpha$ and $a \in \beta$.

– *of stochastic multiactions:*
$<\alpha, r_1> \oplus_a <\beta, r_2> =_{def} <\alpha \oplus_a \beta, r>$, where $r$=min$(r_1, r_2)$, and it is only applicable if $a \in A(<\alpha, r_1>)$ and $\widehat{a} \in A(<\beta, r_2>)$, or viceversa.

- **Relabelling**
  – *of multiactions*: $f(\alpha) =_{def} \sum_{a \in \mathcal{A}} \alpha(a) f(a)$.
  – *of multisets of stochastic multiactions:*
  $f(\{<\alpha_1, r_1>, \ldots, <\alpha_n, r_n>\}) =_{def} \{<f(\alpha_1), r_1>, \ldots, <f(\alpha_n), r_n>\}$

### 2.2 Static s-expressions

Static s-expressions are used to describe the structure of a concurrent system, while dynamic s-expressions describe the current state of a system (they correspond to unmarked and marked Petri nets, respectively). As a system evolves by executing multiactions, the dynamic s-expression describing its current state changes; this is captured by means of both overbars and underbars that decorate the static s-expression.

Static s-expressions of sPBC are those defined by the following BNF expression:

$$E ::= <\alpha, r> \mid E; E \mid E \,\square\, E \mid E \,\|\, E \mid E[f] \mid E \; sy \; a \mid E \; rs \; a \mid [a : E]$$

where $<\alpha, r>$ stands for the *basic multiaction* (simultaneous execution of all the actions in $\alpha$, after a delay that follows a negative exponential distribution with parameter $r$). $E_1$ ; $E_2$ stands for the sequential execution of $E_1$ and $E_2$, while $E_1 \,\square\, E_2$ is an external choice, governed by a *race policy*. $[f]$ is the relabelling operator, and *rs* the restriction over the single action $a$. The parallel operator, $\|$, represents the (disjoint) parallel execution of both components, as in PBC, i.e., there is no synchronization embedded with it. Synchronization is captured by the operator *sy*, thus the process $E \; sy \; a$ behaves in the same way as $E$, but it can also execute some new multiactions, generated by the synchronization of a pair of actions $(a, \hat{a})$. For instance, the process $E = (<\{a\}, r_1> \,\|\, <\{\hat{a}, b\}, r_2>) \; sy \; a$ can make all the movements of $<\{a\}, r_1> \,\|\, <\{\hat{a}, b\}, r_2>$, but we have also a new movement, executing $\{b\}$, which comes from the synchronization between $a$ and $\hat{a}$ (synchronizations become silent). Finally, $[a : E]$ is a derived operator (*scoping*), which is defined by $[a : E] = (E \; sy \; a) \; rs \; a$. This operator is used to obtain a similar effect to that of the restriction in CCS when applied to a parallel composition (synchronization on $a$ is enforced).

Thus, we have extended PBC by introducing stochastic information over the multiactions; we use stochastic multiactions $<\alpha, r> \in \mathcal{SL}$, where $r \in \mathbb{R}^+$, which is the parameter of a negative exponential distribution that defines the delay until this multiaction can be executed.

We will denote static s-expressions by letters: $E$, $F$, $E_i, \ldots,$ and the set of static s-expressions by *StatExpr*.

### 2.3 Dynamic s-expressions

The operational semantics of sPBC is defined by means of dynamic s-expressions, which derive from the static s-expressions, but annotating them with either upper or lower bars to indicate the *active components* at each instant of time.

$$G ::= \overline{E} \mid \underline{E} \mid G; E \mid E; G \mid G \,\square\, E \mid E \,\square\, G \mid G \,\|\, G \mid G[f] \mid G \; sy \; a \mid G \; rs \; a \mid [a : G]$$

The informal interpretation is exactly the same as in PBC:

- By $\overline{E}$ we indicate that the static s-expression $E$ has been activated, so its stochastic multiactions can be executed. It represents the initial state of $E$.
- With $\underline{E}$ we mean that the static s-expression $E$ has finished its execution, so it can execute no more stochastic multiactions. It is the final state of $E$.

These dynamic s-expressions will be denoted by $G$, $H$, $G_i, \ldots,$ and the set of dynamic s-expressions is called *DynExpr*.

## 3 Operational Semantics

The operational semantics of sPBC is defined in a very similar way to that of [5, 6, 7, 9] to define the operational semantics of PBC, but there are some slight differences with the inaction rules, which capture the equivalence of dynamic s-expressions, and there are also some significant differences with the (stochastic) multiaction transition rules, the most important one being that now an interleaving semantics is considered, whose interpretation is described later.

### 3.1 Inaction rules and structural equivalence

Inaction transitions are introduced in PBC to establish the active components of a dynamic expression. For instance, an inaction rule for the parallel operator of PBC is:

$$\overline{E \,\|\, F} \xrightarrow{\emptyset} \overline{E} \,\|\, \overline{F}$$

which means that in order to activate the parallel composition we have to activate each one of its components. But

