Defining Equivalence Relations in sPBC *

Hermenegilda Macià, Valentín Valero, and Fernando Cuartero

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

Abstract. Petri Box Calculus (PBC) is an algebraic model for the description of concurrent systems and sPBC (stochastic Petri Box Calculus) is a Markovian extension of that model. In this paper we define several equivalence relations for regular terms of sPBC, with the final goal to obtain a congruence.

Keywords: Process Algebras, Concurrency, Petri Box Calculus, Stochastic Models.

1 Introduction

PBC (Petri Box Calculus) [3–6] is a model for the description of concurrent systems, its operators have been 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*. In particular, synchronizations are separated from the parallel operator, by including a new operator (sy), in contrast to the way in which the synchronization is handled in classical process algebras, such as CCS [15], where the synchronization is embedded in the parallel operator. We know two timed extensions of PBC, namely tPBC [8] and TPBC [14], which consider a deterministic model of time. Our proposal, sPBC (stochastic Petri Box Calculus) is a Markovian extension of PBC.

In sPBC each multiaction (multiset of actions) has associated a random delay, which follows a negative exponential distribution. Thus, a (basic) stochastic multiaction is represented by a pair $<\alpha, r>$, where α represents a (classical) multiaction of PBC and $r \in \mathbb{R}^+$ is the parameter of the associated exponential distribution. As for stochastic Petri nets [1], in case of conflict we adopt the *race policy*, i.e., whenever two or more stochastic multiactions are possible, the fastest one is executed. The denotational semantics of sPBC is defined using as semantic values a special class of stochastic Petri nets, which are called s-boxes. As a consequence of the introduction of a continuous distribution (negative exponential) to measure delays of multiactions, the probability for two or more

* This work has been supported by the CICYT project "Description and performance of distributed systems and application to multimedia systems, Ref. TIC2003-07848-C02-02"

stochastic multiactions to be executed at the same time is zero. Then our operational semantics satisfies the *Total Order Assumption (TOA)* [2]: *all execution sequences of observable multiactions are totally ordered by precedence*, in contrast to the partial order semantics (*true concurrency*) of PBC.

The goal of this paper is to define a stochastic equivalence relation in sPBC that will be proved to be a congruence. A previous work in sPBC was presented in [13], where we introduced a first version of finite sPBC, not including recursion and iteration operators. There we saw that for the new multiactions generated by the application of the synchronization operator, we needed to select a parameter governing the corresponding exponential distribution. Several criteria could be applied to do that selection. We concluded that intuitively a good option could be to take as the value of the parameter the least of the values involved in the synchronization. But using that definition there are some important problems in order to define an adequate stochastic equivalence relation, which has motivated a new proposal for the semantics of the synchronization operator (see [12] for more details). This is not the only change required to obtain a congruence relation; we also need to introduce some new rules in the operational semantics of sPBC, as well as a special class of transitions and some additional information labelling the stochastic transitions. With the new rules we will be able to distinguish between processes that reach their final state and those that are not able to do that. The new special transitions allow us to capture those pairs of stochastic multiactions that could be executed in parallel. They will be called *ghosts transitions*, since they will not be used to evolve in the labelled transition system in the usual way; they are only included in order to capture the possible execution in parallel of two stochastic multiactions.

The rest of the paper is structured as follows: in Section 2 we present an overview of sPBC (for a fully description, the reader is referred to [13, 12, 11]). The equivalence relations on sPBC and the congruence are presented in Section 3. Finally, Section 4 contains some conclusions and some hints for our future work.

2 Stochastic Petri Box Calculus

As in plain PBC, static s-expressions are used to describe the structure of a concurrent system, while dynamic s-expressions describe the current state of a system (they correspond to unmarked and marked Petri nets, respectively). Static s-expressions of sPBC are those defined by the following BNF expression:

