University of Castilla-La Mancha

A publication of the

Department of Computer Science

A new proposal for the synchronization in sPBC

by H. Macià, V. Valero Ruiz, F. Cuartero, F.L. Pelayo .

Technical Report **#DIAB-02-01-26** June 2002

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

A new proposal for the synchronization in sPBC

Hermenegilda Macià, Valentín Valero, Fernando Cuartero, and Fernando L. Pelayo *

Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
Campus Universitario s/n. 02071. Albacete, SPAIN
{Hermenegilda.Macia, Valentin.Valero, Fernando.Cuartero, Fernando.Lopez}@uclm.es

Abstract. In this paper we present a new proposal for the synchronization of a stochastic extension of finite Petri Box Calculus, sPBC, in which multiactions 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.

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

1 Introduction

The Petri Box Calculus (PBC) [3–7, 10] 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 [13], 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 [9] and TPBC [12]. Our contribution 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 associated a random delay which follows a negative exponential distribution. Thus, 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. 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 behaviour of Stochastic Petri Nets [1].

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. 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 total order semantics, i.e., a semantics that satisfies the *Total Order Assumption (TOA)* [2]: *All execution sequences of observable multiactions are totally ordered by precedence*, as opposed to the partial order semantics (*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.

In a previous paper [11], our goal was 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

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

interpretation. For instance, for the new multiactions generated by the application of the synchronization operator, we needed to select a new parameter for the corresponding exponential distribution and several criteria could be applied to do it. We concluded that a good option could be to take as the value of the parameter the least of the values involved in the synchronization, which at the intuitive level could 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). But with this definition we have found some problems in order to define a stochastic equivalence relation, which has motivated this new proposal for the synchronization, which is the main goal of the present paper.

The paper is structured as follows: in Section 2 we present an overview of the syntax and the operational semantics of finite sPBC (for a fully description, the reader is referred to [11]). Our new proposal for the synchronization is introduced in Section 3. Finally, Section 4 contains some conclusions and the future work.

2 The Stochastic Petri Box Calculus

2.1 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.

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 [13]. 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 α. 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.: $\forall a \in \mathcal{A},\ f(\widehat{a}) = \widehat{f(a)}$. Only bijective functions will be considered.
- 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 by: $A(< \alpha, r >) =_{def} A(\alpha)$.

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 $a \in \beta$.
- **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. 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 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 $E_1 \,\square\, E_2$ is a choice, $[f]$ 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 $(a, \hat{a})$. Finally, $[a : E]$ is a derived operator (*scoping*), which is defined by $[a : E] = (E\ sy\ a)\ rs\ a$.

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, i.e., $\overline{E}$ indicates that the static s-expression E has been activated (its stochastic multiactions can be executed) which represents the initial state of E; and, $\underline{E}$ means that the static s-expression E has finished its execution (it can execute no more stochastic multiactions) which is the final state of E. These dynamic s-expressions will be denoted by G, H, G_i, ..., and the set of dynamic s-expressions is called *DynExpr*.

2.2 Operational Semantics

The operational semantics of sPBC is defined in a very similar way to 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, such as step sequence semantics is not allowed, which will be described later.

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 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 that, all dynamic expressions in the same class behave in the same way. Inaction rules for sPBC are those defined in Tables 1 and 2.

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

Table 1. Inaction rules (I)

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.* Thus, a dynamic s-expression G is operative if it is not possible to apply any inaction rule from it. 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. 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 $\equiv$.

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

Table 2. Inaction rules (II)

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

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

which capture all the possible evolutions of each dynamic s-expression. Therefore, by iterating the application of these rules we will obtain sequences of transitions which will correspond to the occurrence sequences of the corresponding stochastic Petri net that we will associate with it.

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 below. We assume that all the dynamic s-expressions that appear on the left-hand side of each transition in the rules are operative.

(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)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G\ rs\ a \stackrel{<\alpha,r>}{\longrightarrow} G'\ rs\ a}\ a,\hat{a} \notin A(\alpha)$	**(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)**	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G \Vert H \stackrel{<\alpha,r>}{\longrightarrow} G' \Vert H}$	**(C2)**	$\dfrac{H \stackrel{<\alpha,r>}{\longrightarrow} H'}{G \Vert H \stackrel{<\alpha,r>}{\longrightarrow} G \Vert H'}$

Table 3. Rules defining the stochastic transitions (I)

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. 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:

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

where J_1^*, J_2^* are operative dynamic s-expressions, and $G_1 \Vert H_1 \equiv G_2 \Vert H_2 \equiv G' \Vert H'$. 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 1. The function: $BC\ :\ OpDynExpr \longrightarrow \mathcal{P}(\mathcal{B}(SL))$, defines the set of *bags of concurrent stochastic multiactions* of an operative dynamic s-expression in the following way:

- If $G \in OpDynExpr$ is final (i.e. G=$\underline{E}$), we take $BC(G) = \emptyset$.
- If $G \in OpDynExpr$ 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 $\gamma_i \in BC(G)$.
- If $\gamma \in BC(G)$, then: $\gamma \in BC(G;E)$, $\gamma \in BC(E;G)$, $\gamma \in BC(E\Box G)$, $\gamma \in BC(G\Box E)$, $\gamma \in BC(G\, rs\, a)$ (when $a, \widehat{a} \notin A(\gamma)$), $\gamma \in BC(G\, sy\, a)$, $f(\gamma) \in BC(G[f])$.
- If $\gamma_1 \in BC(G)$, $\gamma_2 \in BC(H)$, then $\gamma_1 + \gamma_2 \in BC(G\|H)$.
- $\gamma \in BC(G\, sy\, a)$, and $<\alpha, r_1>$, $<\beta, r_2> \in \gamma$, with $a \in A(\alpha)$, and $\widehat{a} \in A(\beta)$, then: $\gamma' \in BC(G\, sy\, a)$, where: $\gamma' = (\gamma + \{<\alpha \oplus_a \beta, min(r_1, r_2)>\}) \setminus \{<\alpha, r_1>, <\beta, r_2>\}$

$\Box$

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

Table 4. Rules for the operator *sy*

Proposition 1. ([11]). Given an operative dynamic s-expression G, $\gamma \in BC(G)$, and any serialization of the stochastic multiactions of γ: $<\alpha_1, r_1> \ldots <\alpha_n, r_n>$, there exists a transition sequence:
$$G \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_1\, (\stackrel{\emptyset}{\longrightarrow})^*\, G_1^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_2\, (\stackrel{\emptyset}{\longrightarrow})^*\, G_2^* \stackrel{<\alpha_3,r_3>}{\longrightarrow} \ldots (\stackrel{\emptyset}{\longrightarrow})^*\, G_{n-1}^* \stackrel{<\alpha_n,r_n>}{\longrightarrow} G'$$
Moreover, all dynamic s-expressions G' thus obtained are equivalent with respect to $\equiv$. $\Box$