| | | |
|---|---|---|
| $\overline{E;F} \xrightarrow{\emptyset} \overline{E};F$ | $\underline{E};F \xrightarrow{\emptyset} E;\overline{F}$ | $E;\underline{F} \xrightarrow{\emptyset} \underline{E;F}$ |
| $\overline{E\Box F} \xrightarrow{\emptyset} \overline{E}\Box F$ | $\overline{E\Box F} \xrightarrow{\emptyset} E\Box\overline{F}$ | $\underline{E}\Box F \xrightarrow{\emptyset} \underline{E\Box F}$ |
| $E\Box\underline{F} \xrightarrow{\emptyset} \underline{E\Box F}$ | $\overline{E\|F} \xrightarrow{\emptyset} \overline{E}\|\overline{F}$ | $\underline{E}\|\underline{F} \xrightarrow{\emptyset} \underline{E\|F}$ |
| $\overline{E[f]} \xrightarrow{\emptyset} \overline{E}[f]$ | $\underline{E}[f] \xrightarrow{\emptyset} \underline{E[f]}$ | $\overline{E\ sy\ a} \xrightarrow{\emptyset} \overline{E}\ sy\ a$ |
| $\underline{E}\ sy\ a \xrightarrow{\emptyset} \underline{E\ sy\ a}$ | $\overline{E\ rs\ a} \xrightarrow{\emptyset} \overline{E}\ rs\ a$ | $\underline{E}\ rs\ a \xrightarrow{\emptyset} \underline{E\ rs\ a}$ |

**Table 1. Inaction rules (I)**

| | |
|---|---|
| $\dfrac{\forall op \in \{;\ ,\ \Box\},\ G \xrightarrow{\emptyset} G'}{op(G,E) \xrightarrow{\emptyset} op(G',E)}$ | $\dfrac{\forall op \in \{;\ ,\ \Box\},\ G \xrightarrow{\emptyset} G'}{op(E,G) \xrightarrow{\emptyset} op(E,G')}$ |
| $\dfrac{G_1 \xrightarrow{\emptyset} G_1'}{G1 \,\|\, G_2 \xrightarrow{\emptyset} G_1' \,\|\, G_2}$ | $\dfrac{G_2 \xrightarrow{\emptyset} G_2'}{G1 \,\|\, G_2 \xrightarrow{\emptyset} G_1 \,\|\, G_2'}$ |
| $\dfrac{\forall op \in \{sy\ ,\ rs\},\ G \xrightarrow{\emptyset} G'}{op(G,a) \xrightarrow{\emptyset} op(G',a)}$ | $\dfrac{G \xrightarrow{\emptyset} G'}{G[f] \xrightarrow{\emptyset} G'[f]}$ |

**Table 2. Inaction rules (II)**

these transitions do not correspond to any dynamic change of state, being considered reversible, which is captured by the introduction of the reverse transition, $\xleftarrow{\emptyset}$. Thus, an equivalence relation is defined:

$$\equiv \quad =_{def} \quad (\xrightarrow{\emptyset} \cup \xleftarrow{\emptyset})^*$$

As a consequence of this equivalence relation all dynamic expressions in the same class behave in the same way.

Inaction rules for sPBC are those defined in Tables 1 and 2. It is noteworthy that in our case the arrow direction becomes more important. It will allow us to define an adequate class of "canonical" dynamic s-expressions that we call *operative dynamic s-expressions*. Since we are going to define an interleaving semantics we have found some technical difficulties when defining the semantics of synchronization. Specifically, we need to infer from the sequences of transitions the bags of stochastic multiactions that could be executed concurrently. By restricting ourselves to operative dynamic s-expressions we are able to achieve that, although this does not imply any restriction, since any class of equivalent expressions contains at least one operative dynamic s-expression.

**Definition 1** Whenever $G$ and $G'$ are in *DynExpr*, such that $G \xrightarrow{\emptyset} G'$, we say that G' is an immediate derivation of $G$. Furthermore, when $G \xrightarrow{\emptyset} G_1 \xrightarrow{\emptyset} G_2 \xrightarrow{\emptyset} \ldots \xrightarrow{\emptyset} G'$, we will say that $G'$ is a *derivation* of $G$. $\square$

**Definition 2** We say that a dynamic s-expression $G$ is *operative* if it has no derivations. We will denote the set of all the operative dynamic s-expressions by *OpDynExpr*. $\square$

**Example 1** Let $G = \overline{<\alpha_1, r_1> \,\Box\, <\alpha_2, r_2>}$ be a dynamic s-expression. Two inaction rules can be applied for $G$, and thus it is not operative. Two operative dynamic s-expressions can be obtained by the application of these rules:

$$G_1 = \overline{<\alpha_1, r_1>} \ \Box\ <\alpha_2, r_2>$$
$$G_2 = <\alpha_1, r_1> \ \Box\ \overline{<\alpha_2, r_2>}$$

$\square$

It is not difficult to see that from any non-operative dynamic s-expression $G$, we can always reach some operative s-expression; but, in general, this will not be unique, as illustrated in the example above. Although the arrow direction in inaction rules has played a special role up to now, this does not mean that the related expressions should no longer be equivalent. We will therefore also consider the reverse arrow $\xleftarrow{\emptyset}$, and the generated equivalence relation $\equiv$.

### 3.2 Rules defining the stochastic transitions

We define the operational semantics by means of rules from which we derive transitions of the form:

$$G \xrightarrow{<\alpha,r>} G' \quad \text{with} \quad <\alpha, r> \in \mathcal{SL}$$

which capture all the possible evolutions of each dynamic s-expression. These correspond to the execution of a multiaction at a time which follows a negative exponential distribution. Thus, each one of these symbolic transitions can be seen as an abstraction of infinitely many possible evolutions of the dynamic s-expression, each one with a different time associated, and a different probability, which can be computed from the negative exponential distribution.