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

where $<\alpha, r>$ stands for the *basic multiaction*, which corresponds to the simultaneous execution of all the actions in α, after a delay that follows a negative exponential distribution with parameter r. $E_1 ; E_2$ stands for the sequential execution of E_1 and E_2, while $E_1 \,\square\, E_2$ is the choice between its arguments, $E[f]$ is the relabelling operator, and $E\ rs\ a$ denotes the restriction over the single action a, which generates a process that cannot execute any stochastic multiactions $<\alpha, r>$ with the multiaction α containing either the action a or $\hat{a}$. The parallel

operator, $\|$, represents the (independent) parallel execution of both components, where as in PBC there is no any synchronization embedded in the operator. Synchronization is introduced by the operator *sy* and it is based on CCS [15], thus the process $E\,sy\,a$ behaves in the same way as E, but it can also execute those new multiactions generated by the synchronization of a pair of actions $(a, \hat{a})$. $[a : E]$ is the derived operator *scoping* defined by $[a : E] = (E\,sy\,a)\,rs\,a$. Finally, the iteration operator $[E_1 * E_2 * E_3]$ represents the process that performs E_1, then executes several (possibly 0) times E_2, and finishes after performing E_3. In this paper we do not consider the recursion operator because it requires a more sophisticated treatment, as it occurred in plain PBC. However, we do consider the iteration operator, and some infinite behaviours can be described if we adequately combine it with the restriction operator.

We will denote static s-expressions by letters: E, F, $E_i, \ldots$, and the set of static s-expressions by *StatExpr*. However, we need to restrict the syntax of sPBC to those terms for which no parallel behaviour appears at the highest level in a choice or in the two last arguments of an iteration. Terms fulfilling this restriction will be called *regular terms*, and the operational semantics will be only defined for them. This restriction is introduced in order to guarantee that the moment in which the rule for the synchronization is applied does not affect the value that we obtain for the rate of the stochastic multiaction obtained as result of a synchronization (see Examples in [12, 11]).

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

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

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

The operational semantics of sPBC is defined in a very similar way to that of PBC [3, 4, 6], it can be found in [11]. Due to the lack of space we cannot include the rules, but we present a brief description of the two kind of rules it has. Firstly, the inaction rules [11] are introduced in order to establish the active components of a regular dynamic s-expression and by means of them we capture the equivalence of regular dynamic s-expressions. We say that a regular dynamic s-expression G is *operative* if it is not possible to apply any inaction rule to it. We will denote the set of all the operative regular dynamic s-expressions by *OpReDynExpr*. Considering the inaction rules and the reverse transitions, we define the structural equivalence relation for regular dynamic s-expressions as follows:

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

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

The other kind of rule is the stochastic transition, which has the following form:

$$G \xrightarrow{<a,r>} G'$$

where G is an operative regular dynamic s-expression. Again, you may find in [11] the set of rules that define the stochastic transitions. For that purpose two functions must be defined, *BC* and *Conflict*, whose technical definitions can also be found in [11]. The function *BC* gives us all the possible sets of bags of stochastic multiactions that can be executed concurrently by an operative regular dynamic s-expression:

$$BC : OpReDynExpr \to \mathcal{P}(\mathcal{B}(\mathcal{SL}))$$

On the other hand, the function *Conflict* allow us to obtain those stochastic multiactions $<\alpha, r'>$ which are in conflict (or competition) with a stochastic multiaction $<\alpha, r>$ executable from an operative regular dynamic s-expression G:

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

Notice that we only need to consider those stochastic multiactions in conflict executing the same multiaction α (but possibly with a different rate).

An important consequence of the rules defining the stochastic transitions is the following. Taking:

$$E =< \alpha, r_1 > \Box < \alpha, r_2 > \Box \ldots \Box < \alpha, r_n > \quad \text{and} \quad F =<, \alpha, \textstyle\sum_{i=1}^{n} r_i >$$

we obtain for $\overline{E}$, $\overline{F}$ the same delay for executing the multiaction α: $r = \sum_{i=1}^{n} r_i$. Thus, the CTMCs obtained from the transition systems of $\overline{E}$ and $\overline{F}$ are the same, and we can consider that both s-expressions are stochastically equivalent.

Definition 1. We define the labelled (multi)transition system of any regular dynamic s-expression G, $ts(G)$, by $ts(G) = (V, A, v_0)$, where:

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

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

where $[G\rangle = \{G\} \cup \{H' \in ReDynExpr \,|\, \exists < \alpha_1, r_1 >, \ldots, < \alpha_n, r_n > \text{ with } G \equiv G' \xrightarrow{<\alpha_1, r_1>} G_1 \equiv G'_1 \xrightarrow{<\alpha_2, r_2>} \ldots\ G_{n-1} \equiv G'_{n-1} \xrightarrow{<\alpha_n, r_n>} H \equiv H'\}$

