A New Synchronization in Finite Stochastic Petri Box Calculus *

Hermenegilda Macià Valentín Valero Fernando Cuartero 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, FernandoL.Pelayo*}*@uclm.es*

Abstract

The Petri Box Calculus (PBC) combines two well known paradigms of the design of concurrent systems: process algebras and Petri nets. In our first proposal of sPBC (stochastic PBC) [12] we defined a Markovian extension of finite PBC, i.e., we had a Markovian process algebra for which both an operational and a denotational (based on stochastic Petri nets) semantics were defined. Our goal in this paper is to improve the semantics for the synchronization operator, in order to define a stochastic equivalence relation.

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

1. Introduction

The Petri Box Calculus (PBC) [3, 4, 5, 6, 7] is a model used for the description of concurrent systems that has, for instance, the algebraic advantages of the process algebras and the structural advantages of Petri nets. This algebraic model 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 [14], 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 [10] and TPBC [13]. 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 [12], 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 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 (see [9]). We con-

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

cluded 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. With this new proposal for the synchronization we will be able to define an adequate stochastic equivalence relation.

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 [12]). 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. From now onwards we will use the following notation:

- $\mathcal{A}$ will be a countable set of action names. For every $a \in \mathcal{A}$, we have the corresponding *conjugate* action, $\widehat{a} \in \mathcal{A}$, such that $a \neq \widehat{a}$ and $\widehat{\widehat{a}} = a$, as in CCS [14]. 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.

- We define the alphabet of $\alpha \in \mathcal{L}$ by: $A(\alpha) =_{def} \{a \in \mathcal{A} \,|\, \alpha(a) > 0\}$

- Relabelling functions $f : \mathcal{A} \to \mathcal{A}$, which preserve conjugates, i.e.: $\forall a \in \mathcal{A},\ f(\widehat{a}) = \widehat{f(a)}$. 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.

- 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 $a \in \beta$.

2.1.1 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:

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

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, $\parallel$, 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 $[a : E] = (E\ sy\ a)\ rs\ a$.

2.1.2 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.

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

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 similar way to PBC.

2.2.1 Inaction rules and structural equivalence

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

$$\overline{E \parallel F} \xrightarrow{\emptyset} \overline{E} \parallel \overline{F}$$

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.

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

Table 1. Inaction rules (I)

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

Table 2. Inaction rules (II)

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. The set of the operative dynamic s-expressions will be denoted by *OpDynExpr*.

Example 1 Let $E = (<\{a\},5>;E_1) \Box (<\{a\},5>;E_2)$, where E_1, E_2 are static s-expressions, then $G = \overline{E}$ is not operative, and we can reach two different operative dynamic s-expressions from G:

$$G_1 = \overline{<\{a\},5>};E_1 \Box (<\{a\},5>;E_2)$$

$$G_2 = (<\{a\},5>;E_1) \Box \overline{<\{a\},5>};E_2$$

□

2.2.2 Rules defining the stochastic transitions

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

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

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

Table 3. Stochastic transition rules (I)

2.2.3 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. 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 following function

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

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=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: $\gamma \in BC(G;E)$, $\gamma \in BC(E;G)$, $\gamma \in BC(E \Box G)$, $\gamma \in BC(G \Box E)$, $\gamma \in BC(G\, rs\, a)$ (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 $\gamma_1 \in BC(G \| H)$, $\gamma_2 \in BC(G \| H)$, $\gamma_1 + \gamma_2 \in BC(G \| H)$.

– $\gamma \in BC(G\, sy\, a)$, 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:
$\gamma' = (\gamma + \{<\alpha \oplus_a \beta, min(r_1, r_2)>\}) \setminus \{<\alpha, r_1>, <\beta, r_2>\}$

□

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 $a \in A(\alpha_1), \widehat{a} \in A(\alpha_2)$:

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

Table 4. Rules for the *sy* operator

Proposition 1 [12] 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:

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

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

Proof: By structural induction, taking into account that G is operative. □

Definition 2 Given two dynamic s-expressions G, H, such that $G \equiv G'$ and $H' \equiv H$, with

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

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 $[G]_{\equiv}$. □

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

□

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 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 identified 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:

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

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 α, $r = \Sigma_{i=1}^{n} r_i$. 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 (multi)-transition systems:

$$ts(G_1) = (V_1, A_1, v_0^1)$$
$$ts(G_2) = (V_2, A_2, v_0^2)$$

We will say that G_1 and G_2 are *stochastically equivalent* ($G_1 \sim G_2$) if and only if:

- There exists a bijective function $\phi : V_1 \rightarrow V_2$, 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 $\varphi_1(v_1, \alpha, v_1') = \varphi_2(\phi(v_1), \alpha, \phi(v_1'))$.
- For all $(v_2, <\alpha, r>, v_2') \in A_2$ there exists $(\phi^{-1}(v_2), <\alpha, s>, \phi^{-1}(v_2')) \in A_1$ such that $\varphi_2(v_2, \alpha, v_2') = \varphi_1(\phi^{-1}(v_2), \alpha, \phi^{-1}(v_2'))$.

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

Notice that $\varphi_i(v_i, \alpha, v_i')$ 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.
Then, the following dynamic s-expressions will be stochastically equivalent:

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

Furthermore, we also have:

$$\overline{E_1} \,\|\, \overline{F_1} \ \sim\ \overline{E_2} \,\|\, \overline{F_2},$$
$$(\overline{E_1} \,\|\, \overline{F_1})\, sy\, a \ \not\sim\ (\overline{E_2} \,\|\, \overline{F_2})\, sy\, a$$

because $(\overline{E_1} \,\|\, \overline{F_1})\, sy\, a$ can make only one transition labelled by $<\emptyset, 2>$, and $(\overline{E_2} \,\|\, \overline{F_2})\, sy\, a$ 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 on the marking of the net (see [15]).

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 slightly reduces the expressiveness of the language, since we could prefix parallel operators appearing at the highest level of 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 regular static s-expressions E are those static s-expressions of sPBC fulfilling:

$$\begin{array}{rcl} D & ::= & <\alpha, r> \mid D; E \mid D\, sy\, a \mid D\, rs\, a \mid D[f] \mid \\ & & [a : D] \mid D \,\Box\, D \\ E & ::= & <\alpha, r> \mid E; E \mid E\, sy\, a \mid E\, rs\, a \mid E[f] \mid \\ & & [a : E] \mid E \,\|\, E \mid D \,\Box\, D \end{array}$$

A dynamic s-expression is regular if the underlying static s-expression is regular, and the set of the regular operative dynamic s-expressions will be denoted by ***ReOpDynExpr***.

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 regular operative 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 define the *conflict rate* for G and $<\alpha, r>_i$, denoted by $cr(G, <\alpha, r>_i)$, as follows:

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

Finally, we slightly modify the definition of BC for $G\, sy\, a$, in the following way:

- If $\gamma \in BC(G\, sy\, a)$ and $<\alpha_1, r_1>, <\alpha_2, r_2> \in \gamma$ (with either $<\alpha_1, r_1> \neq <\alpha_2, r_2>$ or they are two different instances of the same stochastic multiaction in γ), with $a \in A(\alpha_1)$ and $\hat{a} \in A(\alpha_2)$, then:

$$\gamma' \in BC(G\, sy\, a), \quad \text{where :}$$

[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.

$$\gamma' = (\gamma + \{<\alpha_1, r_1> \oplus_a <\alpha_2, r_2>\}) \setminus \{<\alpha_1, r_1>, <\alpha_2, r_2>\}$$

being $<\alpha_1, r_1> \oplus_a <\alpha_2, r_2> = <\alpha_1 \oplus_a \alpha_2, R>$, with R defined as follows:

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

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. 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' 2) 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

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

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

Table 5. The new proposal for *sy*

$Conflict(G, <\alpha, r>_i)$ is 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. It is defined as follows:

Definition 5 We define the following partial function:

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

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. $Conflict\,(\overline{<\alpha, r>}, <\alpha, r>) = \{<\alpha, r>\}$
2. If $<\alpha, r>$ is executable from G, and $C = Conflict\,(G, <\alpha, r>)$, 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:
   - If $G \not\equiv \overline{E}$, $Conflict\,(G \,\square\, F, <\alpha, r>) = Conflict\,(F \,\square\, G, <\alpha, r>) = C$
   - If $G \equiv \overline{E}$, $Conflict\,(G \,\square\, F, <\alpha, r>) = Conflict\,(F \,\square\, G, <\alpha, r>) = C + \{\!\{ <\alpha, r_j> \mid \exists\, H_i \in OpDynExpr\,, H_i \equiv \overline{F} \text{ and } H_i \xrightarrow{<\alpha, r_j>} H_i' \}\!\}$
   (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 $G\,sy\,a \xrightarrow{<\alpha_1 \oplus_a \alpha_2, R_{12}>} G'\,sy\,a$ obtained by rule $Sy'2$. Then:

$$Conflict(G\,sy\,a, <\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>) = \{\!\{ <\alpha_1 \oplus_a \alpha_2\,,\, R_{ij}> \mid <\alpha_1, r_i> \in C_1,\, <\alpha_2, r_j> \in C_2, \text{ with } R_{ij} = \frac{r_i}{cr(Gsy\,a, <\alpha_1, r_1>)} \cdot \frac{r_j}{cr(Gsy\,a, <\alpha_2, r_2>)} \cdot \min_{i=1,2} \{cr(G\,sy\,a, <\alpha_i, r_i>)\}\!\}$$

considering $C_i = Conflict(G\,sy\,a, <\alpha_i, r_i>)$, $i = 1, 2$. □

Notice that $Conflict(G, <\alpha, r>_i)$ is a partial function, it is only defined if $<\alpha, r>_i$ is executable from G, and 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 $cr(Gsy\,a, <\alpha_i, r_i>)$, for $i = 1, 2$. But notice that this is a well-founded recursive definition, and thus $Conflict\,(Gsy\,a, <\alpha_1 \oplus_a \alpha_2\,,\, R_{12}>)$ is well defined.

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

Example 2 Let us consider the following non-regular and operative dynamic s-expression:

$$G = (((\overline{<\{a\}, r_1>} \| \overline{<\{a\}, r_3>}) \square <\{a\}, r_4>) \| \overline{<\{\hat{a}, \hat{a}\}, r_2>})\; sy\; a$$

It follows that

$$\{<\{a\}, r_1>, <\{\hat{a}, \hat{a}\}, r_2>, <\{a\}, r_3>\} \in BC(G)$$

Then, according to the definition of *BC*, we also have

$$\gamma = \{<\{a\}, r_1>, <\{\hat{a}\}, R_{23}>\} \in BC(G)$$

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, because $<\{\hat{a}\}, R_{23}>$ cannot be executed from G_1, where $G \xrightarrow{<\{a\}, r_1>} G_1$ and

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

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

The following results show that if we consider a regular and operative term G of sPBC, any serialization of a bag of concurrent stochastic multiactions of G is possible, and the multiset of conflicts for any stochastic multiaction executable from G is preserved along the chain of operative dynamic s-expressions obtained. All proofs can be found in [11].

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: $Conflict(G, <\alpha_2, r_2>) = Conflict(H^*, <\alpha_2, r_2>)$ □

Corollary 1 Take 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:

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

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

The following results shows that it does not matter the order in which the different multiactions that are synchronized are executed. And from this, in the transition system we only consider one of these possibilities. Let us observe that we construct the labelled (multi)-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.

Proposition 3 Let G be a regular operative dynamic s-expression, $\gamma = \{<\alpha_1, r_1>, \ldots, <\alpha_n, r_n>\} \in BC(G)$, and a serialization of γ:

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

for which we may apply $n-1$ times rule $Sy'2$ thus obtaining a single transition:

$$G \stackrel{<\beta,R>}{\longrightarrow} G_n$$

Then,

$$R = (\prod_{k=1}^{n} \frac{r_k}{cr(G, <\alpha_k, r_k>)}) \cdot \min_{k=1,\ldots,n} \{cr(G\, sy\, a, <\alpha_k, r_k>)\}$$

Furthermore,

$$cr(G, <\beta, R>) = \min_{k=1,\ldots,n} \{\, cr(G\, sy\, a, <\alpha_k, r_k>)\}$$ □