Definition 2. 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 derived* from G, and we will denote by $[G\rangle$ the set of all the dynamic s-expressions that can be derived from $[G]_\equiv$. $\Box$

Now, we present 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 3. Let G be a dynamic s-expression. We define the transition system associated with 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
- A is the multiset of transitions, given by $A = \{([H]_\equiv, <\alpha, r>, [J]_\equiv) \mid H \in [G\rangle \wedge H \stackrel{<\alpha,r>}{\longrightarrow} J\}$ $\Box$

Note 1. In order to compute the number of instances of each transition $([H]_\equiv, <\alpha, r>, [J]_\equiv)$ in A, we will 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 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. We can solve this problem by enumerating the stochastic multiactions from left to right, in the syntax of the s-expression. Then, when we apply rule $Sy2$, the new stochastic multiaction will be identify with the concatenation of the numbering of the corresponding stochastic multiactions involved in the synchronization, but if we detect that this numbering has been previously obtained, by a previous application of rule $Sy2$, although in different order, we only consider this transition once.

The *race policy* governs the dynamic behaviour of the system when two or more stochastic multiactions are simultaneously enabled. Consequently, the stochastic process associated to the evolution of each dynamic s-expression $\overline{E}$ is a Continuous Time Markov Chain (CTMC). Thus, once we have the transition system of $\overline{E}$ we can generate the corresponding CTMC: which is done by modifying the multigraph $ts(\overline{E})$ combining in 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.

3 The new proposal for Synchronization.

In order to motivate our new proposal for the synchronization in sPBC, let us look at the following s-expressions:

$$\begin{aligned} E_1 &= <\alpha, r_1> \Box <\alpha, r_2> \Box \ldots \Box <\alpha, r_n> \\ E_2 &= <\alpha, \Sigma_{i=1}^{n} r_i> \end{aligned}$$

According to the race policy that we apply to resolve the choice, we obtain for both dynamic s-expressions, $\overline{E_1}$ and $\overline{E_2}$, the same delay for executing the multiaction $\alpha:\ r = \Sigma_{i=1}^{n} r_i$. In consequence, the CTMCs obtained from the transition systems of $\overline{E_1}$ and $\overline{E_2}$ would be the same, and thus we may consider that both s-expressions are stochastically equivalent. This is captured by the following definition.

Definition 4. Let G_1 and G_2 be two dynamic s-expressions, and their corresponding transition systems $ts(G_1) = (V_1, A_1, v_0^1)$, and $ts(G_2) = (V_2, A_2, v_0^2)$. We will say that G_1 and G_2 are *stochastically equivalent* (we will denote it by $G_1 \sim G_2$) if and only if:

- There exists a bijective function $\phi : V_1 \to V_2$, such that $\phi(v_0^1) = v_0^2$
- For all $(v_1, <\alpha, r>, v_1') \in A_1$ there exists $(\phi(v_1), <\alpha, s>, \phi(v_1')) \in A_2$ such that $r_1(v_1, \alpha, v_1') = r_2(\phi(v_1), \alpha, \phi(v_1'))$

where $r_i(v_i, \alpha, v_i') = \sum \{\!| \ r_j \mid (v_i, <\alpha, r_j>, v_i') \in A_i \ |\!\}$; for $i = 1, 2$. □

Notice that $r_i(v_i, \alpha, v_i')$ is the addition of the rates for all the edges in $ts(G_i)$ from v_i to v_i' labelled by α. As we may have several edges labelled with the same rate, this is the sum corresponding to a multiset.

Therefore, the following dynamic s-expressions will be stochastically equivalent too:

$$\begin{aligned} \overline{E_1} &= \overline{<\{a\}, 2>} \quad \sim \quad \overline{E_2} = \overline{<\{a\}, 1> \Box <\{a\}, 1>} \\ \overline{F_1} &= \overline{<\{\hat{a}\}, 3>} \quad \sim \quad \overline{F_2} = \overline{<\{\hat{a}\}, 1> \Box <\{\hat{a}\}, 2>} \end{aligned}$$

Furthermore, we also have:

$$\overline{E_1} \parallel \overline{F_1} \ \sim\ \overline{E_2} \parallel \overline{F_2}, \qquad \text{but} \quad (\overline{E_1} \parallel \overline{F_1})\ sy\ a \ \not\sim\ (\overline{E_2} \parallel \overline{F_2})\ sy\ a$$

because $(\overline{E_1} \parallel \overline{F_1})\ sy\ a$ can make only one transition labelled by $<\emptyset, 2>$, and $(\overline{E_2} \parallel \overline{F_2})\ sy\ a$ can make four different transitions labelled by $<\emptyset, 1>$.

In order to solve this problem, we propose a new semantics for the synchronization, inspired on the apparent rates introduced in [8] for PEPA, although there are remarkable differences with that proposal, because using that proposal it is not possible to obtain a static translation into Petri nets, and the rates of transitions depend in some cases of the marking of the net (see [14]).

3.1 Operational Semantics

We need to restrict the syntax of sPBC to those terms for which no parallel behaviour appears at the highest level in a choice. This restriction hardly reduces the expresiveness of the language, since we could prefix parallel operators appearing at the highest level of a choice by an empty multiaction, the time of which can be interpreted as the time required for the system to instantiate the new processes. Terms fulfilling this restriction will be called *regular terms*, either in static or dynamic s-expressions, and our new operational semantics is only defined for them. Then, a regular static s-expression E is a static s-expression of sPBC which fulfills:

- $E \neq (E_1 \| E_2) \Box E_3$, for any static s-expressions E_1, E_2, E_3.
- $E \neq E_3 \Box (E_1 \| E_2)$, for any static s-expressions E_1, E_2, E_3.
- $E \neq op(E_1 \| E_2) \Box E_3$, for any static s-expressions E_1, E_2, E_3, and $op \in \{sy,\ rs,\ [.]\}$
- $E \neq E_3 \Box op(E_1 \| E_2)$, for any static s-expressions E_1, E_2, E_3, and $op \in \{sy,\ rs,\ [.]\}$

A dynamic s-expression is regular if the underlying static s-expression is regular.

This restriction is introduced in order to guarantee that, the order in which the rule for the synchronization is applied does not affect the final value that we obtain for the rate of the new stochastic multiaction, (this will be later illustrated with an example).

In order to precisely define this new operational semantics of sPBC we just need to change rule $Sy2$, according to our new proposal for the synchronization. We first need to define for every dynamic s-expression G the multiset of associated conflicts for every instance of a stochastic multiaction [1] $<\alpha, r>_i$ executable from G: $Conflict(G, <\alpha, r>_i)$. We will omit the subindex i, if it is clear from the syntax which instance of $<\alpha, r>$ we are considering.