In order to compute the number of different instances of each transition $([H]_{\equiv}, < \alpha, r >, [J]_{\equiv})$ in A, we consider equivalent all the different ways to derive the same stochastic transition (see [12, 10]). Then, when we apply the synchronization rule in order to obtain a new stochastic transition, the generated stochastic multiaction can be annotated with the concatenation of the numbering of the stochastic multiactions involved in the synchronization[1], so that when we detect that a permutation of the numbering has been already obtained by a previous application of the rule, then that new stochastic transition will not be considered. □

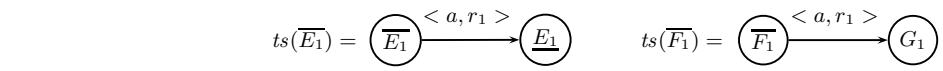
Notice that the *race policy* will govern the dynamic behaviour of the system when two or more stochastic multiactions are simultaneously enabled. Then, due to the fact that we use exponential distributions, the stochastic process associated with the evolution of any regular dynamic s-expression $\overline{E}$, which can be easily obtained from $ts(\overline{E})$, (see [13]), is a Continuous Time Markov Chain (CTMC).

[1] We can enumerate the stochastic multiactions from left to right, in the same order as they appear in the syntax of the s-expression.

3 Equivalence Relations

In this section we follow similar steps to those presented in [4] for PBC in order to find an appropriate definition of congruence. Due to the peculiarities of sPBC with respect to others stochastic process algebras (the evolution of a dynamic s-expression is indicated by means of overbars and underbars, the synchronization is separated of the parallel operator, the presence of multiactions and the multiway synchronization, and also a special approach to obtain the rate for the synchronization), it is necessary to consider a different mechanism in order to obtain a congruence relation to that presented in other stochastic process algebras [7]. Firstly, we observe that the simple notion of isomorphism between transitions systems is not a congruence in sPBC, as the following example shows:

Example 1. Considering $E_1 =< \{a\}, r_1 >,\ F_1 =< \{a\}, r_1 >; < \{b\}, r_2 > rs\ b$ then we may construct $ts(\overline{E_1})$ and $ts(\overline{F_1})$ as follows:



Both labelled transition systems are isomorphic, but if we consider:

$$E_2 = E_1; < \{c\}, r_3 >, \quad F_2 = F_1; < \{c\}, r_3 >$$

we obtain that $ts(\overline{E_2})$ is not isomorphic to $ts(\overline{F_2})$. □

To solve this problem we need to distinguish between dynamic s-expressions that are in its final state (as $\underline{E_1}$), and those that have not yet reached its final state (as G_1 in the figure). Then, we capture the termination by extending the labelled transition system with two new transitions (Table 1). We take 0 as the rate for *Skp*, which intuitively means that it will be never executed, and we take ∞ as the rate for *Rdo*, which means that it would be immediately executed whenever it becomes enabled.

Table 1. Rules Skp and Rdo

(Skp)	$\overline{E} \xrightarrow{<skip,0>} \underline{E}$
(Rdo)	$\underline{E} \xrightarrow{<redo,\infty>} \overline{E}$

We will denote by $ts_2(\overline{E})$ the labelled transition system that we obtain by adding these two new rules. For instance, for E_1, F_1 defined in Example 1, we obtain that now $ts_2(\overline{E_1})$ and $ts_2(\overline{F_1})$ are not isomorphic. However, this is not yet enough to get a congruence in sPBC, since we cannot distinguish between parallelism and choice, as the following example shows.

Example 2. Let us consider:
$E_3 =< \{a\}, r > \| < \{\hat{a}\}, r >, \quad F_3 =< \{a\}, r >; < \{\hat{a}\}, r > \Box < \{\hat{a}\}, r >; < \{a\}, r >$
Then, $ts_2(\overline{E_3})$ and $ts_2(\overline{F_3})$ are isomorphic, but taking $E_4 = E_3\ sy\ a$ and $F_4 = F_3\ sy\ a$, we have that $ts_2(\overline{E_4})$ and $ts_2(\overline{F_4})$ are not isomorphic anymore, because we have a new transition in $ts_2(\overline{E_4})$ obtained by synchronizing, and we do not have any new transition in $ts_2(\overline{F_4})$. □