Consequently, by iterating the application of these rules we will obtain sequences of transitions which will correspond to the occurrence sequences of the stochastic Petri net that we will associate to it. This is, therefore, an important difference with respect to PBC and tPBC, because one of their main features is that they support *true concurrency*. In our case, the elapsed time on the execution of each stochastic multiaction follows a continuous distribution (a negative exponential with parameter $r$), so that the probability to execute two or more multiactions at the same time is zero. Therefore, our operational interleaving semantics properly captures the behaviour that we will find later in the stochastic Petri net associated to each process. Note, moreover, that the *memoryless property* of the negative exponential distribution plays an important role in this, because the time that has elapsed until a component evolves by executing a multiaction does not affect the execution time of the remaining components.

| | | | | | |
|---|---|---|---|---|---|
| **(B)** | $\dfrac{}{\overline{<\alpha,r>} \stackrel{<\alpha,r>}{\longrightarrow} \underline{<\alpha,r>}}$ | **(S1)** | $\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G;F \stackrel{<\alpha,r>}{\longrightarrow} G';F}$ | **(S2)** | $\dfrac{H \stackrel{<\alpha,r>}{\longrightarrow} H'}{E;H \stackrel{<\alpha,r>}{\longrightarrow} E;H'}$ |
| **(Rs)** | $\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G\ rs\ a \stackrel{<\alpha,r>}{\longrightarrow} G'\ rs\ a}\ a,\widehat{a}\notin A(\alpha)$ | **(Re)** | $\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G[f] \stackrel{<f(\alpha),r>}{\longrightarrow} G'[f]}$ | **(E1)** | $\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G \Box F \stackrel{<\alpha,r>}{\longrightarrow} G' \Box F}$ |
| **(E2)** | $\dfrac{H \stackrel{<\alpha,r>}{\longrightarrow} H'}{E \Box H \stackrel{<\alpha,r>}{\longrightarrow} E \Box H'}$ | **(C1)** | $\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G \| H \stackrel{<\alpha,r>}{\longrightarrow} G' \| H}$ | **(C2)** | $\dfrac{H \stackrel{<\alpha,r>}{\longrightarrow} H'}{G \| H \stackrel{<\alpha,r>}{\longrightarrow} G \| H'}$ |

**Table 3. Rules defining the stochastic transitions (I)**

Rules defining the stochastic transitions are those presented in Table 3, together with those corresponding to the synchronization operator, which will be described in detail later. We assume that all the dynamic s-expressions that appear on the left-hand sides of each transition in the rules are operative.

### 3.2.1 Synchronization

Since rules C1 and C2 are the only ones applicable to the parallel operator we conclude that we cannot execute two multiactions at the same time, i.e., we have an interleaving semantics. Next, let us see how we deal with the synchronization operator:

Let $G \stackrel{<\alpha,r>}{\longrightarrow} G'$, $H \stackrel{<\beta,s>}{\longrightarrow} H'$ be two stochastic transitions of two operative dynamic s-expressions. According to the rules in Table 3 we can derive the following transition sequences:

$$\begin{array}{l} G\|H \stackrel{<\alpha,r>}{\longrightarrow} G'\|H\ (\stackrel{\emptyset}{\longrightarrow})^*\ J_1^* \stackrel{<\beta,s>}{\longrightarrow} G_1\|H_1 \\ G\|H \stackrel{<\beta,s>}{\longrightarrow} G\|H'\ (\stackrel{\emptyset}{\longrightarrow})^*\ J_2^* \stackrel{<\alpha,r>}{\longrightarrow} G_2\|H_2 \end{array}$$

where $J_1^*, J_2^*$ are operative dynamic s-expressions, and $G_1\|H_1 \equiv G_2\|H_2 \equiv G'\|H'$. As a matter of fact, we can consider that $<\alpha,r>$ and $<\beta,s>$ might be executed concurrently, even though the probability for executing them at the same time without synchronizing them is null. Thus, in order to adequately define the semantics of the synchronization we need to capture all the possible sets of bags of stochastic multiactions that can be executed concurrently for any operative dynamic s-expression.

**Definition 3** We define the function:

$$BC\ :\ \mathit{OpDynExpr} \longrightarrow \mathcal{P}(\mathcal{B}(SL))$$

defining the set of *bags of concurrent stochastic multiactions* of an operative dynamic s-expression in the following way:

- If $G \in \mathit{OpDynExpr}$ is final (i.e. G=$\underline{E}$), we take $BC(G) = \emptyset$.
- If $G \in \mathit{OpDynExpr}$ is not final:
  - $BC(\overline{<\alpha,r>}) = \{\{<\alpha,r>\}\}$.
  - If $\gamma \in BC(G)$, then for every non-empty submultiset $\gamma_i$ of $\gamma$ we have $\gamma_i \in BC(G)$, i.e., $BC(G)$ is closed by inclusion of multisets.
  - If $\gamma \in BC(G)$, then:
    - $\gamma \in BC(G;E)$
    - $\gamma \in BC(E;G)$
    - $\gamma \in BC(G \Box E)$
    - $\gamma \in BC(E \Box G)$
    - $\gamma \in BC(G\ rs\ a)$, if $a, \widehat{a} \notin A(\gamma)$
    - $\gamma \in BC(G\ sy\ a)$
    - $f(\gamma) \in BC(G[f])$
  - If $\gamma_1 \in BC(G)$, $\gamma_2 \in BC(H)$, then $\gamma_1 + \gamma_2 \in BC(G\|H)$.
  - If $\gamma \in BC(G\ sy\ a)$, and $<\alpha,r_1>$, $<\beta,r_2> \in \gamma$, with $a \in A(<\alpha,r_1>)$, and $\widehat{a} \in A(<\beta,r_2>)$, then: $\gamma' \in BC(G\ sy\ a)$, where:

$$\begin{array}{rl} \gamma' = & (\gamma + \{<\alpha,r_1> \oplus_a <\beta,r_2>\}) \setminus \\ & \{<\alpha,r_1>, <\beta,r_2>\} \end{array}$$

$\square$

Note that this definition is correct, because given a static s-expression $E$, and two operative dynamic s-expressions $G, H$, it follows that all of the following dynamic s-expressions are operative: $G;E$, $E;G$, $G \Box E$, $E \Box G$, $G\ rs\ a$, $G\|H$ and $G\ sy\ a$.

