Draft version: Introducing the iteration in sPBC *

Hermenegilda Macià, Valentín Valero, Diego Cazorla, and Fernando Cuartero

Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha, 02071 Albacete, SPAIN
{Hermenegilda.Macia,Valentin.Valero, Diego.Cazorla
Fernando.Cuartero}@uclm.es

Abstract. The main goal of this paper is to extend sPBC with the iteration operator, providing an operational semantics for the language, as well as a denotational semantics, which is based on stochastic Petri nets. With this new operator we can model some repetitive behaviours, and then we obtain a formal method that can be easily used for the design of communication protocols and distributed systems.

Keywords: Stochastic Petri Nets, Stochastic Process Algebra, Performance Evaluation, Petri Box Calculus.

1 Introduction

Petri Box Calculus (PBC) [4, 3, 5–7] is an algebraic model for the specification of concurrent systems which has a natural and compositional translation into a special kind of labelled Petri nets, called *boxes*. The description of a wider class of systems, such as real-time systems and fault-tolerance systems, is the goal of two timed extensions of PBC that we may find in the literature, namely tPBC [11] and TPBC [15], both of them considering a deterministic model of time. In the same line we presented sPBC in [14, 13], which is a Markovian extension of PBC.

In sPBC we consider that each multiaction has associated a delay which follows a Markovian distribution, as the transition delays of stochastic Petri Nets (SPNs) [1]. Thus, a stochastic multiaction of sPBC is 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. Moreover, as in SPNs, the *race policy* governs the dynamic behaviour.

In the literature we may find some different approaches that deal with stochastic extensions of process algebras such as PEPA [10], TIPP [8] and EMPA [2]. There are some important differences with respect to them. In sPBC we allow

* This work has been supported by the MCyT project "Description and Performance of Distributed Systems and Application to Multimedia Systems, Ref. TIC2003-07848-c02-02"

multiactions (multisets of actions), we consider a synchronization operator totally independent from the parallel operator (as in PBC) and, finally, we obtain the parameter of the new multiaction generated after synchronization following a different technique (see [9] for a complete discussion about all the different alternatives for that). The technique that we use is based on *conflict rates*, which are inspired in the *apparent rates* of PEPA [10]. Nevertheless, in our approach we can find an important advantage: we have been able to obtain a static translation into stochastic Petri nets, while in PEPA rates of transitions are in some cases marking dependent [17].

In our two previous works we only considered finite sPBC [14, 13]. Then the aim of this paper is twofold: including in the syntax the iteration operator and proving that the operational and the denotational semantics (the latter based on a special kind of labelled SPNs, called *s-boxes*) are fully abstract. The iteration operator allows us to describe infinite behaviours and, therefore, it is a powerful tool to describe the behaviour of concurrent systems. Then, with this formalism we can deal with the design of communication protocols and distributed systems with a Markov time, but joining in a single model both the advantages of process algebras and Petri nets.

This paper tries to be, as far as possible, self-contained, so we will first review the syntax, the operational semantics and the denotational semantics of the finite operators. For further details the reader is referred to the previous works of the authors [14, 13]. The paper is structured as follows: in Section 2 we present an overview of the syntax. The operational and the denotational semantics can be found in Sections 3 and 4, respectively. Section 5 contains an example, and finally, Section 6 contains some conclusions and our plans for future work.

2 Syntax of stochastic Petri Box Calculus

In this section we present some notations and the syntax of sPBC, with an informal interpretation of the operators. In this paper we do not consider the recursion operator because it requires a more sophisticated treatment, as it occurs in plain PBC. Nevertheless, with the iteration operator we are expanding the power of description of sPBC significantly, and in fact some potentially infinite behaviours can be described with it.

2.1 Notation

From now onwards we will use the following notation: $\mathcal{A}$ will be a countable set of action names, $\forall\, a \in \mathcal{A}$, $\exists\, \widehat{a} \in \mathcal{A}$, such that $a \neq \widehat{a}$ and $\widehat{\widehat{a}} = a$, as in CCS [16]. 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*). We will consider relabelling functions $f : \mathcal{A} \to \mathcal{A}$, which are functions that preserve conjugates, i.e.: $\forall a \in \mathcal{A},\ f(\widehat{a}) = \widehat{f(a)}$. We will only consider bijective relabelling functions. We define the alphabet of $\alpha \in \mathcal{L}$ by: $A(\alpha) = \{a \in \mathcal{A} \,|\, \alpha(a) > 0\}$, and the set of stochastic multiactions by $\mathcal{SL} = \{< \alpha, r > \,|\, \alpha \in \mathcal{L}\,,\ r \in \mathbb{R}^+\}$. We

allow the same multiaction $\alpha \in \mathcal{L}$ to have different stochastic rates in the same specification. Finally, we define the 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 either $a \in A(\alpha)$ and $\widehat{a} \in A(\beta)$, or $\widehat{a} \in A(\alpha)$ and $a \in A(\beta)$.

2.2 Syntax

As in plain PBC, 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:

$$E ::= <\alpha, r> \mid E; E \mid E \,\Box\, E \mid E \parallel E \mid E[f] \mid E \; sy \; a \mid E \; rs \; a \mid [a : E] \mid [E * E * E]$$

where $<\alpha, r> \in \mathcal{SL}$ stands for the *basic multiaction*, which corresponds to the simultaneous execution of all the actions in α, 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 $E_1 \,\Box\, E_2$ is the choice between its arguments, $E[f]$ is the relabelling operator, and $E \; rs \; a$ denotes the restriction over the single action a (this process cannot execute any stochastic multiactions $<\alpha, r>$ with either $a \in A(\alpha)$ or $\hat{a} \in A(\alpha)$). The parallel operator, $\parallel$, represents the (independent) parallel execution of both components, where there is no any synchronization embedded in the operator (as in PBC). Synchronization is introduced by the operator *sy*, thus the process $E \; sy \; a$ behaves in the same way as E, but it can also execute those new multiactions generated by the synchronization of a pair of actions $(a, \hat{a})$. $[a : E]$ is the derived operator *scoping* defined by $[a : E] = (E \; sy \; a) \; rs \; a$. Finally, the iteration operator $[E_1 * E_2 * E_3]$ represents the process that performs E_1, then executes several (possibly 0) times E_2, and finishes after performing E_3. We can obtain infinite behaviours by adequately combining both the iteration and the restriction operators; for instance, $[<\{a\}, r_1> * <\{b\}, r_2> * <\{c\}, r_3>] \; rs \; c$, represents the process that performs $<\{a\}, r_1>$ once, and then it executes $<\{b\}, r_2>$ infinitely many times.