Therefore, we need some additional information in the labelled transition systems, by means of which we will be able to distinguish those transitions that have been obtained from a parallel operator, and therefore are later usable for a new synchronization. This is why we include a new kind of transitions, which we call *ghosts* transitions. They do not represent a possible evolution of the system, but instead they represent potential executions of pairs of stochastic multiactions in parallel, which would only become real if the corresponding synchronization operator is later applied. Since we need to capture the possible execution of two stochastic multiactions in parallel, we simply distinguish these transitions by annotating them with the corresponding stochastic multiactions over a dashed arrow. The rule that introduces this new kind of transitions is that presented in Table 2. Note that for each concurrent bag $\{< \alpha_1, r_1 >, < \alpha_2, r_2 >\} \in BC(G)$ we would get two different transitions

$$G \overset{<\alpha_1,r_1> \| <\alpha_2\, r_2>}{\dashrightarrow} G_2 \quad \text{and} \quad G \overset{<\alpha_2,r_2> \| <\alpha_1\, r_1>}{\dashrightarrow} G_2$$

But again, we will only include in the labelled transition system one of them, as we did for the synchronizations.

Table 2. Ghost transition rule

(ghost) Let $\{< \alpha_1, r_1 >, < \alpha_2, r_2 >\} \in BC(G)$
$$\frac{G \overset{<\alpha_1,r_1>}{\longrightarrow} G_1 \, (\overset{\emptyset}{\longrightarrow})^* \, G_1^* \overset{<\alpha_2,r_2>}{\longrightarrow} G_2}{G \overset{<\alpha_1,r_1> \| <\alpha_2\, r_2>}{\dashrightarrow} G_2}$$

We now define a new transition system of $\overline{E}$, for each regular static s-expression E, taking into account the ghost and *Skp* and *Rdo* transitions.

Definition 2. Let E be a regular static s-expression. We define $ts_{\mathcal{G}}(\overline{E})$ by: $ts_{\mathcal{G}}(\overline{E}) = (V \cup \{[\underline{E}]_{\equiv}\}, A \cup A_{sr} \cup A_g, v_0)$, *where* $ts(\overline{E}) = (V, A, v_0)$, *and* :

- $A_{sr} = \{([\overline{E}]_{\equiv}, < skip, 0 >, [\underline{E}]_{\equiv}) \,,\, ([\underline{E}]_{\equiv}, < redo, \infty >, [\overline{E}]_{\equiv})\}$
- $A_g = \{\!\!\{ \; ([H]_{\equiv}, < \alpha, r > \| < \beta, s >, [J]_{\equiv}) \,|\, H \in [\overline{E}\rangle \,,\, H \overset{<\alpha,r> \| <\beta,s>}{\dashrightarrow} J \; \}\!\!\}$ □

Then, we may now introduce another notion of isomorphism ($\cong$) between regular static s-expressions.

Definition 3. For any pair of regular static s-expressions E_1 and E_2, with:
$ts_3(\overline{E_1}) = (V^{E_1} \cup \{[\underline{E_1}]_{\equiv}\}, A^{E_1} \cup A^{E_1}_{sr} \cup A^{E_1}_g, v_0^{E_1})$, with $ts(\overline{E_1}) = (V^{E_1}, A^{E_1}, v_0^{E_1})$
$ts_3(\overline{E_2}) = (V^{E_2} \cup \{[\underline{E_2}]_{\equiv}\}, A^{E_2} \cup A^{E_2}_{sr} \cup A^{E_2}_g, v_0^{E_2})$, with $ts(\overline{E_2}) = (V^{E_2}, A^{E_2}, v_0^{E_2})$
we write $E_1 \cong E_2$, if there is a bijective function:

$$h \,:\, V^{E_1} \cup \{[\underline{E_1}]_{\equiv}\} \to V^{E_2} \cup \{[\underline{E_2}]_{\equiv}\}$$

such that

- $h(v_0^{E_1}) = v_0^{E_2}$, $h([\underline{E_1}]_{\equiv}) = [\underline{E_2}]_{\equiv}$
- $\forall v^{E_1}, w^{E_1} \in V^{E_1}$:
  $A^{E_1}(v^{E_1}, < \alpha, r >, w^{E_1}) = A^{E_2}(h(v^{E_1}), < \alpha, r >, h(w^{E_1}))$,
  $A^{E_1}_g(v^{E_1}, < \alpha, r > \| < \beta, s >, w^{E_1}) = A^{E_2}_g(h(v^{E_1}), < \alpha, r > \| < \beta, s >), h(w^{E_1}))$

□

So, if we consider again the processes in Example 2, then we can check that $ts_3(\overline{E_3})$ and $ts_3(\overline{F_3})$ are not isomorphic, because we have a ghost transition in $ts_3(\overline{E_3})$ which does not appear in $ts_3(\overline{F_3})$. Consequently, $E_3 \not\cong F_3$.