We also need to introduce the following definition:

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

which will be called *conflict rate* for G and $<\alpha, r>_i$; and we also need to slightly modify the definition of BC for $G\,sy\,a$, in the following way:

– If $\gamma \in BC\,(G\,sy\,a)$ and $<\alpha, r_1>, <\beta, r_2> \in \gamma$, with $a \in A(\alpha)$ and $\hat{a} \in A(\beta)$, then $\gamma' \in BC\,(G\,sy\,a)$, where $\gamma' = (\gamma + \{<\alpha, r_1> \oplus_a <\beta, r_2>\}) \setminus \{<\alpha, r_1>, <\beta, r_2>\}$, being $<\alpha, r_1> \oplus_a <\beta, r_2> = <\alpha \oplus_a \beta, R_{12}>$, with R_{12} defined as follows:
$$R_{12} = \frac{r_1}{cr(G\,sy\,a, <\alpha, r_1>)} \cdot \frac{r_2}{cr(G\,sy\,a, <\beta, r_2>)} \cdot min\,(\,cr(G\,sy\,a, <\alpha, r_1>), cr(G\,sy\,a, <\beta, r_2>)\,)$$

  In this definition we compute as rate of the new stochastic multiaction the minimum of the *conflict rates* of $<\alpha, r_1>, <\alpha, r_2>$ for $G\,sy\,a$, weighted by a factor, which is introduced in order to guarantee that the addition of the probabilities of execution of all the stochastic multiactions in a conflict is 1. It is important to observe that stochastic multiactions involved in a parallel operator will not be considered as in conflict, which is a remarkable difference with the semantics for the synchronization in PEPA [8].

Then, the new rule for the synchronization is shown in Table 5.

$(Sy'\mathbf{2})$ Let $\{<\alpha_1, r_1>, <\alpha_2, r_2>\} \in BC(G\,sy\,a)$, $a \in A(\alpha)$, $\hat{a} \in A(\beta)$, then

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

$$R_{12} = \frac{r_1}{cr(G\,sy\,a, <\alpha_1, r_1>)}\,\frac{r_2}{cr(G\,sy\,a, <\alpha_2, r_2>)} \cdot min\,(cr(G\,sy\,a, <\alpha_1, r_1>), cr(G\,sy\,a, <\alpha_2, r_2>))$$

Table 5. The new proposal for *sy*

Now, we introduce the definition of *Conflict*.

Definition 5. We define the following partial function:

$$Conflict : OpDynExpr\; \times \mathcal{SL} \longrightarrow \mathcal{B}(\mathcal{SL})$$

$Conflict(G, <\alpha, r>_i)$ defines the multiset of the stochastic multiactions in *conflict* with G for the instance i of the stochastic multiaction $<\alpha, r>$, which must be executable from G. We define this function in a structural way, but let us observe that G cannot be final ($G \not\equiv \underline{E}$), because we are assuming that $<\alpha, r>$ is executable from G.

1. $Conflict\,(\overline{<\alpha, r>}, <\alpha, r>) = \{<\alpha, r>\}$

[1] We can differentiate each instance of a stochastic multiaction $<\alpha, r>$ by enumerating their occurrences from left to right in the syntax of G.

2. If $<\alpha, r>$ is executable from G, and *Conflict* $(G, <\alpha, r>) = C$, then:
   (a) *Conflict* $(G; E, <\alpha, r>) =$ *Conflict* $(E; G, <\alpha, r>) = C$,
   (b) *Conflict* $(G \| H, <\alpha, r>) =$ *Conflict* $(H \| G, <\alpha, r>) = C$,
   (c) If $a, \hat{a} \notin A(\alpha)$, then *Conflict* $(G\; rs\; a, <\alpha, r>) = C$,
   (d) For a bijective function f, *Conflict* $(G[f], <f(\alpha), r>) = f(C)$,
   (e) For the choice operator we need to distinguish the following cases:
      i. If $G \not\equiv \overline{E}$, then *Conflict* $(G \Box F, <\alpha, r>) =$ *Conflict* $(F \Box G, <\alpha, r>) = C$,
      ii. If $G \equiv \overline{E}$, then *Conflict* $(G \Box F, <\alpha, r>) =$ *Conflict* $(F \Box G, <\alpha, r>) =$
      $C + \{\!\mid <\alpha, r_j> \mid \exists H_i \in OpDynExpr\,,\ H_i \equiv \overline{F}$ and $H_i \stackrel{<\alpha, r_j>}{\longrightarrow} H_i' \mid\!\}$
   (f) *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 rule $Sy'2$. Then:

   Conflict $(G sy\, a, <\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>) = \{\!\mid\ <\alpha_1 \oplus_a \alpha_2\,,\, R_{ij}> \mid\ <\alpha_1, r_i> \in C_1,\ <\alpha_2, r_j> \in C_2,$

   with $R_{ij} = \frac{r_i}{cr(G sy\, a, <\alpha_1, r_1>)} \frac{r_j}{cr(G sy\, a, <\alpha_2, r_2>)} \cdot min\,(cr(G sy\, a, <\alpha_1, r_1>), cr(G sy\, a, <\alpha_2, r_2>)) \mid\!\}$

   considering: $C_1 =$ *Conflict* $(G sy\, a, <\alpha_1, r_1>)$ and $C_2 =$ *Conflict* $(G sy\, a, <\alpha_2, r_2>)$ $\Box$

Notice that *Conflict*$(G, <\alpha, r>_i)$ is a partial function, it is only defined if $<\alpha, r>_i$ is executable from G, and that it is well defined. The only case requiring some explanations is that of $G\, sy\, a$ with a stochastic multiaction obtained from a synchronization: $<\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>$, since we need to compute $cr(G sy\, a, <\alpha_i, r_i>)$, for $i = 1, 2$. But notice that this is defined by the (3) point of this definition, and thus *Conflict* $(G sy\, a, <\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>)$ is well defined.