However, we need to restrict the syntax of sPBC to those terms for which no parallel behaviour appears at the highest level in a choice or in the two last arguments of an iteration. In principle, with this restriction we slightly reduce the expressiveness of the language, although we could prefix parallel operators appearing at the highest level of a choice or in one argument of an iteration by an empty multiaction, whose rate could be adequately selected in order to preserve the probability of execution of the multiactions involved in the choice or in the iteration. Terms fulfilling this restriction will be called *regular terms*,

and the operational semantics will be only defined for them. This restriction is introduced in order to guarantee that the moment in which the rule for the synchronization is applied does not affect the value that we obtain for the rate of the stochastic multiaction obtained as result of a synchronization (this will be illustrated in Example 1).

More exactly, regular static s-expressions E are those static s-expressions of sPBC fulfilling:

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

The operational semantics of sPBC is defined on dynamic s-expressions G, which derive from the static s-expressions, annotating them with either upper or lower bars, which indicate the *active components* at the current instant of time. Thus, we have:

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

where $\overline{E}$ denotes the initial state of E, and $\underline{E}$ its final state. We will say that a dynamic s-expression is regular if the underlying static s-expression is regular. The set of regular dynamic s-expressions will be denoted by *ReDynExpr*.

3 Operational semantics

We have two kind of transitions: inaction transitions, annotated with $\emptyset$, which just denote a rewriting of a term by redistributing its bars, in order to prepare the term to apply new transitions; and stochastic transitions, which correspond to the execution of a stochastic multiaction. Therefore, inaction rules define in fact an equivalence between regular dynamic s-expressions as defined in Def. 2. Inaction rules for sPBC are those presented in Tables 1 and 2.

Table 1. Inaction rules (I)

$\overline{E;F} \xrightarrow{\emptyset} \overline{E};F$	$\underline{E};F \xrightarrow{\emptyset} E;\overline{F}$	$E;\underline{F} \xrightarrow{\emptyset} \underline{E;F}$
$\overline{E \,\square\, F} \xrightarrow{\emptyset} \overline{E} \,\square\, F$	$\overline{E \,\square\, F} \xrightarrow{\emptyset} E \,\square\, \overline{F}$	$\underline{E} \,\square\, F \xrightarrow{\emptyset} \underline{E \,\square\, F}$
$E \,\square\, \underline{F} \xrightarrow{\emptyset} \underline{E \,\square\, F}$	$\overline{E \parallel F} \xrightarrow{\emptyset} \overline{E} \parallel \overline{F}$	$\underline{E} \parallel \underline{F} \xrightarrow{\emptyset} \underline{E \parallel F}$
$\overline{E[f]} \xrightarrow{\emptyset} \overline{E}[f]$	$\underline{E}[f] \xrightarrow{\emptyset} \underline{E[f]}$	$\overline{E\ sy\ a} \xrightarrow{\emptyset} \overline{E}\ sy\ a$
$\underline{E}\ sy\ a \xrightarrow{\emptyset} \underline{E\ sy\ a}$	$\overline{E\ rs\ a} \xrightarrow{\emptyset} \overline{E}\ rs\ a$	$\underline{E}\ rs\ a \xrightarrow{\emptyset} \underline{E\ rs\ a}$
$\dfrac{\forall op \in \{;\ ,\ \square\},\ G \xrightarrow{\emptyset} G'}{G\ op\ E \xrightarrow{\emptyset} G'\ op\ E}$	$\dfrac{\forall op \in \{;\ ,\ \square\},\ G \xrightarrow{\emptyset} G'}{E\ op\ G \xrightarrow{\emptyset} E\ op\ G'}$	$\dfrac{G \xrightarrow{\emptyset} G'}{G[f] \xrightarrow{\emptyset} G'[f]}$
$\dfrac{G_1 \xrightarrow{\emptyset} G_1'}{G1 \parallel G_2 \xrightarrow{\emptyset} G_1' \parallel G_2}$	$\dfrac{G_2 \xrightarrow{\emptyset} G_2'}{G1 \parallel G_2 \xrightarrow{\emptyset} G_1 \parallel G_2'}$	$\dfrac{\forall op \in \{sy\ ,\ rs\},\ G \xrightarrow{\emptyset} G'}{G\ op\ a \xrightarrow{\emptyset} G'\ op\ a}$

Table 2. Inaction rules (II)

$$\overline{[\,E * F * E'\,]} \xrightarrow{\emptyset} [\,\overline{E} * F * E'\,]\ [\,\underline{E} * F * E'\,] \xrightarrow{\emptyset} [\,E * \overline{F} * E'\,]$$

$$[\,E * \underline{F} * E'\,] \xrightarrow{\emptyset} [\,E * \overline{F} * E'\,]\ [\,E * \underline{F} * E'\,] \xrightarrow{\emptyset} [\,E * F * \overline{E'}\,]$$

$$[\,E * F * \underline{E'}\,] \xrightarrow{\emptyset} \underline{[\,E * F * E'\,]} \qquad \frac{G \xrightarrow{\emptyset} G'}{[\,G * E * F\,] \xrightarrow{\emptyset} [\,G' * E * F\,]}$$

$$\frac{G \xrightarrow{\emptyset} G'}{[\,E * G * F\,] \xrightarrow{\emptyset} [\,E * G' * F\,]} \qquad \frac{G \xrightarrow{\emptyset} G'}{[\,E * F * G\,] \xrightarrow{\emptyset} [\,E * F * G'\,]}$$

Definition 1. We say that a regular dynamic s-expression G is *operative* if it is not possible to apply any inaction rule from it. We will denote the set of all the operative regular dynamic s-expressions by $OpReDynExpr$. □

Definition 2. We define the structural equivalence relation for regular dynamic s-expressions by:

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

As usual, we denote the class of G with respect to $\equiv$ by $[G]_{\equiv}$. □

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 dynamic s-expressions that appear on the left-hand sides of each transition in the rules are regular and operative.

Table 3. Rules defining the stochastic transitions (I)

(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) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{G\,rs\,a \xrightarrow{<\alpha,r>} G'\,rs\,a}\ a, \widehat{a} \notin A(\alpha)$

(Re) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{G[f] \xrightarrow{<f(\alpha),r>} G'[f]}$

(E1) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{G \square F \xrightarrow{<\alpha,r>} G' \square F}$

(E2) $\dfrac{H \xrightarrow{<\alpha,r>} H'}{E \square H \xrightarrow{<\alpha,r>} E \square H'}$

(C1) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{G \| H \xrightarrow{<\alpha,r>} G' \| H}$

(C2) $\dfrac{H \xrightarrow{<\alpha,r>} H'}{G \| H \xrightarrow{<\alpha,r>} G \| H'}$

(It1) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{[\,G * E * F\,] \xrightarrow{<\alpha,r>} [\,G' * E * F\,]}$

(It2) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{[\,E * G * F\,] \xrightarrow{<\alpha,r>} [\,E * G' * F\,]}$

(It3) $\dfrac{G \xrightarrow{<\alpha,r>} G'}{[\,E * F * G\,] \xrightarrow{<\alpha,r>} [\,E * F * G'\,]}$

Rules in Table 3 define a total order semantics, in the sense that it is not contemplated the possibility of executing two multiactions at the same time.