The equivalence defined by $\cong$ is basically the same that we had for plain PBC. This relation is in fact a congruence in plain PBC, but not in sPBC, as the following example demonstrates.

Example 3. Let us consider the following static s-expressions:

$E_5 = [\, f : ((<\{a,f\},1> \,\square\, <\{\hat{f}\},3>) \,\|\, (<\{a,f\},2> \,\square\, <\{\hat{f}\},3>))\,],$
$F_5 = <\{a\},1> \,\square\, <\{a\},2>$

Their corresponding transition systems are isomorphic. However, if we now consider: $E_6 = (E_5 \,\|\, <\{\hat{a}\},1>) sy\, a\,, \quad F_6 = (\, F_5 \,\|\, <\{\hat{a}\},1>) sy\, a$

It is easy to prove that $E_6 \not\cong F_6$, because in $ts_3(\overline{E_6})$ we have in particular two edges connecting $\overline{E_6}$ with $\underline{E_6}$, both with label $<\emptyset, 1>$; while in $ts_3(\overline{F_6})$ we also have two edges connecting $\overline{F_6}$ with $\underline{F_6}$, one of them labelled by $<\emptyset, 1/3>$, and the other one by $<\emptyset, 2/3>$. This is because $<\{a\},1>$ and $<\{a\},2>$ are in the same set of conflicts in $ts_3(\overline{F_6})$, but in $ts_3(\overline{E_5})$ they are in different conflict sets. □

From the previous example we conclude that we need to expand the transition system in order to capture the conflict information at any state. In order to introduce our new definition of the transition system we need the following lemma (its proof can be found in [10]). From now onwards, we consider each stochastic multiaction with its corresponding numbering: then, as said before, we enumerate the stochastic multiactions from left to right in the syntax of the s-expression and, if the stochastic multiaction is obtained by synchronization, then we take the concatenation of the numbering of the stochastic multiactions involved in the synchronization.

Lemma 1. Let $G \in$ *OpReDynExpr* and $<\alpha, r>_n$ executable from G. Then, $\forall G' \in$ *OpReDynExpr*, $G' \equiv G$, such that $<\alpha, r>_n$ is also executable from G', we have $\mathit{Conflict}(G', <\alpha, r>_n) = \mathit{Conflict}(G, <\alpha, r>_n)$. □

This lemma allows us to define for all $G \in$ *ReDynExpr*:

$$\mathit{Conflict}([G]_{\equiv}, <\alpha, r>_n) = \mathit{Conflict}(G', <\alpha, r>_n)$$

taking any operative $G' \in [G]_{\equiv}$, such that $<\alpha, r>_n$ is executable from G'.

The *new transition system* is now defined, which is structurally identical to ts_3, but each stochastic transition is now labelled with the corresponding stochastic multiaction, its numbering and the conflict set it belongs to. Ghost transitions are also the same that we had in ts_3, but the involved stochastic multiactions are annotated with their numbering. It is important to observe that these little changes do not modify the interpretation that we make of transition systems, i.e., we have exactly the same evolutions that we had in ts_3, we are only including some additional information that will be needed when we define the stochastic equivalence.

Definition 4. Let E be a regular static s-expression. We define the new transition system of $\overline{E}$ by: $nts(\overline{E}) = (V \cup \{[\underline{E}]_{\equiv}\}, A_a \cup A_{sr} \cup A_g, v_0)$, where

- $V = \{\, [H]_{\equiv} \mid H \in [\overline{E}\rangle \,\}$, $v_0 = [\overline{E}]_{\equiv}$,
- $A_a = \{([H]_{\equiv}, <\alpha, r>_{n,C}, [J]_{\equiv}) \mid H \in [\overline{E}\rangle,\ H \stackrel{<\alpha,r>_n}{\longrightarrow} J,$ with $C = Conflict([H]_{\equiv}, <\alpha, r>_n)\}$
- $A_{sr} = \{\, ([\overline{E}]_{\equiv}, <skip, 0>, [\underline{E}]_{\equiv})\,,\ ([\underline{E}]_{\equiv}, <redo, \infty>, [\overline{E}]_{\equiv})\,\}$
- $A_g = \{\, ([H]_{\equiv}, <\alpha, r>_{n_1} \parallel <\beta, s>_{n_2}, [J]_{\equiv}) \mid H \in [\overline{E}\rangle, H \stackrel{<\alpha,r>_{n_1} \parallel <\beta,s>_{n_2}}{\dashrightarrow} J \,\}$

□

Notice that in the particular cases of *skip* and *redo* we do not need to introduce neither the numbering nor the set of conflicts, so that we have maintained these two edges as in ts_3.