Example 1. Let $G sy\, a = \overline{<\{a\}, r_1>} \| \overline{<\{\hat{a}, \hat{a}\}, r_2>} \| (\overline{<\{a\}, r_3>} \Box <\{a\}, r_4>)\ sy\, a$ be a regular operative dynamic s-expression; it follows that $\{<\{a\}, r_1>, <\{\hat{a}, \hat{a}\}, r_2>, <\{a\}, r_3>\} \in BC\,(G sy\, a)$. We may compute:

$$\begin{array}{l} \textit{Conflict}\,(G\, sy\, a, <\{a\}, r_1>) = \{<\{a\}, r_1>\}\ ,\ cr\,(G\, sy\, a, <\{a\}, r_1>) = r_1 \\ \textit{Conflict}\,(G\, sy\, a, <\{\hat{a}, \hat{a}\}, r_2>) = \{<\{\hat{a}, \hat{a}\}, r_2>\}\ ,\ cr\,(G\, sy\, a, <\{\hat{a}, \hat{a}\}, r_2>) = r_2 \\ \textit{Conflict}\,(G\, sy\, a, <\{a\}, r_3>) = \{<\{a\}, r_3>, <\{a\}, r_4>\}\ ,\ cr\,(G\, sy\, a, <\{a\}, r_3>) = r_3 + r_4 \end{array}$$

Applying $Sy'2$ we obtain: $$\frac{G\, sy\, a \stackrel{<\{a\}, r_1>}{\longrightarrow} G_1\, sy\, a \stackrel{<\{\hat{a}, \hat{a}\}, r_2>}{\longrightarrow} G_{12}\, sy\, a}{G\, sy\, a \stackrel{<\{\hat{a}\}, R_{12}>}{\longrightarrow} G_{12}\, sy\, a}$$

where

$$G_1\, sy\, a = \underline{<\{a\}, r_1>} \| \overline{<\{\hat{a}, \hat{a}\}, r_2>} \| (\overline{<\{a\}, r_3>} \Box <\{a\}, r_4>)\ sy\, a,$$

$$G_{12}\, sy\, a = \underline{<\{a\}, r_1>} \| \underline{<\{\hat{a}, \hat{a}\}, r_2>} \| (\overline{<\{a\}, r_3>} \Box <\{a\}, r_4>)\ sy\, a,$$

$$R_{12} = min(r_1, r_2) \quad , \quad \textit{Conflict}\,(G sy\, a, <\{\hat{a}\}, R_{12}>) = \{<\{\hat{a}\}, R_{12}>\}$$

Applying $Sy'2$ again: $$\frac{G\, sy\, a \stackrel{<\{\hat{a}\}, R_{12}>}{\longrightarrow} G_{12}\, sy\, a \stackrel{<\{a\}, r_3>}{\longrightarrow} G_{123}\, sy\, a}{G\, sy\, a \stackrel{<\emptyset, R_{12.3}>}{\longrightarrow} G_{123}\, sy\, a}$$

where

$$G_{123}\, sy\, a = \underline{<\{a\}, r_1>} \| \underline{<\{\hat{a}, \hat{a}\}, r_2>} \| (\underline{<\{a\}, r_3>} \Box <\{a\}, r_4>)\ sy\, a$$

$$R_{12.3} = \frac{R_{12}}{R_{12}} \cdot \frac{r_3}{r_3 + r_4} \cdot min(R_{12}, r_3 + r_4) = \frac{r_3}{r_3 + r_4} \cdot min(r_1, r_2, r_3 + r_4)$$

It is easy to check that for all the alternatives for these evolutions making $<\{a\}, r_1>, <\{\hat{a}, \hat{a}\}, r_2>,$ $<\{a\}, r_3>$, we obtain the same results.

$\Box$

Now we show an example which motivates our syntactical restriction, i.e., it will raise the problem that appear when we consider a parallel behaviour inside a choice.

Example 2. Let $G\,sy\,a = (((\overline{<\{a\},r_1>} \,\|\, \overline{<\{a\},r_3>})\,\Box\, <\{a\},r_4>) \,\|\, \overline{<\{\hat{a},\hat{a}\},r_2>})\,sy\,a$ be a non-regular operative dynamic s-expression. It follows that $\{<\{a\},r_1>, <\{\hat{a},\hat{a}\},r_2>, <\{a\},r_3>\} \in BC\,(G sy\,a)$. Then, according to the definition of BC, we also have $\gamma = \{<\{a\},r_1>, <\{\hat{a}\},R_{23}>\} \in BC\,(G sy\,a)$ where $R_{23} = \frac{r_3}{r_3+r_4} \cdot min(r_2, r_3+r_4)$. However, not every serialization of γ is possible from $G\,sy\,a$, because $<\{\hat{a}\},R_{23}>$ cannot be executed from $G_1\,sy\,a$, where $G\,sy\,a \stackrel{<\{a\},r_1>}{\longrightarrow} G_1\,sy\,a$, and

$$G_1\,sy\,a = (((\underline{<\{a\},r_1>} \,\|\, \overline{<\{a\},r_3>})\,\Box\, <\{a\},r_4>) \,\|\, \overline{<\{\hat{a},\hat{a}\},r_2>})\,sy\,a$$

Actually, we can execute $<\{\hat{a}\},R'_{23}>$ from $G_1\,sy\,a$, with $R'_{23} = min(r_2,r_3)$, and $R_{23} \neq R'_{23}$. $\Box$

The following results show that if we consider a regular term G of sPBC, any serialization of a bag of concurrent stochastic multiactions of G is now possible.

Proposition 2. Let G be a regular operative dynamic s-expression and $\gamma = \{<\alpha_1,r_1>, <\alpha_2,r_2>\} \in BC\,(G)$, with

$$G \stackrel{<\alpha_1,r_1>}{\longrightarrow} H(\stackrel{\emptyset}{\longrightarrow})^* H^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} J\,.$$

Then: $\mathit{Conflict}\,(G, <\alpha_2,r_2>) = \mathit{Conflict}\,(H^*, <\alpha_2,r_2>)$

Proof. (Sketch) By structural induction on the syntax of G:

- **Base case:** $G = G_1 \,\|\, G_2$, where $G_1, G_2 \in OpDynExpr$, $<\alpha_1,r_1>$ is executable from G_1 and $<\alpha_2,r_2>$ is executable from G_2. Thus:

  $$G = G_1 \,\|\, G_2 \stackrel{<\alpha_1,r_1>}{\longrightarrow} G'_1 \,\|\, G_2(\stackrel{\emptyset}{\longrightarrow})^* G'^*_1 \,\|G_2 \stackrel{<\alpha_2,r_2>}{\longrightarrow} G'^*_1 \,\|\, G'_2$$

  Taking $H = G'_1 \,\|\, G_2$, $H^* = G'^*_1 \,\|\, G_2$ and $J = G'^*_1 \,\|\, G'_2$, we obtain:

  $$\begin{array}{c}\mathit{Conflict}\,(G_1 \,\|\, G_2, <\alpha_2,r_2>) = \mathit{Conflict}\,(G_2, <\alpha_2,r_2>) = \\ \mathit{Conflict}\,(G'^*_1 \,\|\, G_2, <\alpha_2,r_2>) = \mathit{Conflict}\,(H^*, <\alpha_2,r_2>)\end{array}$$