Then, in order to define the semantics of the synchronization operator, we need to calculate all the possible sets of bags of stochastic multiactions that could be executed concurrently as result of one or several synchronizations over each operative regular dynamic s-expression.

Definition 3. We define $BC : \mathit{OpReDynExpr} \longrightarrow \mathcal{P}(\mathcal{B}(SL))$, as follows:

- If $G \in \mathit{OpReDynExpr}$ is final, i.e. G=$\underline{\text{E}}$, we take $BC(G) = \emptyset$.
- If $G \in \mathit{OpReDynExpr}$ is not final, we distinguish the following cases:
  - $BC(\overline{<\alpha, r>}) = \{\{<\alpha, r>\}\}$
  - If $\gamma \in BC(G)$, then: $\gamma \in BC(G; E)$, $\gamma \in BC(E; G)$, $\gamma \in BC(E \,\square\, G)$, $\gamma \in BC(G \,\square\, E)$, $\gamma \in BC(G\, rs\, a)$ (when $a, \widehat{a} \notin A(\gamma)$), $\gamma \in BC(G\, sy\, a)$, $f(\gamma) \in BC(G[f])$, $\gamma \in BC([\, G * E * F\,])$, $\gamma \in BC([\, E * G * F\,])$, $\gamma \in BC([\, E * F * G\,])$.
  - If $\gamma_1 \in BC(G)$, $\gamma_2 \in BC(H)$, then $\gamma_1 \in BC(G \| H)$, $\gamma_2 \in BC(G \| H)$, and $\gamma_1 + \gamma_2 \in BC(G \| H)$.
  - $\gamma \in BC(G\, sy\, a)$, and $<\alpha, r_1>, <\beta, r_2> \in \gamma$, (with either $<\alpha, r_1> \neq <\beta, r_2>$ or they are two different instances of the same stochastic multiaction in γ), with $a \in A(\alpha)$, and $\widehat{a} \in A(\beta)$, then: $\gamma' \in BC(G\, sy\, a)$, where: $\gamma' = (\gamma + \{<\alpha \oplus_a \beta, R>\}) \setminus \{<\alpha, r_1>, <\beta, r_2>\}$ and R is the rate of the new stochastic multiaction, to be later defined (see rule *Sy2* in Table 4). □

In order to define the rates for the stochastic multiactions generated by a synchronization we need to identify the situations of conflict. Concretely, for each operative regular dynamic s-expression G we define the multiset of associated conflicts for every instance of a stochastic multiaction $<\alpha, r>_i$ executable from G, which we will denote by $\mathit{Conflict}(G, <\alpha, r>_i)$, but we will only take those stochastic multiactions with exactly the same multiaction α. We will denote this multiset of conflicts by $\mathit{Conflict}(G, <\alpha, r>_i)$, although we will omit the subindex i if it is clear which instance of $<\alpha, r>$ we are considering [1].

Definition 4. We define the following partial function:

$$\mathit{Conflict} : \mathit{OpReDynExpr} \times \mathcal{SL} \longrightarrow \mathcal{B}(\mathcal{SL})$$

which for each instance i of the stochastic multiaction $<\alpha, r>$ executable from G gives us the multiset of stochastic multiactions $<\alpha, r'>$ in *conflict* with it. We define the function in a structural way:

1. $\mathit{Conflict}\,(\overline{<\alpha, r>}, <\alpha, r>) = \{<\alpha, r>\}$
2. If $<\alpha, r>$ is executable from G, and $C = \mathit{Conflict}\,(G, <\alpha, r>)$, then:
   (a) $\mathit{Conflict}\,(G; E, <\alpha, r>) = \mathit{Conflict}\,(E; G, <\alpha, r>) = C$,
   (b) $\mathit{Conflict}\,(G \| H, <\alpha, r>) = \mathit{Conflict}\,(H \| G, <\alpha, r>) = C$,
   (c) If $a, \hat{a} \notin A(\alpha)$, then $\mathit{Conflict}\,(G\, rs\, a, <\alpha, r>) = C$,
   (d) For any bijective function f, $\mathit{Conflict}\,(G[f], <f(\alpha), r>) = f(C)$,

[1] To avoid a more sophisticated formal definition, we have preferred to omit the indices in the definition.

(e) For the choice operator we need to distinguish the following two cases:
- If $G \not\equiv \overline{E}$: $\mathit{Conflict}\,(G \,\square\, F, <\alpha, r>) = \mathit{Conflict}\,(F \,\square\, G, <\alpha, r>) = C$
- If $G \equiv \overline{E}$: $\mathit{Conflict}\,(G \,\square\, F, <\alpha, r>) = \mathit{Conflict}\,(F \,\square\, G, <\alpha, r>) =$
$C + \{\!|\, <\alpha, r_j> \mid \exists\, H_i \in \mathit{OpReDynExpr}\,,\ H_i \equiv \overline{F}$ and $H_i \stackrel{<\alpha, r_j>}{\longrightarrow} H_i' \,|\!\}$

(f) For the iteration operator we have:
- $\mathit{Conflict}(\,[\,G * E * F\,], <\alpha, r>) = C$
- For the two last arguments of an iteration we have:
  - If $G \not\equiv \overline{E'}$: $\mathit{Conflict}\,(\,[\,E * G * F\,], <\alpha, r>) =$
  $\mathit{Conflict}\,(\,[\,E * F * G\,], <\alpha, r>) = C$
  - If $G \equiv \overline{E'}$: $\mathit{Conflict}\,(\,[\,E * G * F\,], <\alpha, r>) =$
  $\mathit{Conflict}\,(\,[\,E * F * G\,], <\alpha, r>) = C +$
  $\{\!|\, <\alpha, r_j> \mid \exists\, H_i \in \mathit{OpReDynExpr}\,,\ H_i \equiv \overline{F}$ and $H_i \stackrel{<\alpha, r_j>}{\longrightarrow} H_i' \,|\!\}$