**Example 2** In order to illustrate the previous definition for the concrete case of the synchronization let us consider the following operative dynamic s-expression:

$$G = \overline{<\{a\},1>} \| \overline{<\{\hat{a},\hat{a}\},2>} \| \overline{<\{a\},3>}$$

To compute $BC(G\ sy\ a)$ we first need to obtain $BC(G)$:

$$\begin{aligned} BC(G) = \{\ & \{<\{a\},1>,\ <\{\hat{a},\hat{a}\},2>,\ <\{a\},3>\},\\ & \{<\{a\},1>,\ <\{\hat{a},\hat{a}\},2>\},\\ & \{<\{\hat{a},\hat{a}\},2>,\ <\{a\},3>\},\\ & \{<\{a\},1>,\ <\{a\},3>\},\ \{<\{a\},1>\},\\ & \{<\{\hat{a},\hat{a}\},2>\},\ \{<\{a\},3>\}\ \} \end{aligned}$$

We then obtain the new bags in $BC(G\ sy\ a)$ by synchronizing on conjugated pairs of actions $(a,\hat{a})$:

$$\begin{aligned} BC(G\ sy\ a) = BC(G) \cup \{\ & \{<\{\hat{a}\},1>,\ <\{a\},3>\},\\ & \{<\{a\},1>,<\{\hat{a}\},2>\},\ \{<\emptyset,1>\},\\ & \{<\{\hat{a}\},1>\},\ \{<\{\hat{a}\},2>\}\ \} \end{aligned} \qquad \square$$

We may now define the rules for the synchronization operator (Table 4). Rule *Sy1* is quite obvious, it captures the fact that the synchronization preserves the behaviour of $G$, while rule *Sy2* captures the synchronization of two multiactions $<\alpha_1,r_1>$, $<\alpha_2,r_2>$ which are executed consecutively (but drawing up some inaction rules in the meantime in order to get a new operative dynamic s-expression).

**(Sy1)** $$\frac{G \stackrel{<\alpha,r>}{\longrightarrow} H}{G\ sy\ a \stackrel{<\alpha,r>}{\longrightarrow} H\ sy\ a}$$

**(Sy2)** With $\{<\alpha_1,r_1>,<\alpha_2,r_2>\} \in BC(G_1\ sy\ a)$, and $a \in A(<\alpha_1,r_1>), \widehat{a} \in A(<\alpha_2,r_2>)$:

$$\frac{G_1\ sy\ a \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_2\ sy\ a\ (\stackrel{\emptyset}{\longrightarrow})^*\ G_2^*\ sy\ a \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_3\ sy\ a}{G_1\ sy\ a \stackrel{<\alpha_1,r_1>\oplus_a<\alpha_2,r_2>}{\longrightarrow} G_3\ sy\ a}$$

**Table 4. Rules for the operator** *sy*

**Proposition 1** Given an operative dynamic s-expression $G$, $\gamma \in BC(G)$, and any serialization of the stochastic multiactions of $\gamma$: $<\alpha_1,r_1> \ldots <\alpha_n,r_n>$, there exists a transition sequence:

$$\begin{aligned} G \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_1\ (\stackrel{\emptyset}{\longrightarrow})^*\ G_1^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_2\ (\stackrel{\emptyset}{\longrightarrow})^*\ G_2^* \stackrel{<\alpha_3,r_3>}{\longrightarrow} \ldots\\ \ldots (\stackrel{\emptyset}{\longrightarrow})^*\ G_{n-1}^* \stackrel{<\alpha_n,r_n>}{\longrightarrow} G' \end{aligned}$$

Moreover, all dynamic s-expressions $G'$ thus obtained are equivalent with respect to $\equiv$.

**Proof:** By structural induction on the syntax of $G$, the base case (a simple stochastic multiaction) is trivial. A simple application of the induction hypothesis solves the external choice, sequential composition, restriction and relabelling. For the parallel operator the only case requiring some comments is when $\gamma = \gamma_1 + \gamma_2$, with $\gamma_1 \in BC(G_1)$, $\gamma_2 \in BC(G_2)$, but in this case we may apply the induction hypothesis to both of them, to get two sequences of transitions which can be combined in the adequate way, because the evolutions for both components are independent.

