Extending PBC with Markovian multiactions *

Hermenegilda Macià Soler
Univ. Castilla-La Mancha, Dpto. Matemáticas,
Albacete, Spain, 02071
e-mail: hmacia@pol-ab.uclm.es

and

Valentín Valero Ruiz
Univ. Castilla-La Mancha, Dpto. Informática,
Albacete, Spain, 02071
e-mail: valentin@info-ab.uclm.es

and

David de Frutos Escrig
Univ. Complutense, Dpto. Sist. Inf. y Programación,
Madrid, Spain, 02071
e-mail: defrutos@eucmvx.sim.ucm.es

and

Fernando Cuartero Gómez
Univ. Castilla-La Mancha, Dpto. Informática,
Albacete, Spain, 02071
e-mail: fernando@info-ab.uclm.es

ABSTRACT

In this paper we present a stochastic extension of finite PBC, sPBC, in which actions have associated a stochastic delay, taken from a negative exponential distribution. We define for the language both an operational semantics and a denotational one, using a kind of labelled stochastic Petri nets 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: Distributed Systems and Parallelism, Performance Evaluation, Stochastic Petri Nets, Stochastic Process Algebra, PBC.

RESUMEN

En este artículo presentamos una extensión estocástica del PBC finito, sPBC, en el cual cada acción tiene asociada una duración estocástica, proveniente de una distribución exponencial negativa. Para el mismo lenguaje definimos una semántica operacional y otra denotacional, para esta última utilizando una clase especial de redes de Petri estocásticas y etiquetadas. En este artículo nos centramos en el sPBC finito, esto es, sin considerar el operador iterativo ni la definición recursiva de procesos.
Palabras clave: Sistemas distribuidos y paralelismo, Evaluación de desempeño, Redes de Petri estocásticas, Álgebras de Procesos estocásticas, PBC.

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

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, like CCS [15], where the synchronization is embedded into 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 Nets, called *boxes*. Two extensions of the language including time have been recently defined, like 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 durations, thus obtaining a stochastic (Markovian) extension of the PBC called sPBC.

Therefore, in sPBC each multiaction will have an associated random duration which follows a negative exponential distribution. So, a (basic) stochastic multiaction will be represented by a pair $<\alpha, r>$, where α represents a (classical) multiaction of PBC and $r \in \mathbb{R}_*^+$ is the parameter of the associated exponential distribution. Thus, if we denote by T the random variable that describes the duration of $<\alpha, r>$ we have that $P\{T \leq t\} = 1 - e^{-rt}$ and $E(T) = 1/r$.

Thus, 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], but with the main features of (finite) PBC, such as the separation of synchronization and parallelism.

Moreover, we have extended the plain boxes of PBC to obtain a stochastic version, on the basis of Stochastic Petri Nets [1, 2, 3]. Next, let us observe the main consequence of the introduction of a continuous distribution (a negative exponential one) to measure the duration of multiactions: we have that 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 probability zero to be executed in parallel (even if they are simultaneously enabled). As a consequence, we get a natural *interleaving semantics*, in opposition to the *true concurrency* that exhibits PBC. Even so, we still have a certain degree of parallelism at the low level, because we work with multiactions that are multisets of actions, like in PBC.

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 synchronisation 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 nice and detailed discussion of the 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 one of those involved in the synchronization, which at the intuitive level can be interpreted as taking as duration of the new multiaction (capturing the synchronization) the duration of the slower one of those involved in the synchronization. Moreover, in case of conflicts we adopt the *race policy*, i.e., whenever two or more stochastic multiaction are possible, the faster one is executed, in the same way as this *race policy* governs the dynamic behaviour of Stochastic Petri Nets.

With the use of the negative exponential distribution we have some noteworthy additional advantages, like the consequences that derive from the memoryless property of this distribution. An important fact that we may conclude 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 nice translation of sPBC-expressions into a class of labelled Stochastic Petri Nets, called s-boxes. Finally, we present in this paper 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 SPN's.

The paper is structured as follows: in Section 2 we present the syntax of finite sPBC-expressions, the operational semantics 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 section we define the syntax of finite sPBC, i.e., without considering the iteration and recursion operators.