(g) $\mathit{Conflict}\,(G\,sy\,a, <\alpha, r>) = C$,

3. Let $\{<\alpha_1, r_1>, <\alpha_2, r_2>\} \in BC\,(G\,sy\,a)$, $a \in A(\alpha_1)$, $\hat{a} \in A(\alpha_2)$ and $G\,sy\,a \stackrel{<\alpha_1 \oplus_a \alpha_2, R_{12}>}{\longrightarrow} G'\,sy\,a$ obtained by applying rule *Sy2* . Then:
$\mathit{Conflict}\,(G\,sy\,a, <\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>) =$
$\{\!|\ <\alpha_1 \oplus_a \alpha_2\,,\, R_{ij}> \mid <\alpha_1, r_i> \in C_1,\ <\alpha_2, r_j> \in C_2,$ where
$R_{ij} = \frac{r_i}{cr(G sy\, a, <\alpha_1, r_1>)} \frac{r_j}{cr(G sy\, a, <\alpha_2, r_2>)} \cdot \min\limits_{i=1,2} \{cr(G\,sy\,a, <\alpha_i, r_i>)\}\ |\!\}$
taking: $C_i = \mathit{Conflict}(G\,sy\,a, <\alpha_i, r_i>)$, $i = 1, 2$, and $cr(\,G, <\alpha, r>_i)$ is the so called *conflict rate* for G and $<\alpha, r>_i$, defined by:

$$cr(\,G, <\alpha, r>_i) = \sum_{<\alpha, r_j> \in \mathit{Conflict}\,(\,G, <\alpha, r>_i)} r_j \cdot n_j$$

where n_j is the number of instances of $<\alpha, r_j>$ in $\mathit{Conflict}(G, <\alpha, r>_i)$.□

Rules for the synchronization operator are shown in Table 4. Observe that we take as rate of the generated stochastic multiaction the minimum of the conflict rates of $<\alpha_1, r_1>$, $<\alpha_2, r_2>$, weighted by a factor, which is introduced in order to guarantee that an equivalence relation defined in [12] is in fact a congruence. For short, we will denote the stochastic multiaction obtained by synchronization of the stochastic multiactions $<\alpha_1, r_1>$ and $<\alpha_2, r_2>$ by $<\alpha_1, r_1> \oplus_a <\alpha_2, r_2>$.

Definition 5. For each $G \in \mathit{ReDynExpr}$ we define the set of the dynamic s-expressions that can be derived from $[G]_{\equiv}$ by:

$$[G\rangle = \{G' \mid G' \in [G]_{\equiv}\} \cup \{H' \in \mathit{ReDynExpr} \mid \exists <\alpha_1, r_1>, \ldots, <\alpha_n, r_n> \in \mathcal{SL}$$
$$\text{with } G \equiv G' \stackrel{<\alpha_1, r_1>}{\longrightarrow} G_1 \equiv G_1' \stackrel{<\alpha_2, r_2>}{\longrightarrow} \ldots\ G_{n-1} \equiv G_{n-1}' \stackrel{<\alpha_n, r_n>}{\longrightarrow} H \equiv H'\}$$

□

In [13] we proved for finite sPBC (without iteration) that for any bag γ of stochastic multiactions executable from a regular dynamic s-expression G, any serialization of γ can be executed from G, and the multiset of conflicts for any stochastic multiaction in γ is preserved along the serialized execution of γ. This result can be easily extended to sPBC with iteration, and we can use it in order to compute the rate of the stochastic multiactions obtained after a number of synchronizations.

Table 4. Rules for the synchronization operator

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

(**Sy2**) Let $\{<\alpha_1, r_1>, <\alpha_2, r_2>\} \in BC(G\ sy\ a)$, $a \in A(\alpha_1)$, $\hat{a} \in A(\alpha_2)$, then

$$\frac{G\ sy\ a \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_1\ sy\ a\ (\stackrel{\emptyset}{\longrightarrow})^*\ G_1^*\ sy\ a \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_{12}\ sy\ a}{G\ sy\ a \stackrel{<\alpha_1 \oplus_a \alpha_2\ ,\ R>}{\longrightarrow} G_{12}\ sy\ a}$$

$$R = \frac{r_1}{cr(G\ sy\ a, <\alpha_1, r_1>)} \frac{r_2}{cr(G\ sy\ a, <\alpha_2, r_2>)} \cdot \min_{i=1,2} \{cr(G\ sy\ a, <\alpha_i, r_i>)\}$$

Proposition 1. Let G be a regular operative dynamic s-expression, $\gamma = \{<\alpha_1, r_1>, <\alpha_2, r_2>, \ldots, <\alpha_n, r_n>\} \in BC(G)$, and a serialization of γ, for which we may apply $n-1$ times rule *Sy2*:

$$G \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_1 (\stackrel{\emptyset}{\longrightarrow})^* G_1^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} \ldots \stackrel{<\alpha_n,r_n>}{\longrightarrow} G_n$$

to obtain a single transition $G \stackrel{<\beta,R>}{\longrightarrow} G_n$.

Then we have:
$$R = (\prod_{k=1}^{n} \frac{r_k}{cr(G, <\alpha_k, r_k>)}) \cdot \min_{k=1,\ldots,n} \{cr(G, <\alpha_k, r_k>)\}$$
$$cr(G, <\beta, R>) = \min_{k=1,\ldots,n} \{cr(G, <\alpha_k, r_k>)\}$$

Proof It can be found in [12]. □

Consequently, for all the possible transition sequences obtained by a serialization of γ, if we can apply rule *Sy2* a number of times to reach a single stochastic multiaction, then we have that it does not matter the order in which rule *Sy2* has been applied, neither the transition sequence used, i.e., we will always obtain the same stochastic multiaction.

Corollary 1. Let G be a regular operative dynamic s-expression, $\gamma = \{<\alpha_1, r_1>, <\alpha_2, r_2>, \ldots, <\alpha_n, r_n>\} \in BC(G)$, and two permutations of the set $\{1, \cdots, n\}$: $\{i_1, \cdots, i_n\}$ and $\{j_1, \cdots, j_n\}$. Assuming that there are two serializations:

$$G \stackrel{<\alpha_{i_1},r_{i_1}>}{\longrightarrow} G_1 (\stackrel{\emptyset}{\longrightarrow})^* G_1^* \stackrel{<\alpha_{i_2},r_{i_2}>}{\longrightarrow} \ldots \stackrel{<\alpha_{i_n},r_{i_n}>}{\longrightarrow} G_n$$
$$G \stackrel{<\alpha_{j_1},r_{j_1}>}{\longrightarrow} G'_1 (\stackrel{\emptyset}{\longrightarrow})^* G'^*_1 \stackrel{<\alpha_{j_2},r_{j_2}>}{\longrightarrow} \ldots \stackrel{<\alpha_{j_n},r_{j_n}>}{\longrightarrow} G'_n$$

from which we may apply $n-1$ times rule *Sy2* (for the same actions $a_1, \ldots, a_{n-1}$, possibly repeated, but the same number of times in both cases), to obtain a single transition $G \stackrel{<\beta_i,R_i>}{\longrightarrow} G_n$ and $G \stackrel{<\beta_j,R_j>}{\longrightarrow} G'_n$, respectively, then we have: $G_n \equiv G'_n$ and $<\beta_i, R_i> = <\beta_j, R_j>$. □

Now we show an example that motivates the need for the syntactical restriction introduced, specifically in the case of iteration. With this example we will raise the problem that appears when we consider a parallel behaviour in the highest level in the body of an iteration.