The case of the synchronization operator, i.e., when $G = G_1\ sy\ a$, is somewhat more involved, and we need to distinguish the different cases that may occur:

- If $\gamma \in BC(G_1)$, the result is immediate, by applying the induction hypothesis.
- If $\gamma = (\gamma_1 + \{<\alpha_1,r_1> \oplus_a <\alpha_2,r_2>\}) \setminus \{<\alpha_1,r_1>,<\alpha_2,r_2>\}$, with $\gamma_1 \in BC(G)$, and $a \in A(<\alpha_1,r_1>), \widehat{a} \in A(<\alpha_2,r_2>)$, then it could be the case that $\gamma_1 \in BC(G_1)$ or it could also have been obtained in the same way, by joining some multiactions; but this procedure cannot be applied infinitely, and therefore we may apply a new induction on the number of times that we have applied the synchronization of actions. So, our base case is $\gamma_1 \in BC(G_1)$, for which we may apply the hypothesis of the first induction, considering a serialization in which $<\alpha_1,r_1>$ and $<\alpha_2,r_2>$ appear one before the other. Although this sequence starts from $G_1$, the same could be obtained starting from $G_1\ sy\ a$ (rule *Sy1*), and then, by applying rule *Sy2*, we get a sequence of transitions that corresponds to the given serialization of $\gamma$. In fact, it should be noted that the order of the remaining stochastic multiactions in $\gamma_1$ is unimportant, and thus, for any serialization of $\gamma$ we may obtain the corresponding transition sequence.

  For the inductive case, we now have, as hypothesis, the property holding for $\gamma_1 \in BC(G\ sya)$, which allows us to argue along the same lines as in the base case, by considering a serialization of $\gamma_1$. The only difference is that in this case the sequence of rules that we obtain starts directly from $G_1\ sy\ a$, instead of from $G_1$.

$\square$

## 3.3 Operational semantics

In this section we define the transition system that defines the operational semantics of sPBC, whose states are the equivalence classes of dynamic s-expressions, with respect to the $\equiv$ relation.

**Definition 4** Let $G, G', H, H'$ be dynamic s-expressions, such that $G' \equiv G$, $H' \equiv H$, and $G' \stackrel{<\alpha,r>}{\longrightarrow} H'$. Then, we will say that $H$ and $H'$ are both *dynamic immediate derivations* of $G$. Next, considering the class $[G]_\equiv$, we will denote by $act(G)$ the multiset of stochastic multiactions that we may execute in a single step from any of the processes in that class. Thus, $<\alpha,r> \in act(G)$ if and only if there are two dynamic s-expressions $G'$ and $H$ such that $G' \in [G]_\equiv$ and $G' \stackrel{<\alpha,r>}{\longrightarrow} H$. In order to define the number of times that each stochastic multiaction appears in $act(G)$ we consider the essentially different ways in which we can derive such a stochastic transition. For instance, for the following dynamic s-expression $G = \overline{<\alpha,r> \,\square\, <\alpha,r>}$, we have two essentially different ways to obtain $<\alpha,r>$, since to do so we can apply either rule *E1* or *E2*.

But we have used the expression "essentially different" because we must be careful with the use of the synchronization rule *Sy2*, since each application can be carried out in two different ways depending on the order in which the two involved stochastic multiactions are executed. It is clear that in this case both derivations are essentially the same, so that we will only count one of these two ways of getting the transition. For instance, for the dynamic s-expression:

$$G = \overline{(<\{a\}, r> \| <\{\widehat{a}\}, r>)\ sy\ a}$$

we will obtain $<\emptyset, r>$ only once.

Then, we will denote the sum of all the rates of the stochastic multiactions in $act(G)$ by $r(G)$:

$$r(G) =_{def} \sum_{<\alpha_i, r_i> \in act(G)} r_i \cdot n_i$$

where $n_i$ is the number of instances of $<\alpha_i, r_i>$ in $act(G)$. Finally, given two dynamic s-expressions $G, H$, such that:

$$G \equiv G' \stackrel{<\alpha_1, r_1>}{\longrightarrow} G_1 \equiv G_1' \stackrel{<\alpha_2, r_2>}{\longrightarrow} \dots\ G_{n-1} \equiv$$
$$\equiv G_{n-1}' \stackrel{<\alpha_n, r_n>}{\longrightarrow} H' \equiv H$$

we will say that $H$ is a *dynamic s-expression derived* from $G$, and we will denote by $[G\rangle$ the set of all the dynamic s-expressions that can be derived from $[G]_\equiv$. □

**Definition 5** Given a static s-expression E, we say that a dynamic s-expression $G$ is *reachable* from $E$ if $G \in [\overline{E}\rangle$. □

**Definition 6** Let $G$ be a dynamic s-expression; we then define the transition system associated to $G$ by $ts(G) = (V, A, v_0)$, where $V = \{[H]_\equiv \mid H \in [G\rangle\}$ is the set of states, $v_0 = [G]_\equiv$ is the initial state, and $A$ is the multiset of transitions, given by:

$$A = \{([H]_\equiv, <\alpha, r>, [J]_\equiv) \mid H \in [G\rangle \wedge H \stackrel{<\alpha, r>}{\longrightarrow} J\}$$

□

### 3.4 Race policy

The *race policy* governs the dynamic behaviour of the system when two or more stochastic multiactions are simultaneously enabled. This happens in two different cases: either when we have a choice, or when we are in the scope of a parallel operator. When several stochastic multiactions are possible, the fastest one will win the race. Consequently, the stochastic multiaction with the greatest rate (of all that are enabled) can win the race with the largest probability. Therefore, the race policy replaces the non-quantitative non-deterministic selection that we have in the classical process algebras, such as CCS and PBC, by a probabilistic decision, as in Markovian process algebras.