- **General case:** A simple application of the induction hypothesis solves the sequential composition, restriction or labelling. For the choice operator, i.e., either $G = G' \,\Box\, F$ or $G = F \,\Box\, G'$, notice that $G' \not\equiv \overline{E}$, for a certain E, due to the syntactical restriction that we have introduced, because we are considering a parallel behaviour in G. Then, we just need to apply the induction hypothesis for G'. Let us now consider $G = G'\,sy\,a$, this case is a bit more involved, and we need to distinguish the following cases:
  - If $\gamma \in BC(G')$: immediate, by applying the induction hypothesis.
  - If $\gamma \notin BC(G')$: In this case at least one of the involved stochastic multiactions has been obtained by using rule $Sy'\,2$. Then, we can proceed by induction on the number of times that this rule has been applied. For simplicity, we only consider the case in which $<\alpha_2,r_2>$ has been obtained by applying $Sy'\,2$, (for $<\alpha_1,r_1>$ we could reason in the same way, and if both have been obtained by using that rule, we could apply a double induction):
    - **Base case** $<\alpha_2,r_2> = <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>$
      Then, $\{<\alpha_1,r_1>, <\alpha'_2,r'_2>, <\alpha''_2,r''_2>\} \in BC\,(G')$, and

      $$G = G'\,sy\,a \stackrel{<\alpha_1,r_1>}{\longrightarrow} H'\,sy\,a(\stackrel{\emptyset}{\longrightarrow})^* H'^*\,sy\,a = H^* \stackrel{<\alpha'_2,r'_2>\oplus_a<\alpha''_2,r''_2>}{\longrightarrow} J'\,sy\,a = J$$

      We apply the induction hypothesis for $\{<\alpha_1,r_1>, <\alpha'_2,r'_2>\}$ and $\{<\alpha_1,r_1>, <\alpha''_2,r''_2>\}$ from G':

      $$\begin{array}{l}\mathit{Conflict}(G', <\alpha'_2,r'_2>) = \mathit{Conflict}(H'^*, <\alpha'_2,r'_2>), \\ \mathit{Conflict}(G', <\alpha''_2,r''_2>) = \mathit{Conflict}(H'^*, <\alpha''_2,r''_2>)\end{array}$$

      Therefore, by definition of *Conflict*:

      $$\mathit{Conflict}(G, <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>) = \mathit{Conflict}(H^*, <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>)$$

    - **General case** We are now supposing that: $<\alpha_2,r_2> = <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>$, where $<\alpha''_2,r''_2>$ is obtained by applying $Sy'\,2$ n-1 times. Then, by the internal induction hypothesis: $\mathit{Conflict}(G', <\alpha''_2,r''_2>) = \mathit{Conflict}(H'^*, <\alpha''_2,r''_2>)$ and by the external induction hypothesis: $\mathit{Conflict}(G', <\alpha'_2,r'_2>) = \mathit{Conflict}(H'^*, <\alpha'_2,r'_2>)$. Therefore,

      $$\mathit{Conflict}(G, <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>) = \mathit{Conflict}(H^*, <\alpha'_2,r'_2> \oplus_a <\alpha''_2,r''_2>)$$

$\square$

Corollary 1. Let G be a regular operative dynamic s-expression and $\gamma = \{<\alpha_1, r_1>, <\alpha_2, r_2>, \ldots, <\alpha_n, r_n>\} \in BC(G)$, with

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

Then: $\quad \mathit{Conflict}\,(G, <\alpha_i, r_i>) = \mathit{Conflict}\,(G_{i-1}^*, <\alpha_i, r_i>)$ $\square$

Corollary 2. Given a regular operative dynamic s-expression G, $\gamma \in BC(G)$, and any serialization of the stochastic multiactions of γ: $<\alpha_1, r_1> . <\alpha_2, r_2> \ldots <\alpha_n, r_n>$ there exists a transition sequence:

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

with $\mathit{Conflict}\,(G, <\alpha_i, r_i>) = \mathit{Conflict}\,(G_{i-1}^*, <\alpha_i, r_i>)$, for $i = 2, \ldots, n$.
Moreover, all dynamic s-expressions G' thus obtained are equivalent with respect to $\equiv$.

Proof. It is an immediate consequence of the previous corollary, and Prop.1 . $\square$

Lemma 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 the rule $Sy'2$, thus obtaining a transition for each case: $G \stackrel{<\beta_i,R_i>}{\longrightarrow} G_n$ and $G \stackrel{<\beta_j,R_j>}{\longrightarrow} G'_n$. Then: $G_n \equiv G'_n$ and $<\beta_i, R_i> = <\beta_j, R_j>$.

Proof. (Sketch) We obtain $G_n \equiv G'_n$ from the previous corollary, and $\beta_i = \beta_j$ from the definition of the synchronization operator on multiactions. In order to show that $R_i = R_j$ we can proceed by induction on the number (n) of stochastic multiactions in γ.
Base case: $n = 2$, we apply the rule $Sy'2$ once: $\{<\alpha_1, r_1>, <\alpha_2, r_2>\} \in BC(G)$

$$G \stackrel{<\alpha_1,r_1>}{\longrightarrow} G_1 (\stackrel{\emptyset}{\longrightarrow})^* G_1^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_{12}$$
$$G \stackrel{<\alpha_2,r_2>}{\longrightarrow} G_2 (\stackrel{\emptyset}{\longrightarrow})^* G_2^* \stackrel{<\alpha_2,r_2>}{\longrightarrow} G'_{12}$$

Then $R_{12} = R_{21} = \frac{r_1}{rc\,(G,<\alpha_1,r_1>)} \cdot \frac{r_2}{rc\,(G,<\alpha_2,r_2>)} \cdot min(rc\,(G, <\alpha_1, r_1>), rc\,(G, <\alpha_2, r_2>)$

General case: We assume that the property holds for $n-1$, so let us consider the application of $Sy'2$, for example in the following case (the remaining cases are analogous):

$$(<\alpha_1, r_1> \oplus_a \ldots \oplus_a <\alpha_{k-1}, r_{k-1}>) \oplus_a (<\alpha_k, r_k> \oplus_a \ldots \oplus_a <\alpha_n, r_n>)$$

Then, we may apply the induction hypothesis for boht part, and according to our definition for the synchronization, the rate for the new stochastic multiaction can be computed as follows:

$$\frac{R_{1\ldots k-1}}{rc(G, <\alpha_1, r_1> \oplus_a \ldots \oplus_a <\alpha_{k-1}, r_{k-1}>)} \cdot \frac{R_{k\ldots n}}{rc(G, <\alpha_k, r_k> \oplus_a \ldots \oplus_a <\alpha_n, r_n>)} \cdot$$
$$min(rc(G, <\alpha_1, r_1> \oplus_a \ldots \oplus_a <\alpha_{k-1}, r_{k-1}>), rc(G, <\alpha_k, r_k> \oplus_a \ldots \oplus_a <\alpha_n, r_n>)$$
$$= \frac{r_1}{rc\,(G, <\alpha_1, r_1>)} \cdots \frac{r_n}{rc\,(G, <\alpha_n, r_n>)} \cdot min(rc\,(G, <\alpha_1, r_1>), \ldots, rc\,(G, <\alpha_n, r_n>))$$

In consequence, we would obtain the same result for any serialization of γ. $\square$

So, it does not matter the order to obtain the synchronizations (if they are possible). And from this, in the transition system we only consider one of these possibilities.