Example 1. Let G be the following non-regular operative dynamic s-expression:

$$G = (\,[\,<\{a\}, r_1> \,*\, (\overline{<\{b\}, r_2>} \,\|\, \overline{<\{b\}, r_3>} \,\|\, \overline{<\{\hat{b}, \hat{b}\}, r_4>}) * \,<\{b\}, r_5>\,]\,)\; sy\; a$$

It follows that $\gamma = \{<\{b\}, r_2>, <\{b\}, r_3> <\{\hat{b}, \hat{b}\}, r_4>\} \in BC(G)$. Then, according to the definition of BC, we also have $\gamma_1 = \{<\{b\}, r_2>, <\{\hat{b}\}, R_{34}>\}$ $\in BC\,(G)$, with

$$R_{34} = \frac{r_3}{r_3 + r_5} \cdot min\{r_3 + r_5, r_4\}$$

However, not every serialization of γ_1 is possible from G, because $<\{\hat{b}\}, R_{34}>$ cannot be executed from G_1, where $G \xrightarrow{<\{b\}, r_2>} G_1$ and

$$G_1 = ([<\{a\}, r_1> \,*\, (\underline{<\{b\}, r_2>} \,\|\, \overline{<\{b\}, r_3>} \,\|\, \overline{<\{\hat{b}, \hat{b}\}, r_4>}) * \,<\{b\}, r_5>])\; sy\; a$$

Actually, we can execute $<\{\hat{b}\}, R'_{34}>$ from G_1, with $R'_{34} = min\{r_3, r_4\}$, and in general we have $R_{34} \neq R'_{34}$. □

Definition 6. We define the labelled (multi)transition system of any regular dynamic s-expression G, $ts(G) = (V, A, v_0)$, where:

- $V = \{[H]_{\equiv} \,|\, H \in [\,G\,\rangle\}$ is the set of states.
- $v_0 = [G]_{\equiv}$ is the initial state.
- A is the multiset of transitions, given by:

$$A = \{\!\!\{\ ([H]_{\equiv}, <\alpha, r>, [J]_{\equiv}) \,|\, H \in [\,G\,\rangle \ \wedge\ H \xrightarrow{<\alpha, r>} J\ \}\!\!\}$$

In order to compute the number of different instances of each transition $([H]_{\equiv}, <\alpha, r>, [J]_{\equiv})$ in A, we consider equivalent all the different ways to derive the same stochastic transition by considering the different serializations of the same γ (Corollary 1). Therefore, for each equivalence class, we will only consider one of its representatives, which can be chosen imposing that the stochastic multiactions in each $\gamma \in BC(G)$ will be executed in the same order as they appear in the syntax of the s-expression G, i.e., we enumerate the stochastic multiactions from left to right in the syntax of the s-expression, and then, when we apply rule *Sy2*, the generated stochastic multiaction can be annotated with the concatenation of the numbering of the corresponding stochastic multiactions involved in the synchronization, so that when we detect that a permutation of the numbering has been already obtained, by a previous application of the rule, then that new stochastic transition will not be considered (see [14] for more details). □

The *race policy* will govern the dynamic behaviour of the system when two or more stochastic multiactions are simultaneously enabled (i.e., when several stochastic multiactions are possible, the fastest one will win the race). Then, as we are using exponential distributions, the stochastic process associated with the evolution of every regular dynamic s-expression $\overline{E}$ is a Continuous Time Markov Chain (CTMC), which can be easily obtained from $ts(\overline{E})$ in the same way as we showed in [14]: we modify the multigraph $ts(\overline{E})$ 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.

4 Denotational semantics

We now 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 SPNs, but they have the same structure as Petri boxes of PBC. These 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 λ 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 $\mathcal{B}(\mathcal{L}) \times \mathcal{L}$; i.e., $\lambda(t)$ is a relation which associates elements of $\mathcal{L}$ to bags of multiactions. By convention, ${}^\circ\Sigma$ and Σ° 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, our boxes are defined to be labelled simple nets such that the following conditions hold: ${}^\circ\Sigma \neq \emptyset \neq \Sigma^\circ$, ${}^\bullet({}^\circ\Sigma) = \emptyset = (\Sigma^\circ)^\bullet$ and $\forall t \in T : {}^\bullet t \neq \emptyset \neq t^\bullet$. A box is said to be *plain* when for every $t \in T$, $\lambda(t)$ is a constant relation, i.e., an element of $\mathcal{L}$.

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, λ) is a plain box, and $\mu : T \longrightarrow \mathbb{R}^+$ is a stochastic function, which associates a rate to every transition. □

A plain s-box can be either marked or not [2]. We will denote by M_e the marking in which only *entry places* are marked (each one with a single token); on the other hand, M_x will denote the marking in which only *exit places* are marked, each one with a single token. We say that a marking M is k-safe if for all $s \in S$, $M(s) \leq k$, and we say that M is clean if it is not a proper super-multiset of ${}^\circ\Sigma$ nor Σ°. Then, a marked plain s-box is k-safe if all its reachable markings are k-safe, safe if all its its reachable markings are 1-safe, and clean if all its reachable markings are clean.

4.1 Algebra of s-boxes

For each transition that we can obtain compositionally we need to know which transitions are in conflict with it, in order to compute its *conflict rates*. Thus, we enumerate stochastic multiactions appearing from left to right in the syntax of regular static s-expressions, and we preserve this enumeration in the corresponding transitions of the stochastic Petri net. Only with the synchronization operator we can obtain some new transitions, which will be annotated with the concatenation of the numeration of the transitions involved.

Another decision that we must take is the selection of the operator box that we will use for the iteration, since we have two proposals in plain PBC

[2] A marked plain s-box is essentially a kind of marked labelled stochastic Petri net, whose behaviour follows the classical *firing rule* of SPNs.

for that [5]; one of them provides us with a 1-safe version (with six transitions in the operator box), but there is also a simpler version, which has only three transitions in the operator box. In general, in PBC, with the latter version we may generate 2-safe nets, which only occurs when a parallel behaviour appears at the highest level of the body of the iteration. Nevertheless, in our case, and due to the syntactical restriction introduced, this particular case cannot occur, so that the net obtained will be always 1-safe.

In order to define the semantic function that associates a plain s-box with every regular term of sPBC, we need to consider the following functions:

$$\eta : T \longrightarrow \mathbb{N}^* \quad \text{and} \quad \kappa : T \longrightarrow \mathcal{P}(\mathbb{N}^*)$$

where $\eta(t)$ stands for the numeration of t according to our criterion (enumeration from left to right, and concatenation in case of synchronization), and $\kappa(t)$ identifies the set of transitions in conflict with t. These functions will be defined in a structural way, as we construct the corresponding plain s-box.