Definition 5. Let E be a regular static s-expression, with $nts(\overline{E}) = (V \cup \{[\underline{E}]_{\equiv}\}, A_a \cup A_{sr} \cup A_g, v_0)$ and $(v, <\alpha, r>_{n,C}, w) \in A_a$. Then, we define the conflict rate of C by $rc(C) = \sum\{\, r_j \mid <\alpha, r_j>_n \in C\,\}$, and we will denote the set that consists of the sets of conflicts at the state v with respect to the multiaction α by $\mathcal{C}(v, \alpha) = \{\, C_j \mid (v, <\alpha, r_i>_{n',C_j}, w_k) \in A_a\}$. We also define the transition rate from v to w with respect to the multiaction α and the set of conflicts C, by $r(v, \alpha, C, w) = \sum\{\, r_j \mid (v, <\alpha, r_j>_{n',C}, w) \in A_a\,\}$

□

The definition of the equivalence relation $\sim$ follows:

Definition 6. For any pair of regular static s-expressions E_1, E_2, with:
$nts(\overline{E_1}) = (V^{E_1} \cup \{[\underline{E_1}]_{\equiv}\}, A_a^{E_1} \cup A_{sr}^{E_1} \cup A_g^{E_1}, v_0^{E_1})$,
$nts(\overline{E_2}) = (V^{E_2} \cup \{[\underline{E_2}]_{\equiv}\}, A_a^{E_2} \cup A_{sr}^{E_2} \cup A_g^{E_2}, v_0^{E_2})$
We will say that E_1 and E_2 are stochastically equivalent, and we will denote it by $E_1 \sim E_2$, if there exists a bijective function:

$$\phi\ :\ V^{E_1} \cup \{[\underline{E_1}]_{\equiv}\} \to V^{E_2} \cup \{[\underline{E_2}]_{\equiv}\}$$

and for all pair (v, α) such that $\mathcal{C}(v, \alpha) \neq \emptyset$, there exists a bijective function:
$h_{v,\alpha} : \mathcal{C}(v, \alpha) \to \mathcal{C}(\phi(v), \alpha)$, such that $rc(C_j) = rc(h_{v,\alpha}(C_j))$, $\forall C_j \in \mathcal{C}(v, \alpha)$ fulfilling the following conditions:

- $\phi([\overline{E_1}]_{\equiv}) = [\overline{E_2}]_{\equiv}$ and $\phi([\underline{E_1}]_{\equiv}) = [\underline{E_2}]_{\equiv}$
- Taking $\phi(v^1) = v^2, \phi(w^1) = w^2, h_{v^1,\alpha}(C_1) = C_1'$ and $h_{v^1,\beta}(C_2) = C_2'$, then:
  - If $(v^1, <\alpha, r_1>_{n_1,C_1}, w^1) \in A_a^{E_1}$, then $\exists s_1, n_1'$ such that $(v^2, <\alpha, s_1>_{n_1',C_1'}, w^2) \in A_a^{E_2}$, with $r(v^1, \alpha, C_1, w^1) = r(v^2, \alpha, C_1', w^2)$.
  - If $(v^2, <\alpha, s_1>_{n_1',C_1'}, w^2) \in A_a^{E_2}$, then $\exists r_1, n_1$ such that $(v^1, <\alpha, r_1>_{n_1,C_1}, w^1) \in A_a^{E_1}$, with $r(v^2, \alpha, C_1', w^2) = r(v^1, \alpha, C_1, w^1)$.
  - If $(v^1, <\alpha, r_1>_{n_1} \parallel <\beta, r_2>_{n_2}, w^1) \in A_g^{E_1}$, then
    - $\exists (v^1, <\alpha, r_1>_{n_1,C_1}, x^1) \in A_a^{E_1},\ \exists (v^1, <\beta, r_2>_{n_2,C_2}, y^1) \in A_a^{E_1}$,
    - $\exists\, s_1, s_2, n_1', n_2'$ such that $(v^2, <\alpha, s_1>_{n_1'} \parallel <\beta, s_2>_{n_2'}, w^2) \in A_g^{E_2}$, with $(v^2, <\alpha, s_1>_{n_1',C_1'}, \phi(x^1)) \in A_a^{E_2} \ \wedge\ (v^2, <\beta, s_2>_{n_2',C_2'}, \phi(y^1)) \in A_a^{E_2}$.
  - If $(v^2, <\alpha, s_1>_{n_1'} \parallel <\beta, s_2>_{n_2'}, w^2) \in A_g^{E_2}$, then:
    - $\exists (v^2, <\alpha, s_1>_{n_1',C_1'}, x^2) \in A_a^{E_2},\ \exists (v^2, <\beta, s_2>_{n_2',C_2'}, y^2) \in A_a^{E_2}$,
    - $\exists\, r_1, r_2, n_1, n_2$ such that $(v^1, <\alpha, r_1>_{n_1} \parallel <\beta, r_2>_{n_2}, w^1) \in A_g^{E_1}$, with $(v^1, <\alpha, r_1>_{n_1,C_1}, \phi^{-1}(x^2)) \in A^{E_1} \wedge (v^1, <\beta, r_2>_{n_2,C_2}, \phi^{-1}(y^2)) \in A_a^{E_1}$.