Proposition 3. Given a regular operative dynamic s-expression G, $<\alpha, r>$ a stochastic multiaction executable from G , and $\mathit{Conflict}(G, <\alpha, r>) = \{<\alpha, r>, <\alpha, r_1>, \ldots, <\alpha, r_n>\}$.
Then, for each $<\alpha, r_i> \in \mathit{Conflict}(G, <\alpha, r>)$, there is another regular operative dynamic s-expression G_i, where $G_i \equiv G$, $<\alpha, r_i>$ is executable from G_i , and $\mathit{Conflict}(G_i, <\alpha, r_i>) = \mathit{Conflict}(G, <\alpha, r>)$.

Proof. (Sketch) By structural induction on the syntax of G.

- **Base case:** $G = \overline{<\alpha, r>}$ and it is immediate.
- **General case:** A simple application of the induction hypothesis solves the sequential composition, restriction or labelling.
  For the case of the choice operator, i.e., either $G = G_1 \,\Box\, F$ or $G = F \,\Box\, G_1$, we need to distinguish two cases:
  - If $G_1 \not\equiv \overline{E}$, for any regular static s-expression E, we just need to apply the induction hypothesis.
  - If $G_1 \equiv \overline{E}$ for some regular static s-expression E, we also distinguish:
    - If $<\alpha, r_i> \in \mathit{Conflict}(G_1, <\alpha, r>)$, then we can apply the induction hypothesis too.
    - If $<\alpha, r_i> \notin \mathit{Conflict}(G_1, <\alpha, r>)$, then there exists H_i, operative, $H_i \equiv \overline{F}$, such that $<\alpha, r_i>$ is the only stochastic multiaction executable from H_i (due to the syntactical restriction), so if we consider $G_i = E \,\Box\, H_i$, it follows that G_i is operative, $G_i \equiv G$ and $\mathit{Conflict}(G, <\alpha, r>) = \mathit{Conflict}(G_i, <\alpha, r_i>)$.

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 $<\alpha, r>$ is executable from G_1, then we can apply the induction hypothesis.
- In other case, if $<\alpha, r>$ comes from the application of rule $Sy'\,2$, we can proceed by induction on the number of times that the rule $Sy'\,2$ has been applied.
  - **Base case:** $<\alpha, r>$ comes from the application of rule $Sy'\,2$ once. If we consider:
    $$<\alpha, r> = <\alpha_1, r'> \oplus_a <\alpha_2, r''>\,,$$
    Then $\{<\alpha_1, r'>\,,\, <\alpha_2, r''>\} \in BC\,(G_1 \; sy \; a)$, and $<\alpha, r>$ has been obtained applying rule $Sy'2$:
    $$G_1 \; sy \; a \stackrel{<\alpha_1, r'>}{\longrightarrow} G_1^1 \; sy \; a (\stackrel{\emptyset}{\longrightarrow})^* \, G_1^{1^*} \; sy \; a \stackrel{<\alpha_2, r''>}{\longrightarrow} G_1^{12} \; sy \; a$$
    Considering:
    $$\begin{array}{l} C_1 \;=\; \mathit{Conflict}\,(G_1 \; sy \; a, <\alpha_1, r'>) \;=\; \mathit{Conflict}\,(G_1, <\alpha_1, r'>) \\ C_2 \;=\; \mathit{Conflict}\,(G_1 \; sy \; a, <\alpha_2, r''>) \;=\; \mathit{Conflict}\,(G_1^{1^*} \; sy \; a, <\alpha_2, r''>) \;= \\ \mathit{Conflict}\,(G_1^{1^*}, <\alpha_2, r''>) \;=\; \mathit{Conflict}\,(G_1, <\alpha_2, r''>) \end{array}$$
    If $<\alpha, r_i> \in \mathit{Conflict}\,(G, <\alpha, r>)$, then $<\alpha, r_i> = <\alpha_1, r_i'> \oplus_a <\alpha_2, r_i''>$ with $<\alpha_1, r_i'> \in C_1$ and $<\alpha_2, r_i''> \in C_2$, and applying the induction hypothesis on G_1, considering both stochastic multiactions: $\exists\, G_{1_i}$ operative, $G_{1_i} \equiv G_1$, such that $\{<\alpha_1, r_i'>, <\alpha_2, r_i''>\} \in BC\,(G_{1_i})$ with:
    $$\mathit{Conflict}\,(G_{1_i}, <\alpha_1, r_i'>) \;=\; C_1, \quad \mathit{Conflict}\,(G_{1_i}, <\alpha_2, r_i''>) \;=\; C_2$$
    And due to the definition of *Conflict* and considering $G_i \;=\; G_{1_i} \; sy \; a$, it follows that G_i is operative, $G_i \equiv G$, $<\alpha, r_i>$ is executable from G_i and
    $$\mathit{Conflict}\,(G_i, <\alpha, r_i>) \;=\; \mathit{Conflict}\,(G, <\alpha, r>)$$
  - **General case:** $<\alpha, r>$ has been obtained applying $Sy'\,2$ n times. Then:
    $$<\alpha, r> = <\alpha_1, r'> \oplus_a <\alpha_2, r''>\,,$$
    where $<\alpha_1, r'>$ has been obtained applying $Sy'\,2$ k times, and $<\alpha_2, r''>$ has been obtained applying $Sy'\,2$ n-k-1 times. Both for $<\alpha_1, r'>$ and $<\alpha_2, r''>$ we can apply the induction hypothesis, and then we can reason as in the base case to conclude the proof.

□

Let us observe that we construct the labelled transition system of any regular dynamic s-expression G in the same way as in Def. 3, i.e., we remove *redundant transitions*, using the same procedure as in Note 1. The race policy is still applied in order to decide which stochastic multiaction is executed when several of them are simultaneously possible, and we can also define the CTMC of G in the same way.