Corollary 2 Let G be a regular operative dynamic s-expression, $\gamma = \{<\alpha_1, r_1>, \ldots, <\alpha_n, r_n>\} \in BC(G)$, and two permutations of the set $\{1, .., n\} : \{i_1, \cdots, i_n\}$ 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 *Sy'2* (for the same actions $a_1, \ldots, a_{n-1}$ possibly repeated, but the same number of times for both cases), thus obtaining a single 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 $<\beta_i, R_i> = <\beta_j, R_j>$. □

Example 3 Let us recall our introductory example:

$$\begin{array}{lcl} \overline{E_1} = \overline{<\{a\}, 2>} & \sim & \overline{E_2} = \overline{<\{a\}, 1> \,\square\, <\{a\}, 1>} \\ \overline{F_1} = \overline{<\{\hat{a}\}, 3>} & \sim & \overline{F_2} = \overline{<\{\hat{a}\}, 1> \,\square\, <\{\hat{a}\}, 2>} \end{array}$$

where $\overline{E_1} \,\|\, \overline{F_1} \ \sim \ \overline{E_2} \,\|\, \overline{F_2}$. Now, according to our new proposal, we also have:

$$(\overline{E_1} \,\|\, \overline{F_1})\, sy\, a \ \sim \ (\overline{E_2} \,\|\, \overline{F_2})\, sy\, a$$

because $(\overline{E_1} \,\|\, \overline{F_1})\, sy\, a$ can make only one transition labelled by $<\emptyset, 2>$, and $(\overline{E_2} \,\|\, \overline{F_2})\, sy\, a$ 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 with a single token. Thus, a plain s-box is essentially a kind of labelled stochastic Petri net, whose behaviour follows the classical *firing rule* of SPNs.

We will say that a safe s-box Σ is *clean* if all markings reachable in (Σ, M_e) satisfy $\Sigma^\circ \subseteq M \Rightarrow \Sigma^\circ = M$.

3.2.1 Algebra of s-boxes

Now we are going to adapt the denotational semantics that we introduced in [12] 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 compositionally, 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 in the case of synchronization we can obtain some new transitions. They will be enumerated 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:

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

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 $cr(t)$:

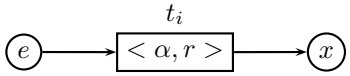
$$cr(t) = \sum_{\eta(t_j) \in \kappa(t)} \mu(t_j)$$

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]

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

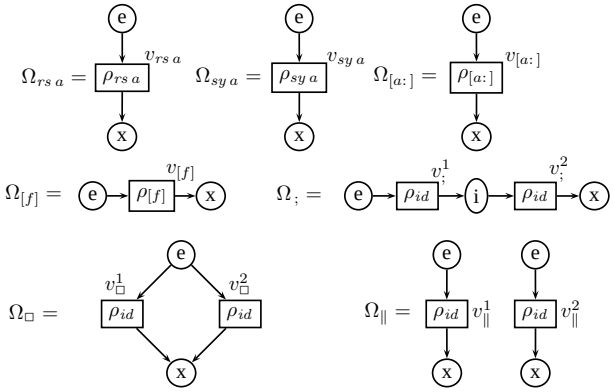


Figure 1. The operator s-boxes for finite sPBC

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 functions for T_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:

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

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

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

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

- $Box_s(E_1[f]) = \Omega_{[f]}(Box_s(E_1))$. 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:

$$\eta(t) = \eta_1(t) \qquad t \in T_1$$
$$\kappa(t) = \kappa_1(t) \qquad t \in T_1$$

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

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

$$\kappa(t) =$$

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

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.

- $Box_s(E_1\, rs\, a) = \Omega_{rs\, a}(Box_s(E_1))$. 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.