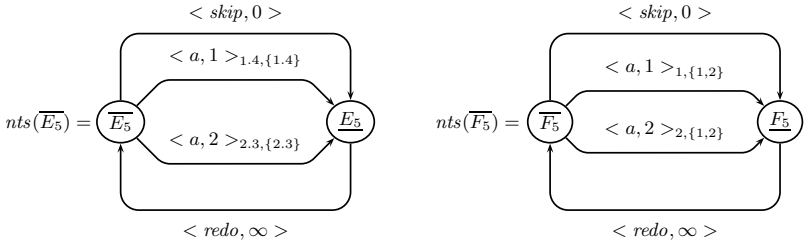
□

Corollary 1. The pairs of regular static s-expressions (E, F) that verify $E \sim F$ have isomorphic CTMCs. □

Example 4. Let us consider again the static s-expressions of Example 3:

$E_5 = [\, f : ((<\{a, f\}, 1>_1 \;\Box\; <\{\hat{f}\}, 3>_2) \,\|\, (<\{a, f\}, 2>_3 \;\Box\; <\{\hat{f}\}, 3>_4))]$,
$F_5 = <\{a\}, 1>_1 \;\Box\; <\{a\}, 2>_2$

Their corresponding transition systems ($nts(\overline{E_5})$ and $nts(\overline{F_5})$) are the following:



It follows that
$\mathcal{C}([\overline{E_5}]_{\equiv}, a) = \{\{<\{a\}, 1>_{1.4}\}, \{<\{a\}, 2>_{2.3}\}\}$,
$\mathcal{C}([\overline{F_5}]_{\equiv}, a) = \{\{<\{a\}, 1>_1, <\{a\}, 2>_2\}\}$

Therefore, there cannot be a bijection between $\mathcal{C}([\overline{E_5}]_{\equiv}, a)$ and $\mathcal{C}([\overline{F_5}]_{\equiv}, a)$, and consequently, $E_5 \not\sim F_5$. □

Example 5. The following static s-expressions are stochastically equivalent:

$$E_7 = <\{a\}, 2> \;\sim\; F_7 = <\{a\}, 1> \Box <\{a\}, 1>,$$
$$E_8 = <\{\hat{a}\}, 3> \;\sim\; F_8 = <\{\hat{a}\}, 1> \Box <\{\hat{a}\}, 2>$$

Furthermore, $E_7 \| E_8 \sim F_7 \| F_8$, and taking $E_9 = (E_7 \| E_8)\ sy\ a$, $F_9 = (F_7 \| F_8)\ sy\ a$, we also have $E_9 \sim F_9$. □

The following theorem states that $\sim$ is indeed a congruence relation. Its proof is large and a bit tedious, it can be found in [10].