For each transition $t \in T$, we also define its corresponding *conflict rate*, and we will denote it by $cr(t)$:

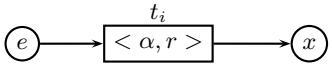
$$cr(t) = \sum_{\eta(t_j) \in \kappa(t)} \mu(t_j)$$

Then, the structure of the net is obtained as in PBC, combining both refinement and relabelling. Consequently, the s-boxes thus obtained will be safe and clean. Therefore, the denotational semantics for regular static s-expressions can be formally defined by the following homomorphism:

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

As previously mentioned, we have to define η, κ for every operator of sPBC.

– $Box_s(< \alpha, r >_i) = N_{<\alpha,r>_i} =$



taking $\eta(t_i) = i$ and $\kappa(t_i) = \{i\}$.

For the remaining operators of sPBC the corresponding operator s-boxes are shown in Fig. 1, where the relabelling functions $\rho_{op} \subseteq \mathcal{B}(\mathcal{SL}) \times \mathcal{SL}$ that appear in that Figure are defined as follows:[3]

- $\rho_{id} = \{(\{< \alpha, r >\}, < \alpha, r >) \mid < \alpha, r > \in \mathcal{SL}\}$
- $\rho_{[f]} = \{(\{< \alpha, r >\}, < f(\alpha), r >) \mid < \alpha, r > \in \mathcal{SL}\}$
- $\rho_{rs\,a} = \{(\{< \alpha, r >\}, < \alpha, r >) \mid < \alpha, r > \in \mathcal{SL} \ \wedge \ a, \widehat{a} \notin A(\alpha)\}$

Thus, the corresponding semantic functions are defined as follows, where $Box_s(E_i) = (S_i, T_i, W_i, \lambda_i, \mu_i)$ is the plain s-box corresponding to E_i, and η_i , κ_i are the enumeration and conflict functions for T_i, $i = 1, 2, 3$.

– $Box_s(E_1 \,;\, E_2) = \Omega_{;}(Box_s(E_1), Box_s(E_2))$. Then, we take:

$$\eta(t) = \begin{cases} \eta_1(t) & \text{if } t \in T_1 \\ \eta_2(t) & \text{if } t \in T_2 \end{cases} \qquad \kappa(t) = \begin{cases} \kappa_1(t) & \text{if } t \in T_1 \\ \kappa_2(t) & \text{if } t \in T_2 \end{cases}$$

[3] We separate the definition of $\rho_{sy\,a}$, which will be presented later, when we formally define $Box_s(E_1\ sy\ a)$.

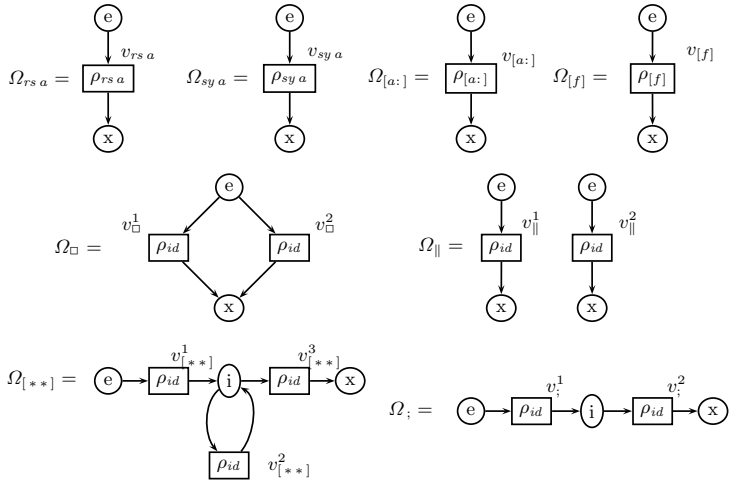


Fig. 1. The operator s-boxes for sPBC

- $Box_s(E_1 \parallel E_2) = \Omega_{\|}(Box_s(E_1), Box_s(E_2))$. η and κ are defined in exactly the same way as in the previous case.
- $Box_s(E_1[f]) = \Omega_{[f]}\,(Box_s\,(E_1)\,)$.

$$\eta(t) = \eta_1(t)\ , \quad t \in T_1 \quad \text{and} \quad \kappa(t) = \kappa_1(t)\ , \quad t \in T_1$$

- $Box_s(E_1 \,\square\, E_2) = \Omega_{\square}(Box_s(E_1), Box_s(E_2))$.

$$\eta(t) = \begin{cases} \eta_1(t) & \text{if } t \in T_1 \\ \eta_2(t) & \text{if } t \in T_2 \end{cases}$$

$$\kappa(t) = \begin{cases} \kappa_1(t) \cup \kappa_2(t') & \text{if } t \in T_1,\ {}^\bullet t \in {}^\circ Box_s(E_1),\ \exists\, t' \in T_2,\ {}^\bullet t' \in {}^\circ Box_s(E_2),\ \lambda(t) = \lambda(t') \\ \kappa_1(t) & \text{if } t \in T_1,\ {}^\bullet t \in {}^\circ Box_s(E_1),\ \nexists\, t' \in T_2,\ {}^\bullet t' \in {}^\circ Box_s(E_2),\ \lambda(t) = \lambda(t') \\ \kappa_1(t) & \text{if } t \in T_1,\ {}^\bullet t \notin {}^\circ Box_s(E_1) \\ \kappa_2(t) \cup \kappa_1(t') & \text{if } t \in T_2,\ {}^\bullet t \in {}^\circ Box_s(E_2),\ \exists\, t' \in T_1,\ {}^\bullet t' \in {}^\circ Box_s(E_1),\ \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_2,\ {}^\bullet t \in {}^\circ Box_s(E_2),\ \nexists\, t' \in T_1,\ {}^\bullet t' \in {}^\circ Box_s(E_1),\ \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_2,\ {}^\bullet t \notin {}^\circ Box_s(E_2) \end{cases}$$

- $Box_s(\,[\,E_1 * E_2 * E_3\,]\,) = \Omega_{[**]}(Box_s(E_1), Box_s(E_2), Box_s(E_3))$. No new transitions are introduced with this operator, so the numeration of transitions is preserved. However, it is clear that this operator will introduce some new conflicts. Specifically, those transitions in T_2 having their preconditions in ${}^\circ Box_s(E_2)$ are in conflict with those transitions in T_3 with preconditions in ${}^\circ Box_s(E_3)$, if they have the same associated multiaction (the same label). Notice that since we are working with regular terms, $Box_s(E_2)$ and $Box_s(E_3)$ will have a single entry place. Formally:

$$\eta(t) = \begin{cases} \eta_1(t) & \text{if } t \in T_1 \\ \eta_2(t) & \text{if } t \in T_2 \\ \eta_3(t) & \text{if } t \in T_3 \end{cases}$$