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

Notation

From now onwards 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 $\widehat{\widehat{a}} = a$, like 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 non-empty 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 α. In particular, we consider that $\{\emptyset\} \in \mathcal{L}$, which is used to specify the execution of one action 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 $f : \mathcal{A} \to \mathcal{A},$ 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*:
    $\{< \alpha_1, r_1 >, \ldots, < \alpha_n, r_n >\} + \{< \beta_1, s_1 >, \ldots, < \beta_n, s_n >\} =_{def}$
    $\{< \alpha_1, r_1 >, \ldots, < \alpha_n, r_n >, < \beta_1, s_1 >, \ldots, < \beta_n, s_n >\}$
- Synchronization
  - *of multiactions*: $\alpha \oplus_a \beta =_{def} \gamma$ where:
    $\gamma(b) =_{def} \begin{cases} \alpha(b) + \beta(b) - 1 & \text{if } b = a \vee b = \widehat{a} \\ \alpha(b) + \beta(b) & \text{otherwise} \end{cases}$
    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:*
    $f(\{<\alpha_1, r_1>, \ldots, <\alpha_n, r_n>\}) =_{def} \{<f(\alpha_1), r_1>, \ldots, <f(\alpha_n), r_n>\}$

Static s-Expressions

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

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

where $[a : E]$ is a derivated operator, which is defined by $[a : E] = (E \,sy\, a) \,rs\, a$. 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 is executed.

These static s-expressions are used to describe the structural behaviour of concurrent systems, while the stochastic information associated to every multiaction defines the time required to complete its execution; besides, by means of this stochastic information we will define the probability with which this action will be executed in case of conflict, as we will detail below. As usual, we will denote static s-expressions by the letters: E, F, $E_i, \ldots$.

Dynamic s-Expressions

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

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

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 their 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, ...

3. OPERATIONAL SEMANTICS

The operational semantics of sPBC is defined in a very similar way as in [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 significative differences with the (stochastic) multiaction transition rules, the most important one being that now an interleaving semantics is considered, whose interpretation is described later.

Inaction Rules and Structural Equivalence

In PBC any dynamic s-expression in the same equivalence class with respect to $\equiv$ can make the same steps, i.e., all these dynamic s-expressions behave exactly in the same way. However, when considering stochastic multiactions we have found some technical difficulties following this idea, and thus we have decided to consider a kind of dynamic s-expressions, which we call *operative*, which will be those able to execute stochastic multiactions. Therefore, now the inaction rules will be used to reach the corresponding *operative* dynamic s-expression, from which the process may evolve by executing a multiaction. These inaction rules are defined in Tables 1 and 2; note that, obviously, no time elapses with their execution.

Definition 1 Whenever G and G' are two dynamic s-expressions, such that $G \xrightarrow{\emptyset} G'$, we say that G' is an *immediate derivation.*

Besides, when $G \xrightarrow{\emptyset} G_1 \xrightarrow{\emptyset} G_2 \xrightarrow{\emptyset} \ldots \xrightarrow{\emptyset} G'$, we will say that G' is a *derivate* of G. □

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

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, so that it is not operative. The dynamic s-expressions obtained by means of these rules are the following:

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

which are both operative, because no inaction rule can be applied for them. □

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

Table 1: Inaction rules (I)

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

Table 2: Inaction rules (II)

It is obvious from the rules in Tables 1 and 2 that from any non-operative dynamic s-expression G, we can reach always an operative s-expression; but, in general this will not be unique, i.e., we may reach several operative dynamic s-expressions starting from a non-operative one, as we have seen in the previous example. Nevertheless, as in classical PBC [5, 13] all the dynamic s-expressions that can be obtained by applying the inaction rules, and also those obtained by reversing the rules are considered to be equivalent too. This is captured by the following equivalence relation:

Definition 3 We define the structural equivalence relation for dynamic s-expressions, $\equiv =_{def} (\xrightarrow{\emptyset} \cup \xleftarrow{\emptyset})^*$. By $[G]_\equiv$ we denote the class of G with respect to $\equiv$.

$\Box$

Rules Defining the Stochastic Transitions

We define the operational semantics by means of rules with 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. They 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.

As a consequence, 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. Thus, this is an important difference with respect to PBC and tPBC, because one of the main features of them 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 we have 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. Besides, let us note that the *memoryless property* of the negative exponential distribution plays an important role in this fact, because the time that has elapsed until a component evolves by executing a multiaction does not affect to the execution time of the remaining components.

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.