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

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

$$(\Gamma, \alpha + \{a\}) \in \rho_{sy\, a} \quad \wedge \quad (\Delta, \beta + \{\hat{a}\}) \in \rho_{sy\, a}$$
$$\text{then} \quad (\Gamma + \Delta, \alpha + \beta) \in \rho_{sy\, a}$$

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,

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

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 obtain by synchronizing (in different order) the same set of transitions. 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 $E = (<\{a\}, r_1> \parallel <\{\hat{a}, \hat{a}\}, r_2> \parallel (<\{a\}, r_3> \,\Box\, <\{a\}, r_4>))\, sy\, a$ be a regular static s-expression. Its corresponding plain s-box is depicted in Fig. 2, with:

$$\begin{array}{ll} \eta(t_1) = 1, & \kappa(t_1) = \{1\}, \\ \eta(t_2) = 2, & \kappa(t_2) = \{2\}, \\ \eta(t_3) = 3, & \kappa(t_3) = \{3, 4\}, \\ \eta(t_4) = 4, & \kappa(t_4) = \{3, 4\}, \\ \eta(t_{1.2}) = 1.2, & \kappa(t_{1.2}) = \{1.2\}, \\ \eta(t_{2.3}) = 2.3, & \kappa(t_{2.3}) = \{2.3, 2.4\}, \\ \eta(t_{2.4}) = 2.4, & \kappa(t_{2.4}) = \{2.3, 2.4\}, \\ \eta(t_{1.2.3}) = 1.2.3, & \kappa(t_{1.2.3}) = \{1.2.3, 1.2.4\}, \\ \eta(t_{1.2.4}) = 1.2.4, & \kappa(t_{1.2.4}) = \{1.2.3, 1.2.4\}, \end{array}$$

$$\begin{array}{l} R_{12} = min(r_1, r_2) \\ R_{23} = \frac{r_3}{r_3 + r_4} \cdot min(r_2, r_3 + r_4) \\ R_{24} = \frac{r_4}{r_3 + r_4} \cdot min(r_2, r_3 + r_4) \\ R_{123} = \frac{r_3}{r_3 + r_4} \cdot min(r_1, r_2, r_3 + r_4) \\ R_{124} = \frac{r_4}{r_3 + r_4} \cdot min(r_1, r_2, r_3 + r_4) \end{array}$$

□

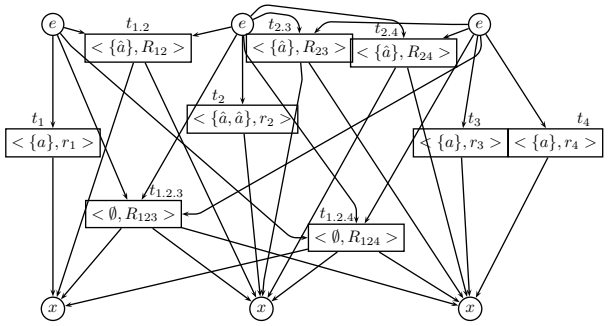


Figure 2. plain s-box corresponding to E

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. □

4. Conclusions and Future Work

sPBC is a stochastic extension of PBC, which was presented in [12]. In the first part of this paper we have summarized the main results of that work. We may observe that an important difference with respect to PBC is that in sPBC a total order semantics has been considered, although we still keep parallelism at the level of multiactions.

In this paper we have introduced a new proposal for the synchronization in sPBC, with the goal to obtain a stochastic equivalence relation. We have defined both an operational and a denotational semantics for this new version of sPBC, and we have seen that they are equivalent. The stochastic equivalence relation presented in this paper allows us to identify some processes with the same behaviour, such as:

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

However, this relation is not yet a congruence, because we need to capture some additional information, such as the termination and the distinction between parallelism and choices. Then, our current work is focused on the definition of an adequate stochastic equivalence relation on sPBC, with the intention of keeping (as far as possible) the classical laws of PBC. A preliminary version of this work can be found in [11].

We also intend to extend our results to the infinite case, by including both iteration and recursion. Finally, we will introduce some additional capabilities, such as urgent multiactions.

References

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

[2] J. 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, 1992.

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

[10] 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.

[11] H. Macià, V. Valero, and F. Cuartero. *A Congruence relation in finite sPBC*. Technical Report, DIAB-02-01-31, Department of Computer Science, University of Castilla-La Mancha, October 2002.

[12] 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.

[13] O. Marroquín and D. d. Frutos. *Extending the Petri Box calculus with Time*. Proc. Int. Conf. on Theory and Application of Petri Nets 2001. Springer, LNCS. Vol. 2075, 2001.

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

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