Theorem 1. Let E_1, E_2, E, E' be regular static s-expressions such that $E_1 \sim E_2$. Then:
$E_1 ; E \sim E_2 ; E$, $E ; E_1 \sim E ; E_2$, $E_1 \| E \sim E_2 \| E$, $E \| E_1 \sim E \| E_2$,
$E_1 \Box E \sim E_2 \Box E$, $E \Box E_1 \sim E \Box E_2$, $E_1\ sy\ a \sim E_2\ sy\ a$, $E_1\ rs\ a \sim E_2\ rs\ a$,
$E_1[f] \sim E_2[f]$ (for a bijective function f), $[\, E_1 * E' * E\,] \sim [\, E_2 * E' * E\,]$,
$[\, E' * E_1 * E\,] \sim [\, E' * E_2 * E\,]$, $[\, E' * E * E_1\,] \sim [\, E' * E * E_2\,]$. □

4 Conclusions and Future Work

sPBC is a stochastic (Markovian) extension of PBC, presented in [13]. In this paper we have considered a new definition of the semantics of the synchronization operator [12], which is based on the so called *conflict rates*, with the goal to define an adequate stochastic equivalence relation, which is a congruence. We have considered a new class of labelled transition systems, in which we have introduced some new transitions: *Skp, Rdo* and the *ghosts transitions*; and also

some additional information capturing the *conflict* information. With the rules *Skp, Rdo* we can distinguish between s-expressions that reach their final state and that are not able to do that. On the other hand, with the *ghosts transitions* we can difference a parallel of a choice. All of them, as well as the *conflict* information will not be used to evolve a process in the usual way, they are only included in order to obtain an adequate equivalent relation $\sim$, that it is indeed a congruence.

Our current work on the subject is focussed on the definition of a stochastic bisimulation based on the probabilistic bisimulation initially introduced in [9], and to prove that it is indeed a congruence too. Furthermore, we are trying to extend our results considering the complete set of operators (i.e., including the recursion operator). Finally, our intention is also to introduce in the model some additional capabilities, such as immediate multiactions.

References

1. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Wiley, 1995.
2. J.C.M. Baeten. The Total Order Assumption. In *Proc. Workshop What good is partial order,* Sheffiel (E. Best, ed.), Hildesheimer Informatik-Berichte 13/92, Universitat Hildesheim, pages 1-11, 1992.
3. E. Best, R. Devillers, and M. Koutny. A Consistent Model for Nets and Process Algebras. In the book *The Handbook on Process Algebras,* North Holland, Ch. 14, pp. 873-944, 2001.
4. E. Best, R. Devillers, and M. Koutny. *Petri Net Algebra.* EATC, Springer, 2001.
5. E. Best, R. Devilllers, and J. Hall. The Box Calculus: A New Causal Algebra with Multi-label Communication. In *Advances in Petri Nets, LNCS 609, Springer, pp. 21-69*, 1992.
6. E. Best and M. Koutny. A Refined View of the Box Algebra. In *Proc. Application and Theory of Petri Nets.* LNCS 935, Springer, pp. 1-20, 1995.
7. H. Hermanns, U. Herzog, and J.P. Katoen. Process algebra for performance evaluation. In *Theorical Computer Science, Vol. 274 (1-2),* pages 43-78, 2002.
8. M. Koutny. A Compositional Model of Time Petri Nets. *International Conference on Theory and Application of Petri Nets, LNCS 1825,* pp. 303-322, Springer, 2000.
9. K. Larsen and A. Skou. Bisimulation through Probabilistic Testing. *Information and Computation, 94(1):1-28*, 1991.
10. H. Macià. *sPBC: Una Extensión Markoviana del Petri Box Calculus.* PhD thesis, Dpto. de Informática, Universidad de Castilla-La Mancha (in spanish), 2003.
11. H. Macià, V. Valero, D. Cazorla, and F. Cuartero. Introducing the iteration in sPBC. In *Proc. of FORTE'04, LNCS 3235, Springer, pp. 292-308*, 2004.
12. H. Macià, V. Valero, F. Cuartero, and F.L. Pelayo. A new proposal for the synchronization in sPBC. In *Proc. ACSD'03,* pp. 216-225, IEEE C.S. Press, 2003.
13. H. Macià, V. Valero, and D. de Frutos. sPBC: A Markovian Extension of Finite Petri Box Calculus. In *Proc. PNPM'01,* pp. 207-216. IEEE C.S. Press, 2001.
14. O. Marroquín and D. de Frutos. Extending the Petri Box Calculus with Time. In *Proc. Int. Conf. of TAPN'01. LNCS 2075, Springer,* pp. 195-207, 2001.
15. R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.