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

Table 3: Rules defining the stochastic transitions (I)

Synchronization

Since rules C1 and C2 are the only applicable ones to the parallel operator we conclude that we cannot execute two multiactions at the same time, i.e., we have an interleaving semantics, although the overlapped execution of actions is captured at the low level by the consideration of multiactions. Next, let us see how we deal with the synchronization operator:

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

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

where J_1^*, J_2^* are operative dynamic s-expressions, and $G_1\|H_1 \equiv G_2\|H_2 \equiv G'\|H'$. Then, we can consider that $<\alpha,r>$ and $<\beta,s>$ might be executed concurrently, even if the probability to execute them at the same time without synchronizing them was null. In order to adequately define the semantics of the synchronization we need some construction to join two multiactions in a single one.

Definition 4 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 γ_i of γ we have that $\gamma_i \in BC(G)$, 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 $\gamma_1 \in BC(G)$, and $\gamma_2 \in BC(H)$, 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
$\gamma' = (\gamma + \{<\alpha, r_1> \oplus_a <\beta, r_2>\}) \setminus \{<\alpha, r_1>, <\beta, r_2>\}$

$\Box$

Let us note that the previous definition is correct, because given a static s-expression E, and two operative dynamic s-expressions G, H, we have 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 $G\, sy\, a$.

Using this definition we may 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 making in the meantime some inaction rules in order to get a new operative dynamic s-expression).

(Sy1) $\dfrac{G \overset{<\alpha,r>}{\longrightarrow} H}{G\, sy\, a \overset{<\alpha,r>}{\longrightarrow} H\, sy\, a}$

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

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

Table 4: Rules for the operator *sy*

Proposition 1 Given an operative dynamic s-expression G and $\gamma \in BC(G)$, we have that for any serialization of the stochastic multiactions of γ: $<\alpha_1, r_1> \ldots <\alpha_n, r_n>$ there is a sequence of transitions:

$$G \overset{<\alpha_1,r_1>}{\longrightarrow} G_1\, (\overset{\emptyset}{\longrightarrow})^* G_1^* \overset{<\alpha_2,r_2>}{\longrightarrow} G_2\, (\overset{\emptyset}{\longrightarrow})^* G_2^* \overset{<\alpha_3,r_3>}{\longrightarrow} \ldots \overset{<\alpha_n,r_n>}{\longrightarrow} G'$$

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

$\Box$

Operational Semantics

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

Definition 5 Let G, G', H, H' be dynamic s-expressions, such that $G' \equiv G$, $H' \equiv H$, and $G' \overset{<\alpha,r>}{\longrightarrow} H'$. Then, we will say that H is a *dynamic immediate derivate* of G. Then, 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, we have that $<\alpha, r> \in act(G)$ if and only if there is some dynamic s-expression $G' \in [G]_\equiv$ such that $G' \overset{<\alpha,r>}{\longrightarrow} H$. 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 $G = \overline{<\alpha, r> \Box <\alpha, r>}$, we have two essentially different ways to obtain $<\alpha, r>$, since to do it we can apply either rule *E1* or *E2*.

But we had said “essentially different” since we must take care of the use of the synchronization rule *Sy2*, because each time it can be applied we can do it 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, and so we will only count one of these two ways of getting the transition. For instance, for the following dynamic s-expression $\overline{E} = \overline{(<\{a\},r> \,\|\, <\{\hat{a}\},r>)\; sy\, a}$, we will obtain $<\emptyset,r>$ only once.

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

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

where n_i is the number of instances of $<\alpha_i,r_i>$ in $act(G)$. Besides, by $reach(G)$ we will denote the set of dynamic s-expressions derivated from the class $[G]_\equiv$ in a single step. Finally, given two dynamic s-expressions G, H, such that:

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

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

Definition 6 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 7 Let G be a dynamic s-expression, we 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: $A = \{\, (\, [H]_\equiv,\ <\alpha,r>,\ [J]_\equiv\,) \mid H \in [\,G\,\rangle\ \wedge\ <\alpha,r> \in act(H)\ \wedge J$ is an immediate of H $\}$

□

Race Policy

The *race policy* governs the dynamic behaviour of the system when two or more stochastic multiactions are simultaneously enabled. In this way, we will have that several stochastic multiactions are possible, but the faster one will have a greater probability to be executed the first. This corresponds to that with the maximum rate in terms of the exponential distribution. In consequence, 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 $act(G)$. The *waiting time* in $[G]_\equiv$ follows a negative exponential distribution with parameter $r(G)$ (it is the minimum of a set of negative exponential distributions).