Example 3. Let us recall our introductory example:
$$\begin{array}{rcl} \overline{E_1} = \overline{<\{a\}, 2>} & \sim & \overline{E_2} = \overline{<\{a\}, 1> \Box <\{a\}, 1>} \\ \overline{F_1} = \overline{<\{\hat{a}\}, 3>} & \sim & \overline{F_2} = \overline{<\{\hat{a}\}, 1> \Box <\{\hat{a}\}, 2>} \end{array}$$
where $\overline{E_1} \parallel \overline{F_1} \;\sim\; \overline{E_2} \parallel \overline{F_2}$. Now, according to our new proposal, we also have:
$$(\overline{E_1} \parallel \overline{F_1}) \; sy \; a \;\sim\; (\overline{E_2} \parallel \overline{F_2}) \; sy \; a$$
because $(\overline{E_1} \parallel \overline{F_1}) \; sy \; a$ can make only one transition labelled by $<\emptyset, 2>$, and $(\overline{E_2} \parallel \overline{F_2}) \; sy \; a$ can make the following four different empty transitions, for which the sum of the rates is 2: $<\emptyset, \frac{1}{3}>$, $<\emptyset, \frac{2}{3}>$, $<\emptyset, \frac{1}{3}>$, $<\emptyset, \frac{2}{3}>$. □

3.2 Denotational Semantics

Now, 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 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 $\mathcal{B}(\mathcal{L}) \times \mathcal{L}$. 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 $\forall t \in T : {}^\bullet t \neq \emptyset \neq t^\bullet \;\wedge\; {}^\bullet t \cap t^\bullet = \emptyset$. $\lambda(t)$ 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 $\mathcal{L}$.

Definition 6. 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. 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.

Algebra of s-boxes Now we are going to adapt the denotational semantics that we introduced in [11] for the new proposal of synchronization. We will preserve the semantics objects (the s-boxes), but now we will need more information, concretely for each transition that we obtain compositionaly, we need to know which transitions are in *conflict* with it, in order to compute its *conflict rate*. That can be easier if we enumerate the stochastic multiactions from left to right, in the syntax of the regular static s-expression, and we preserve this enumeration in the corresponding transitions of the Petri Net. Only new transitions (that they do not correspond to stochastic multiactions in the syntax of the regular static s-expression), can appear in the case of synchronization. They will be enumerate with the concatenation of the numbering of the transitions (stochastic multiactions) that it comes from.

In order to define the semantic function that associates a plain s-box with every regular term of finite 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 number of transition t according to our criterion (enumeration from left to right and concatenation in case of synchronization), and $\kappa(t)$ is 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 denote it by $cr(t)$:

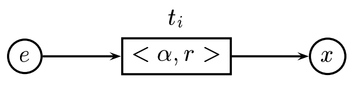
$$cr(t) = \sum_{\eta(t_j) \in \kappa(t)} \mu(t_j)$$

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 also to define η, κ for every operator of finite 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 finite 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:[2]

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

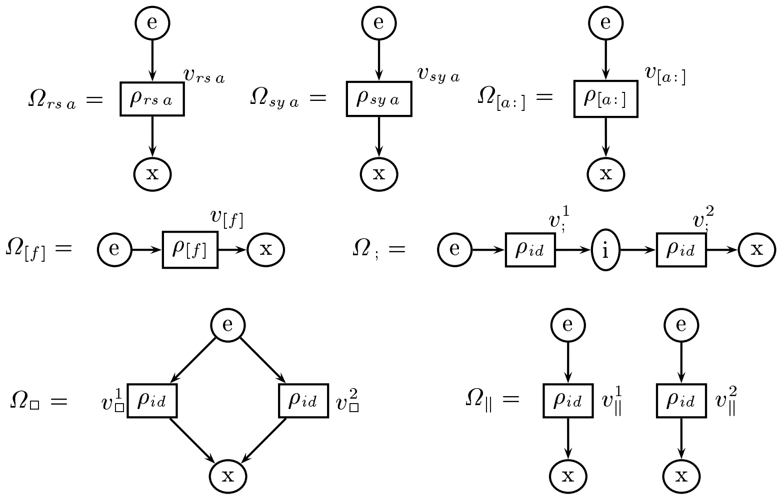


Fig. 1. The operator s-boxes for finite sPBC

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 the enumeration and conflict function for E_i, $i = 1, 2$.

– $Box_s(E_1 \,;\, E_2) = \Omega_{;}(Box_s(E_1), Box_s(E_2))$. Notice that this definition does not introduce any new conflicts, and neither new transitions. Then we take:

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

– $Box_s(E_1 \,\|\, E_2) = \Omega_{\|}(Box_s(E_1), Box_s(E_2))$. Similar to the previous one, η and κ are defined in exactly the same way.

– $Box_s(E_1[f]) = \Omega_{[f]}\,(Box_s\,(E_1)\,)$. Since we are only changing the label of some multiactions and we only consider bijective functions, we are preserving both the conflicts and the enumeration, thus:

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

– $Box_s(E_1 \ rs\, a) = \Omega_{rs\,a}\,(Box_s\,(E_1)\,)$. With this definition we remove all transitions labelled with a multiaction including the action a (or $\hat{a}$), thus, this does not affect the enumeration of the remaining transitions, neither does the conflict set of the remaining transitions, because let us remember that all transitions in $\kappa(t)$ must be labelled with the same multiaction.

$$\eta(t) = \eta_1(t)\ , \quad t \in T_1\ , \ a, \hat{a} \notin \lambda(t) \qquad \text{and} \qquad \kappa(t) = \kappa_1(t)\ , \quad t \in T_1\ , \ a, \hat{a} \notin \lambda(t)$$

– $Box_s(E_1 \,\Box\, E_2) = \Omega_{\Box}(Box_s(E_1), Box_s(E_2))$. No new transitions are introduced with this operator, so the enumeration of transitions is preserved. However, it is clear that this operator will introduce some new conflicts. Specifically, those transitions in T_1 having their preconditions in ${}^{\circ}Box_s(E_1)$ are in conflict with those transitions in T_2 with preconditions in ${}^{\circ}Box_s(E_2)$. Since we are working with regular terms $Box_s(E_1)$ and $Box_s(E_2)$ will have a single entry place, and thus $Box_s(E_1 \,\Box\, E_2)$ will have a single entry place too. Formally:

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

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

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

Notice that $\kappa(t)$ is well defined for the first and fourth cases, because $\kappa_2(t')$ coincide for every $t' \in T_2$, ${}^\bullet t' \in {}^\circ Box_s(E_2)$, $\lambda(t) = \lambda(t')$, and respectively for the other case.

- $Box_s(E_1\ sy\ a) = \Omega_{sy\ a}\,(Box_s\,(E_1)\,)$. We take the following relation for the synchronization: $\rho_{sy\ a} \subseteq \mathcal{B}(\mathcal{L}) \times \mathcal{L}$, as the least relabelling relation containing ρ_{id} 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:

$$\frac{\mu_1(t_1)}{rc(t_1)} \cdot \frac{\mu(t_2)}{rc(t_2)} \cdot min(rc(t_1), rc(t_2))$$

Moreover,

$$\begin{aligned} &\eta(t) = \eta_1(t_1)\,.\,\eta_1(t_2) \\ &\kappa(t) = \kappa_1(t_1) \otimes \kappa_1(t_2) \ = \ \{n_1\,.\,n_2 \,|\, n_1 \in \kappa_1(t_1)\,,\, n_2 \in \kappa_2(t_2)\,\} \end{aligned}$$

In order not to introduce redundant transitions, we only consider in the plain s-box a single one of the possible transitions with the same numbering obtained by concatenation, but it is possible in different order.