$$\kappa(t) = \begin{cases} \kappa_1(t) & \text{if } t \in T_1 \\ \kappa_2(t) \cup \kappa_3(t') & \text{if } t \in T_2,\ {}^\bullet t \in {}^\circ Box_s(E_2),\ \exists\, t' \in T_3,\ {}^\bullet t' \in {}^\circ Box_s(E_3),\ \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_2,\ {}^\bullet t \in {}^\circ Box_s(E_2),\ \nexists\, t' \in T_3,\ {}^\bullet t' \in {}^\circ Box_s(E_3),\ \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_2,\ {}^\bullet t \notin {}^\circ Box_s(E_2) \\ \kappa_3(t) \cup \kappa_2(t') & \text{if } t \in T_3,\ {}^\bullet t \in {}^\circ Box_s(E_3),\ \exists\, t' \in T_2,\ {}^\bullet t' \in {}^\circ Box_s(E_2),\ \lambda(t) = \lambda(t') \\ \kappa_3(t) & \text{if } t \in T_3,\ {}^\bullet t \in {}^\circ Box_s(E_3),\ \nexists\, t' \in T_2,\ {}^\bullet t' \in {}^\circ Box_s(E_2),\ \lambda(t) = \lambda(t') \\ \kappa_3(t) & \text{if } t \in T_3,\ {}^\bullet t \notin {}^\circ Box_s(E_3) \end{cases}$$

Notice that $\kappa(t)$ is well defined for the second and fifth cases, because $\kappa_3(t')$ coincides for every $t' \in T_3,\ {}^\bullet t' \in {}^\circ Box_s(E_3),\ \lambda(t) = \lambda(t')$, and respectively for the other case.

- $Box_s(E_1\ rs\ a) = \Omega_{rs\ a}\,(Box_s\,(E_1)\,)$.
  $\eta(t) = \eta_1(t)\,,\ t \in T_1\,,\ a, \hat{a} \notin \lambda(t) \quad \text{and} \quad \kappa(t) = \kappa_1(t)\,,\ t \in T_1\,,\ a, \hat{a} \notin \lambda(t)$
- $Box_s(E_1\ sy\ a) = \Omega_{sy\ a}\,(Box_s\,(E_1)\,)$. We take the following relation for the synchronization: $\rho_{sy\ a} \subseteq \mathcal{B}(\mathcal{SL}) \times \mathcal{SL}$, as the least relabelling relation containing ρ_{id}, and fulfilling:
  $(\Gamma, \alpha + \{a\}) \in \rho_{sy\ a} \quad \wedge \quad (\Delta, \beta + \{\hat{a}\}) \in \rho_{sy\ a}\ \text{ then }\ (\Gamma + \Delta, \alpha + \beta) \in \rho_{sy\ a}$
  Thus, $\rho_{sy\ a}$ allows us to obtain the net structure, as well as the multiactions labelling the transitions. Now, for every $t_1, t_2 \in T_1,\ \lambda(t_1) = \alpha + \{a\},\ \lambda(t_2) = \beta + \{\hat{a}\}$, a new transition t is generated by the synchronization, whose label is $\alpha + \beta$, and its rate is computed as follows:
  $$\frac{\mu_1(t_1)}{cr(t_1)} \cdot \frac{\mu(t_2)}{cr(t_2)} \cdot min(cr(t_1), cr(t_2))$$
  Moreover,
  $$\eta(t) = \eta_1(t_1)\,.\,\eta_1(t_2)$$
  $$\kappa(t) = \kappa_1(t_1) \otimes \kappa_1(t_2)\ =\ \{n_1\,.\,n_2 \,|\, n_1 \in \kappa_1(t_1)\,,\ n_2 \in \kappa_2(t_2)\,\}$$
  Notice that in order not to introduce redundant transitions, we only consider in the plain s-box a single one of the possible transitions that we can obtain by synchronizing (in different order) the same set of transitions. Furthermore, those transitions that were in T_1 have the same label, rate, numeration and conflict as they had in $Box_s(E_1)$. On the other hand, with this construction we can obtain in principle infinite nets, as it occurs in PBC, but, taking into account that the obtained nets are safe, the arcs having non-unitary weight will not enable the corresponding transitions, and thus, these transitions and arcs can be removed from the net structure, without affecting its behaviour.

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

Theorem 1. For any regular static s-expression E, the transition system $ts(\overline{E})$ associated with $\overline{E}$, 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, because we take a total order semantics both in the algebra and in s-boxes. Furthermore, the stochastic multiactions obtained in the algebra and the corresponding transitions in the plain s-box are labelled with the same rate; thus, the transition system $ts(\overline{E})$ and the reachability graph of the marked SPN $(Box_s(E), M_e)$ behave in exactly the same way. □

5 A simple example: The Producer/Consumer System

In this section we consider the classical *Producer/Consumer* system, firstly considering a buffer with capacity 1, and afterwards we will see how to extend the specification to a more general case (buffer with capacity n).

Each multiaction α in the specification has associated a delay that follows a negative exponential distribution with rate r_α. There are three different components: P (Producer), C (Consumer) and B (Buffer). The three components work in parallel, but they have to synchronize in a set of actions: i (for *initiating* the process), f (for *finishing* it), s (for *storing* an item into the buffer) and g (for *getting* an item from the buffer). The specification of every component is as follows.

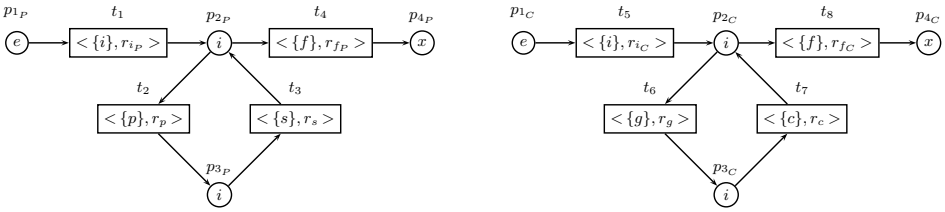
Producer: At the beginning, it is ready to initiate the process: $<\{i\}, r_{i_P}>$. Then, it starts a cyclic behaviour consisting of producing an item $<\{p\}, r_p>$, followed by storing the item into the buffer $<\{s\}, r_s>$. Finally, it ends its execution ($<\{f\}, r_{f_P}>$). This behaviour can be modelled by:

$$Producer = [<\{i\}, r_{i_P}> * (<\{p\}, r_p>; <\{s\}, r_s>) * <\{f\}, r_{f_P}>]$$

Consumer: At the beginning, it is ready to initiate the process: $<\{i\}, r_{i_C}>$. Then, it starts a cyclic behaviour consisting of getting an item from the buffer $<\{g\}, r_g>$, followed by consuming the item $<\{c\}, r_c>$. Finally, it ends its execution ($<\{f\}, r_{f_C}>$). The corresponding specification in sPBC follows:

$$Consumer = [<\{i\}, r_{i_C}> * (<\{g\}, r_g>; <\{c\}, r_c>) * <\{f\}, r_{f_C}>]$$

The corresponding plain s-boxes are:



Buffer$_1$: we first consider a buffer with capacity 1; the corresponding specification in sPBC is:

$$Buffer_1 = [<\{\hat{i}, \hat{i}\}, r_{i_B}> * (<\hat{s}, r_{\hat{s}}> ; <\hat{g}, r_{\hat{g}}>) * <\{\hat{f}, \hat{f}\}, r_{f_B}>]$$

Finally, the complete specification of the System is:

$$System = [A : (Producer \parallel Consumer \parallel Buffer_1)]$$

where $A = \{i, f, s, g\}$.