Once a stochastic multiaction has been executed, the other ones that were active remain active, and due to the *memoryless property* of the exponential distribution we do not need to take care of the time that has elapsed, because the time that now must elapse until their execution follows the same negative exponential distribution than before.

Example 2 Let us consider the following static s-expression:

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

In Fig.1 we may see the transition system associated to $\overline{E}$. 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 posible synchronizations:

$$\begin{array}{l}
\gamma_1 = \{<\{\hat{a}\},r_{12}>\,,\ <\{a\},r_3>\} \text{ with } r_{12} = min(r_1,r_2) \\
\gamma_2 = \{<\{\hat{a}\},r_{23}>\,,\ <\{a\},r_1>\} \text{ with } r_{23} = min(r_2,r_3) \\
\gamma_3 = \{<\{\emptyset\},r_{123}>\} \text{ with } r_{123} = min(r_1,r_2,r_3)
\end{array}$$

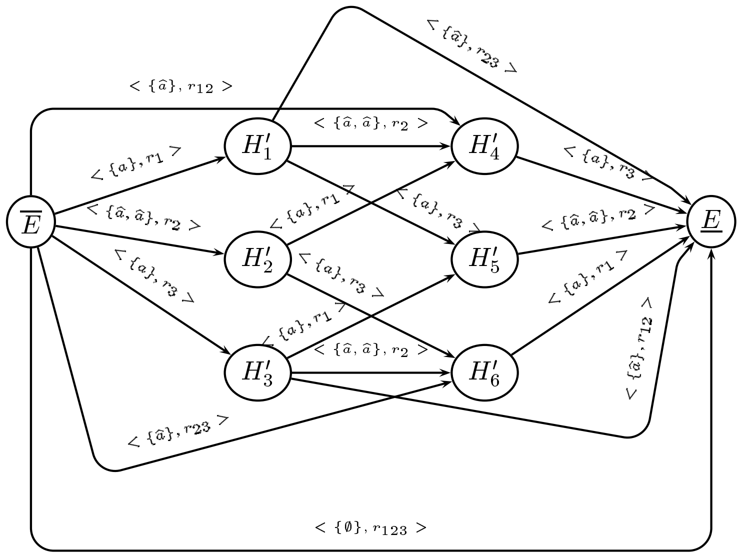

Figure 1: Transition System for E

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:

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

□

Stochastic Processes Generated from the Operational Semantics

As usual, once we have the transition system of $\overline{E}$ we may generate the corresponding CTMC. For it, we start with the multigraph $ts_{\overline{E}}$, and 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 $\overline{E}$ into a dynamic s-expression in $[G]_{\equiv}$, with $G \in [\,\overline{E}\,\rangle$. Then we have that $X(t)$ is a Markovian process.

□

Example 3 Given the static s-expressions E_1 and E_2, with $E_1 = (< \{a\}, r_1 >; < \{b\}, r_2 >) \Box < \{c\}, r_3 >$, and $E_2 = < \{a\}, r_1 >; (< \{b\}, r_2 > \Box < \{c\}, r_3 >)$, the corresponding CTMC's are those in Fig.2.

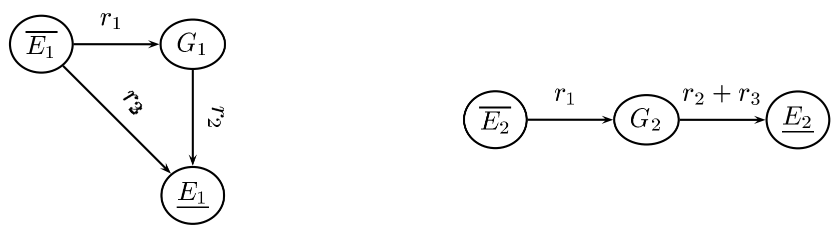

Figure 2: CTMC's for E_1 and E_2

$\Box$

4. DENOTATIONAL SEMANTICS

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

Definition 8 A plain stochastic Petri box (or just *plain s-box*) is a tuple $\Sigma = (S, T, W, \lambda, \mu)$, where (S, T, W, λ) is a classical 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 with a single token, too. Thus, a plain s-box is essentially a special kind of labelled Stochastic Petri nets, whose behaviour follows the classical *firing rule* of SPN's. As usual, 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, these CTMC obtained from the net model will be equivalent to the ones that we can construct directly from the transition system of dynamic s-expressions.