We may compute the probability for each stochastic multiaction to win this race by:

$$P(G, <\alpha_i, r_i>) = r_i \cdot n_i / r(G)$$

where $G$ is the dynamic s-expression from which $<\alpha_i, r_i>$ is executed, and $n_i$ is the number of instances of $<\alpha_i, r_i>$ in $act(G)$ (whichever instance of this multiaction in $act(G)$ could win the race). The *waiting time* in $[G]_\equiv$ follows an exponential distribution with parameter $r(G)$ (the minimum of a set of exponential distributions).

When a stochastic multiaction has been executed for an operative dynamic s-expression, for the other ones that remain active, and due to the *memoryless property* of the exponential distribution, we do not need to take care of the time elapsed, because the time that must now elapse until their execution follows the same distribution as before. Notice that in the case of a choice, the stochastic multiactions that were enabled in the non-selected branch are not enabled any more, since inaction transitions cannot be undone after the execution of a stochastic transition.

**Example 3** Let us consider the following static s-expression:

$$E = F\ sy\ a\,, \quad \text{with}$$
$$F = (<\{a\}, r_1> \ \|\ <\{\widehat{a}, \widehat{a}\}, r_2>) \| <\{a\}, r_3>$$

In Fig.1 we can see the transition system associated to $\overline{E}$.

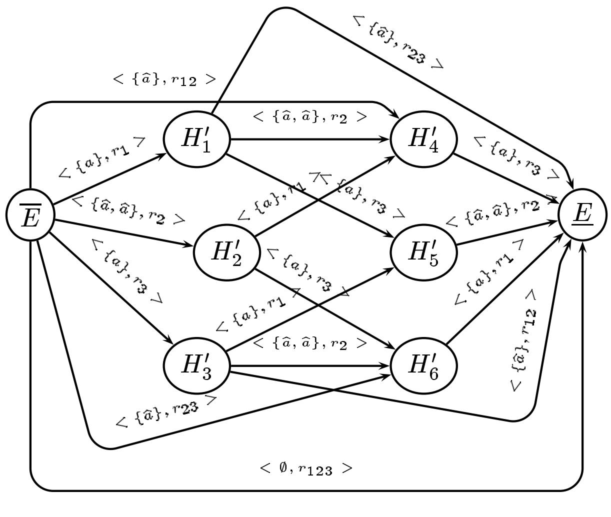


**Figure 1. Transition System for $E$**

We see that $\overline{E}$ can make all the transitions of $\overline{F}$, but with different probabilities.This is because it can make some additional transitions, which become from the following possible synchronizations:

$$\begin{aligned}\gamma_1 &= \{< \{\widehat{a}\}, r_{12} >, < \{a\}, r_3 >\} \\ \gamma_2 &= \{< \{\widehat{a}\}, r_{23} >, < \{a\}, r_1 >\} \\ \gamma_3 &= \{< \emptyset, r_{123} >\}\end{aligned}$$

with $r_{12} = min(r_1, r_2)$, $r_{23} = min(r_2, r_3)$, and $r_{123} = min(r_1, r_2, r_3)$. We may compute the probability with which we reach $\underline{E}$ by following each of the possible paths along that graph. For instance, for the path:

$$< \{a\}, r_1 > . < \{\widehat{a}, \widehat{a}\}, r_2 > . < \{a\}, r_3 >$$

the corresponding probability is:

$$\frac{r_1}{r_1 + r_2 + r_3 + r_{12} + r_{23} + r_{123}} \cdot \frac{r_2}{r_2 + r_3 + r_{23}}$$

while for the path $< \{\widehat{a}\}, r_{12} > . < \{a\}, r_3 >$, the probability is:

$$\frac{r_{12}}{r_1 + r_2 + r_3 + r_{12} + r_{23} + r_{123}}$$

$\square$

### 3.5 Stochastic processes generated from the operational semantics

Once we have the transition system of $\overline{E}$ we can generate the corresponding CTMC. We start with the multigraph $ts(\overline{E})$, and we modify it by combining into a single edge all the edges connecting the same pair of nodes. These new edges will be labelled by the sum of the rates of the combined edges.

The stochastic process associated to the evolution of each dynamic s-expression $\overline{E}$ is a Markovian process, in particular, a Continuous Time Markov Chain (CTMC), which can be obtained by the application of the following proposition.

**Proposition 2** Given a static s-expression $E$, we define the stochastic process $X(t)$, by $X(t) = [G]_{\equiv}$ for each possible evolution in time $t$ of the dynamic s-expression $\overline{E}$ in a dynamic s-expression in $[G]_{\equiv}$, with $G \in [\,\overline{E}\,\rangle$. Then, $X(t)$ is a Markovian process.

**Proof:** Immediate, since we are using a negative exponential distribution and we are applying the race policy. $\square$

**Example 4** For $E_1 = (< \{a\}, r_1 >; < \{b\}, r_2 >) \square < \{c\}, r_3 >$, and $E_2 = < \{a\}, r_1 >; (< \{b\}, r_2 > \square < \{c\}, r_3 >)$, the corresponding CTMCs are those shown in Fig.2.

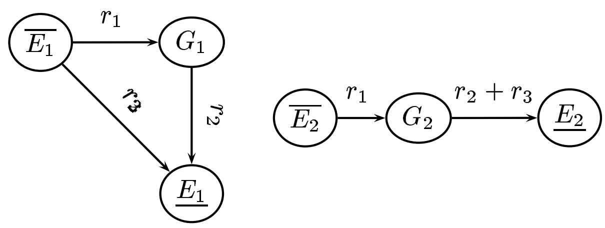


**Figure 2. CTMCs for $E_1$ and $E_2$**

$\square$

## 4 Denotational Semantics

In this section we present a denotational semantics for s-expressions, which is obtained taking Stochastic Petri nets as plain boxes [5, 6, 9]. Therefore, the semantic objects that we use will be called stochastic Petri boxes or just *s-boxes*. Thus, these *s-boxes* are essentially SPNs, but they have the same structure as Petri boxes.