The generalization to a buffer of capacity n ($n \geq 2$) is straightforward, we just need to change the specification of the buffer as follows:

$$I_1 = (<\hat{s}, r_{\hat{s}}>; <\hat{g}, r_{\hat{g}}>)$$
$$I_n = [<\hat{s}, r_{\hat{s}}> * I_{n-1} * <\hat{g}, r_{\hat{g}}>], \quad n \geq 2$$
$$Buffer_n = [<\{\hat{i}, \hat{i}\}, r_{i_B}> * I_n * <\{\hat{f}, \hat{f}\}, r_{f_B}>]$$

The corresponding plain s-box for $Buffer_n$ is shown in Fig. 2.

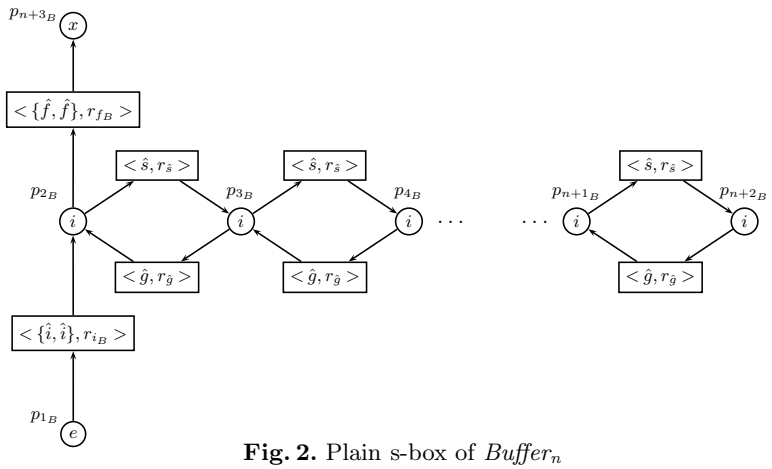


Fig. 2. Plain s-box of $Buffer_n$

In this case:

$$System_n = [\, A \,:\, (\, Producer \parallel Consumer \parallel Buffer_n \,)\,]$$

where $A = \{i, f, s, g\}$.

6 Conclusions and Future Work

sPBC is a Markovian extension of PBC which preserves the main features of that model. Thus, the syntax of sPBC is a natural stochastic extension of PBC, by annotating the multiactions with rates, which represent the parameter of an exponential distribution.

In this paper we have extended the operational and the denotational semantics that we presented in [14] for finite sPBC, by including the iteration operator, and considering the new version for the semantics of the synchronization operator, which is inspired in that one presented in [13].

The denotational semantics of sPBC is defined using as semantic objects a special kind of labelled stochastic Petri nets, called *s-boxes*. This will be a static translation in the sense that the rates of the transitions of the corresponding SPNs will not be marking dependent.

Our work in progress is focused to the definition of a stochastic bisimulation, which will capture more precisely those processes that can be considered equivalent taking into account the stochastic information. Our plans for future work include the treatment of the recursion operator, and the inclusion of some additional features in the language, such as immediate multiactions.

References

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

2. M. Bernando and R. Gorrieri. A Tutorial on EMPA : A Theory of Concurrent Process with Nondeterminism, Priorities, Probabilities and Time. *Theorical Computer Science, 202:1-54*, 1998.
3. E. Best, R. Devillers, and M. Koutny. Petri Nets, Process Algebras and Programming Languages. In *Lectures on Petri Nets II: Applications,* W. Reisig and Rozenberg (eds.), Advances in Petri Nets, Volume 1492, Springer-Verlag, pp. 1-84,, 1998.
4. E. Best, R. Devillers, and M. Koutny. A Consistent Model for Nets and Process Algebras. In the book *The Handbook on Process Algebras,* J.A. Bergstra, A. Ponse and S.S. Smolka (Eds.), North Holland, Chapter 14, pages 873-944, 2001.
5. E. Best, R. Devillers, and M. Koutny. *Petri Net Algebra*. EATC, Springer, 2001.
6. E. Best, R. Devilllers, and J. Hall. The Box Calculus: A New Causal Algebra with Multi-label Communication. In *Advances in Petri Nets,* G. Rozenberg (Eds.), LNCS 609, Springer, pages 21-69, 1992.
7. E. Best and M. Koutny. A Refined View of the Box Algebra. In *Proc. Application and Theory of Petri Nets.* LNCS 935, Springer, pp. 1-20, 1995.
8. H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences and Axioms for MTIPP. In *Proc. of the 2nd Workshop on Process Algebra and Performance Modelling,* U. Herzog and M. Rettelbach, (Eds.) Regensberg/Erlangen, pp.71-88, 1994.
9. J. Hillston. The nature of the synchronization. In *Int. Workshop on Process Algebra and Performance Modelling*, 1994.
10. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
11. M. Koutny. A Compositional Model of Time Petri Nets. *International Conference on Theory and Application of Petri Nets, 2000, LNCS 1825,* pp. 303-322, Springer, 2000.
12. H. Macià. *sPBC: Una Extensión Markoviana del Petri Box Calculus*. PhD thesis, Departamento de Informática, Universidad de Castilla-La Mancha (in spanish), 2003.
13. H. Macià, V. Valero, F. Cuartero, and F.L. Pelayo. A new proposal for the synchronization in sPBC. In *Proc. Third IEEE Int. Conference on Application of Concurrent to System Design (ACSD'03),* pp. 216-225, IEEE Computer Society Press, 2003.
14. H. Macià, V. Valero, and D. de Frutos. sPBC: A Markovian Extension of Finite Petri Box Calculus. In *Proc. 9th IEEE Int. Workshop on Petri Nets and Performance Models (PNPM'01),* pp. 207-216. IEEE Computer Society Press, 2001.
15. O. Marroquín and D. de Frutos. Extending the Petri Box Calculus with Time. In *Proc. Int. Conf. on Theory and Application of Petri Nets 2001. LNCS 2075, Springer,* pp. 195-207, 2001.
16. R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.
17. M. Ribaudo. Stochastic Petri Net Semantics for Stochastic Process Algebra. In *Proc. 6th Int. Workshop on Petri Nets and Performance Models, (PNPM'95),* Durham, 1995.