Furthermore, those transitions that were in T_1 have the same label, rate, enumeration and conflict as they had in $Box_s(E_1)$.

Example 4. Let $E = (<\{a\}, r_1> \,\|\, <\{\hat{a}, \hat{a}\}, r_2> \,\|\, (<\{a\}, r_3> \Box <\{a\}, r_4>)\,)\ sy\ a$ be a regular static s-expression. Its corresponding plain s-box is depicted in Fig. 2.

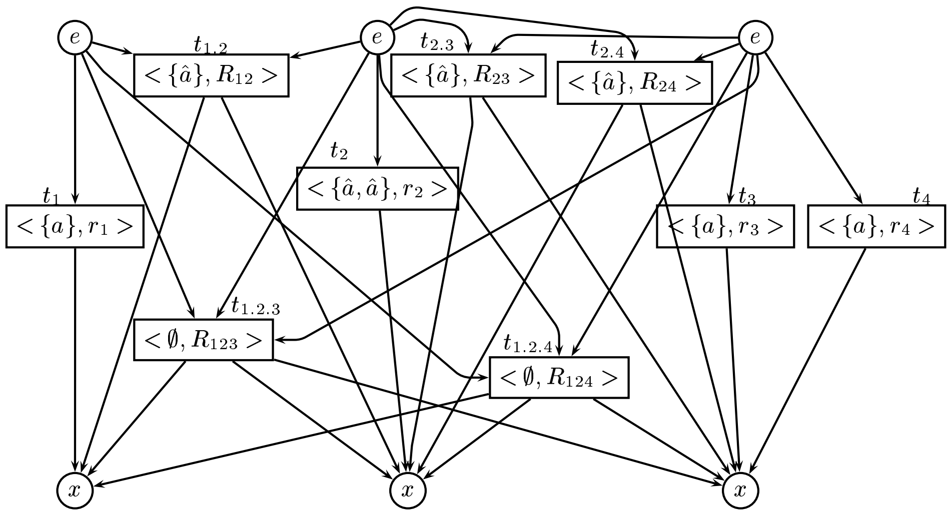


Fig. 2. s-box corresponding to E

where:
$$\begin{array}{lll} \eta(t_1) = 1, & \kappa(t_1) = \{1\}, & \eta(t_2) = 2, \quad \kappa(t_2) = \{2\}, \\ \eta(t_3) = 3, & \kappa(t_3) = \{3, 4\}, & \eta(t_4) = 4, \quad \kappa(t_4) = \{3, 4\}, \\ \eta(t_{1.2}) = 1.2, & \kappa(t_{1.2}) = \{1.2\}, & R_{12} = min(r_1, r_2) \\ \eta(t_{2.3}) = 2.3, & \kappa(t_{2.3}) = \{2.3, 2.4\}, & R_{23} = \frac{r_3}{r_3+r_4} \cdot min(r_2, r_3 + r_4) \\ \eta(t_{2.4}) = 2.4, & \kappa(t_{2.4}) = \{2.3, 2.4\}, & R_{24} = \frac{r_4}{r_3+r_4} \cdot min(r_2, r_3 + r_4) \\ \eta(t_{1.2.3}) = 1.2.3, & \kappa(t_{1.2.3}) = \{1.2.3, 1.2.4\}, & R_{123} = \frac{r_3}{r_3+r_4} \cdot min(r_1, r_2, r_3 + r_4) \\ \eta(t_{1.2.4}) = 1.2.4, & \kappa(t_{1.2.4}) = \{1.2.3, 1.2.4\}, & R_{124} = \frac{r_4}{r_3+r_4} \cdot min(r_1, r_2, r_3 + r_4) \end{array}$$ □

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 to $\overline{E}$, and the reachibility 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 a total order 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. $\square$

4 Conclusions and Future Work

We have introduced a new proposal for the synchronization in a stochastic extension of finite PBC (sPBC), which preserves the main features of PBC. We have defined both an operational and a denotational semantics for this new version of sPBC, and we have seen that they are equivalent.

Our current work is focussed on the definition of a stochastic equivalence relation on sPBC, with the intention of keeping (as far as possible) the classical laws of PBC.

On the other hand, we also intend to extend our results to the infinite case, by including both iteration and recursion. Finally, we will introduce additional capabilities, such as urgent multiactions, priorities, and so on.

References

1. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Wiley, 1995.
2. J.C.M. Baeten. *The Total Order Assumption.* In Proceedings Workshop *"What good is partial order"*, Sheffiel (E. Best, ed.), Hildesheimer Informatik-Berichte 13/92, Universitat Hildesheim 1992, pp. 1-11.
3. E. Best, R. Devillers, and M. Koutny. *A Consistent Model for Nets and Process Algebras,* in the book *The Handbook on Process algebras, Chapter 14, pag. 873-944* (Bergstra, J.A. and Ponse, A. and Smolka, S.S.,eds)*,* North Holland, 2001.
4. E. Best, R. Devillers, and M. Koutny. *Petri Nets, Process Algebras and Programming Languages.* Lectures on Petri Nets II: Applications, W. Reisig and Rozenberg (eds.), Advances in Petri Nets, 1-84, Springer-Verlag, 1998.
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.* Advances in Petri Nets. Springer, LNCS, 609: 21-69.
7. 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.
8. J. Hillston. *A Compositional Approach to Performance Modelling.* PhD thesis, Dep. of Computer Science, University of Edinburgh, April 1994.
9. 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, pp. 303-322, 2000.
10. M. Koutny, J. Esparza, and E. Best. *Operational Semantics for the Petri Box Calculus.* Proc. of CONCUR'94 (ed. B. Jonsson and J. Parrow). Springer, LNCS, 836:210–225, 1994.
11. H. Macià, V. Valero, and D. de Frutos-Escrig. *sPBC: A Markovian Extension of Finite Petri Box Calculus.* Proc. 9th Int. Workshop on Petri Nets and Performance Models (IEEE), 2001 pp. 207-216.
12. O. Marroquín and D. de Frutos. *Extending the Petri Box calculus with Time.* Proc. Int. Conf. on Theory and Application of Petri Nets 2001. Springer, LNCS. Vol. 2075, 2001.
13. R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.
14. M. Ribaudo. *Stochastic Petri Net Semantics for Stochastic Process Algebra.* Proc. 6th Int. Workshop on Petri Nets and Performance Models, Durham, 1995.