### 4.1 Petri boxes

Boxes of PBC are labelled Petri nets fulfilling some restrictions. Concretely, they are labelled Petri nets $\Sigma = (S, T, W, \lambda)$, where $(S, T, W)$ is a Petri net, and $\lambda$ is a labelling function, which labels places with values from $\{e, i, x\}$, representing *entry places*, *internal places*, and *exit places* respectively; and transitions with elements in $\mathcal{B}(\mathcal{L}) \times \mathcal{L}$. By convention, ${}^\circ\Sigma$ and $\Sigma^\circ$ will denote the set of *e-labelled* places and the set of *x-labelled* places, respectively. Given a place $s \in S$, we will denote by ${}^\bullet s$ ($s^\bullet$) the set of input (output) transitions of $s$ (called preconditions and postconditions of $s$ respectively). A similar notation is used for preconditions and postconditions of transitions. Both can be easily extended to sets of places and sets of transitions.

Then, boxes are defined to be labelled nets such that the following conditions hold:

- ${}^\circ\Sigma \neq \emptyset \neq \Sigma^\circ$
- ${}^\bullet({}^\circ\Sigma) = \emptyset = (\Sigma^\circ)^\bullet$
- $\forall t \in T : {}^\bullet t \neq \emptyset \neq t^\bullet \;\wedge\; {}^\bullet t \cap t^\bullet = \emptyset$

$\lambda(t)$ is a relation which associates elements of $\mathcal{L}$ to bags of multiactions. A box is said to be *plain* when for every $t \in T$ $\lambda(t)$ is a constant relation, i.e., an element of $\mathcal{L}$.

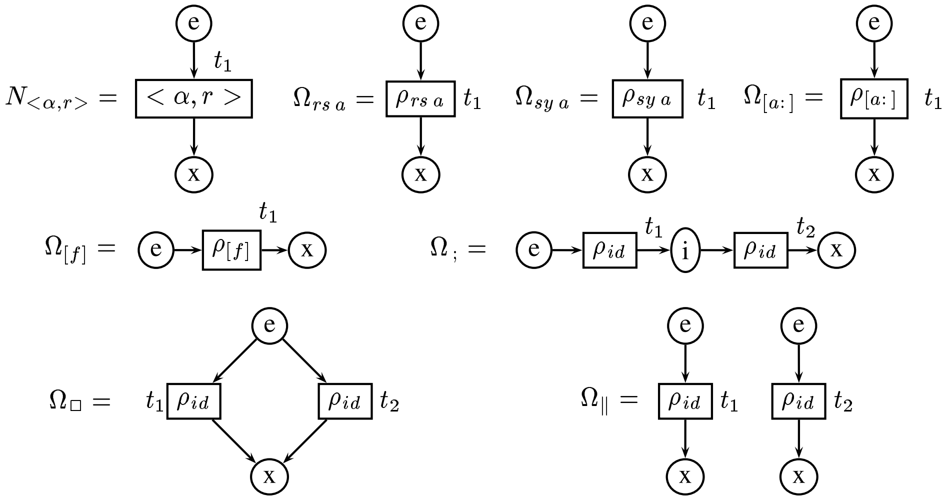


**Figure 3. A plain s-box $N_{<\alpha,r>}$, and the operator s-boxes for finite sPBC**

**Definition 7** A plain stochastic Petri box (or just *plain s-box*) is a tuple $\Sigma = (S, T, W, \lambda, \mu)$, where $(S, T, W, \lambda)$ is a plain box, and $\mu : T \longrightarrow \mathbb{R}^+$ is a stochastic function, which associates a rate to every transition. $\Box$

A plain s-box can be either marked or not. We will denote by $M_e$ the marking in which only the *entry places* are marked (each one with a single token); on the other hand, $M_x$ will denote the marking in which only the *exit places* are marked, each one also with a single token. Thus, a plain s-box is essentially a special kind of labelled Stochastic Petri net, whose behaviour follows the classical *firing rule* of SPNs. As usual, by using the underlying CTMC of these s-boxes we can develop performance evaluation studies. Since this denotational semantics is equivalent to our former operational semantics, the CTMC obtained from the net model will be equivalent to those that we can construct directly from the transition system of dynamic s-expressions.

## 4.2 Algebra of s-boxes

In order to define the algebra of s-boxes we introduce in Fig.3 a collection of *operators on s-boxes*, on the basis of the classical operator boxes of PBC. Relabelling relations $\rho_{op} \subseteq \mathcal{B}(\mathcal{SL}) \times \mathcal{SL}$ appearing in that Figure are defined as follows:

- $\rho_{id} = \{(\{< \alpha, r >\}, < \alpha, r >) \mid\ < \alpha, r > \in \mathcal{SL}\}$
- $\rho_{[f]} = \{(\{< \alpha, r >\}, < f(\alpha), r >) \mid\ < \alpha, r > \in \mathcal{SL}\}$
- $\rho_{rs\,a} = \{(\{< \alpha, r >\}, < \alpha, r >) \mid\ < \alpha, r > \in \mathcal{SL}\ \wedge$ $a, \widehat{a} \notin A(\alpha)\}$
- $\rho_{sy\,a}$ is the least relabelling relation containing $\rho_{id}$, such that: $(\Gamma, < \alpha + \{a\}, r_1 >) \in \rho_{sy\,a}\ \wedge$ $(\Delta, < \beta + \{\widehat{a}\}, r_2 >) \in \rho_{sy\,a} \quad \Rightarrow$ $(\Gamma + \Delta, < \alpha + \beta, min(r_1, r_2) >) \in \rho_{sy\,a}$
- $\rho_{[a:\,]} = \{(\Gamma, < \alpha, r >) \in \rho_{sy\,a} \mid\ a, \widehat{a} \notin A(\alpha)\}$

As we are using the same mechanism to that followed for PBC in order to obtain the plain boxes, by combining both refinement and relabelling, and since we are using plain s-boxes with the same structure, we will also obtain safe and clean plain s-boxes as the output of the application of that mechanism. The denotational semantics for s-expressions can then be formally defined by the following homomorphism:

$$Box_s(< \alpha, r >) = N_{<\alpha,r>}$$
$$Box_s(OP(E_1, \ldots, E_n)) = \Omega_{OP}(Box_s(E_1), \ldots, Box_s(E_n))$$

And, as usual, we take $M_e$ as the initial marking for the plain s-box thus generated.

Next we show that given a static s-expression $E$, the operational semantics for $\overline{E}$ and the semantics of the corresponding plain s-box are isomorphic.

**Theorem 1** For any static s-expression $E$, the transition system $ts(\overline{E})$ associated to $\overline{E}$, and the reachability graph of the marked SPN $(Box_s(E), M_e)$ are isomorphic.

**Proof:** It is clear that at the functional level we have the same isomorphism as in PBC, once we take an interleaving semantics both in the algebra and for s-boxes. Now, since in both semantics conflicts are solved by applying the race policy, and as the stochastic multiactions obtained in the algebra and the corresponding transitions in the plain s-box are labelled with the same rate, both behave in exactly the same way. $\Box$

## 5 Conclusions and Future Work

We have introduced a stochastic extension of finite PBC (sPBC), which preserves the main features of PBC. Thus, we have defined both an operational and a box semantics for sPBC, and we have seen that they are equivalent. An important difference with respect to PBC is that now an interleaving semantics has been considered in both cases, although we still keep parallelism at the low level of multiactions. This is a natural consequence of the fact that in both semantics we are using continuous distributions (in particular, negative exponential distributions). This has also played an important role when defining the behaviour of the *synchronization* operator. Another remarkable difference is that in sPBC non-determinism that appears in PBC is replaced by a probabilistic branching, which is solved by using the race policy.

Our immediate future work will focus on the extensions of our results to the infinite case, by including both iteration and recursion. Furthermore, we also intend to extend the model with additional capabilities, such as urgent multiactions, priorities, and so on.

## References


[1] M. Ajmone Marsan. *Stochastic Petri Nets: An Elementary Introduction*. Lecture Notes in Computer Science; Advances in Petri Nets 1989, 424:1–29, 1990.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[3] M. Ajmone Marsan, A. Bobbio, and S. Donatelli. *Petri nets in performance analysis: An introduction.* Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models, 1491, 1998.

[4] M. Bernando, R. Gorrieri, and L. Donatello. *MPA : A Stochastic Process Algebra*. Technical report, Laboratory of Computer Science, University of Bologna, 1994.

[5] E. Best, R. Devillers, and M. Koutny. *Petri Nets, Process Algebras and Concurrent Programming Languages*, Reisig, W. and Rozenberg, G. (eds). Lecture Notes in Computer Science: Lecture on Petri Nets II: Applications, 1492, 1998.

[6] E. Best, R. Devillers, and M. Koutny. *A Consistent Model for Nets and Process Algebras*, to appear as a chapter in the "The Handbook on Process algebras" (Bergstra, J.A. and Ponse, A. and Smolka, S.S.,eds), North Holland, 2000.

[7] E. Best, R. Devillers, and M. Koutny. *Petri Net Algebra*, EATCS, Springer. 2000.

[8] E. Best, R. Devillllers, and J. Hall. *The Box Calculus : A New Causal Algebra with Multi-label Communication*. Advances in Petri Nets. Springer, Lecture Notes in Computer Science, 609:21–69, 1992.

[9] E. Best and M. Koutny. *A Refined View of the Box Algebra*. Proc. Application and Theory of Petri Nets. Springer, Lecture Notes in Computer Science, 935:1–20, 1995.

[10] N. Götz, U. Herzog, and M. Rettelbach. *TIPP: A Language for Timed Processes and Performance Evaluation*. Technical report, University of Erlangen-Nürnberg, Germany, 1992.

[11] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, Dep. of Computer Science, University of Edinburgh, April 1994.

[12] J. Hillston. *The nature of the synchronization*. Int. Workshop on Process Algebra and Performance Modelling, 1994.

[13] M. Koutny. *A Compositional Model of Time Petri Nets*. International Conference on Theory and Application of Petri Nets, 2000, Lecture Notes in Computer Science no. 1825, pages 303–322, 2000.

[14] O. Marroquín and D. de Frutos. *Extending the Petri Box Calculus with Time*. To appear in Proc. Int. Conf. on Theory and Application of Petri Nets, LNCS. 2001.

[15] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[16] M. Ribaudo. *Stochastic Petri Net Semantics for Stochastic Process Algebra*. Proc. 6th. Int. Workshop on Petri Nets and Performance Models, Durham, 1995.