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.

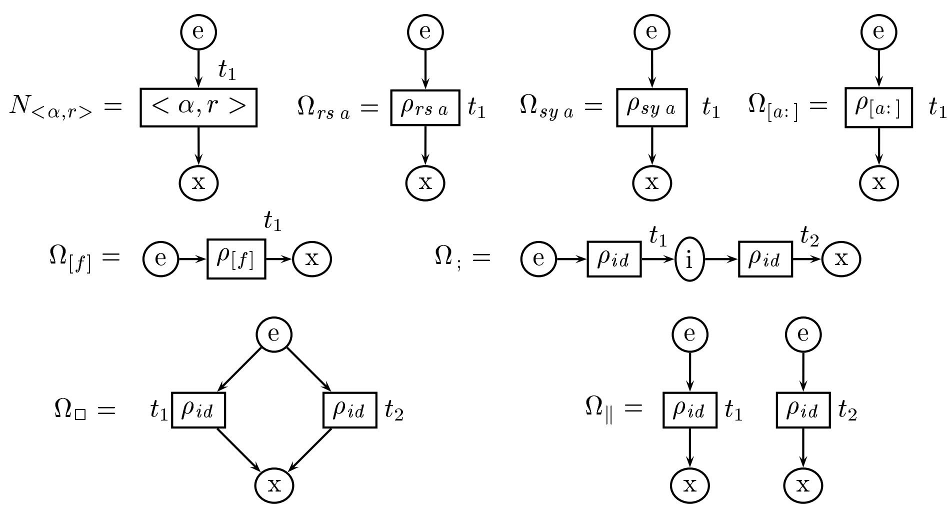

Figure 3: A plain s-box $N_{<\alpha,r>}$, and the operator s-boxes for finite sPBC

There each relabelling relation $\rho_{op} \subseteq \mathcal{B}(\mathcal{SL}) \times \mathcal{SL}$ is 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 ρ_{id}, such that
  $(\Gamma,<\alpha+\{a\},r_1>) \in \rho_{sy\,a} \wedge (\Delta,<\beta+\{\widehat{a}\},r_2>) \in \rho_{sy\,a} \quad \Rightarrow$
  $(\Gamma+\Delta,<\alpha+\beta,r_{12}>) \in \rho_{sy\,a}$, with $r_{12} = min(r_1,r_2)$
- $\rho_{[a:\,]} = \{(\Gamma,<\alpha,r>) \in \rho_{sy\,a} \mid a,\widehat{a} \notin A(\alpha)\}$

Since we are following the same mechanism of PBC for obtaining 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 output of the application of that mechanism. Then, the denotational semantics for s-expressions can 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 equivalent, in the exact sense expressed by the following theorem.

Theorem 1 Given a static s-expression E, and a trace of stochastic multiactions obtained from its transition system: $<\alpha_1,r_1> \ldots <\alpha_n,r_n>$, there is an occurrence sequence $M_e[\sigma\rangle M'$ of $Box_s(E)$, with $\sigma = t_1 \ldots t_n$, where $\lambda(t_k) = \alpha_k$, and $\mu(t_k) = r_k$, $k = 1,\ldots,n$.

Conversely, for each occurrence sequence $M_e[\sigma\rangle M'$ of $Box_s(E)$ with $\sigma = t_1 \ldots t_n$, there is a trace of stochastic multiactions that can be obtained from the transition system of $\overline{E}$: $<\alpha_1,r_1> \ldots <\alpha_n,r_n>$, with $\lambda(t_k) = \alpha_k$ and $\mu(t_k) = r_k$, $k = 1,\ldots,n$.

$\square$

5. CONCLUSIONS AND FUTURE WORK

We have introduced a stochastic extension of finite PBC (sPBC), which keeps 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.

[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. *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.

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

[7] E. Best, R. Devillers, and M. Koutny. *Petri Net Algebra.* EATC, Springer, 2001.

[8] E. Best, R. Devilllers, and J. Hall. *The Box Calculus : A New Causal Algebra with Multi-label Communication.* Advances in Petri Nets. Springer, LNCS, 609: 21-69.

[9] E. Best and M. Koutny. *A Refined View of the Box Algebra.* Proc. Application and Theory of Petri Nets. Springer, LNCS, 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.