Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100210

Modelling a Video Conference System with sPBC

Hermenegilda Macià[1,*], *Valentín Valero*[1], *Fernando Cuartero*[1], *M. Carmen Ruiz*[1] *and Igor V. Tarasyuk*[2]

[1] Escuela Superior de Ingeniería Informática, Universidad de Castilla-La Mancha, 02071 Albacete, Spain
[2] A.P. Ershov Institute of Informatics Systems, SB RAS, 6, Acad. Lavrentiev pr., 630090 Novosibirsk, Russian Federation

Received: 10 Aug. 2015, Revised: 28 Oct. 2015, Accepted: 29 Oct. 2015
Published online: 1 Mar. 2016

Abstract: Stochastic Petri Box Calculus (sPBC) with immediate multiactions is an algebraic model for the description of concurrent systems, whose activities have a random time associated (governed by an exponential distribution) or they are immediate (no time is required for their execution). One of the main features of sPBC, in contrast to other classical stochastic process algebras, is that it considers multiactions instead of single actions. Furthermore, a description in this version of sPBC has a natural and easy translation into Generalized Stochastic Petri Nets (GSPNs). In this paper we show how the calculus can be applied to information science phenomena, specifically, to model and analyze a Video Conference System. We will see that this particular kind of system can be easily described and analyzed within sPBC with immediate multiactions. This case study illustrates the power and flexibility of our stochastic process algebra in the area of control and systems engineering.

Keywords: Modelling, Concurrent Systems, Petri Box Calculus, Stochastic Process Algebra, Generalized Stochastic Petri Nets.

1 Introduction and Related Work

Algebraic process calculi are a well-known formal model for specification of computer and communication systems and for analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by process expressions, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In the last decades, stochastic extensions of PAs were proposed and widely used. Stochastic process algebras (SPAs) do not just specify actions that can occur (qualitative features), like ordinary PAs, but they associate with actions quantitative parameters (quantitative characteristics), such as rates or probabilities, related to the distributions of the random action delays or durations. Some well-known SPAs are Markovian TImed Processes for Performance evaluation (MTIPP) [10], Performance Evaluation Process Algebra (PEPA) [11] and Extended Markovian Process Algebra (EMPA) [3].

PAs specify concurrent systems in a compositional way via an expressive formal syntax. On the other hand, Petri nets (PNs) provide a graphical representation of such systems and capture explicit asynchrony in their behaviour. To combine the advantages of both models, a semantics of algebraic formulas in terms of PNs is defined.

Petri Box Calculus (PBC) [4,5,6,7] is a flexible and expressive PA, based on Calculus of Communicating Systems (CCS) [20] and intended for the description and analysis of concurrent systems. Its goal was to propose a compositional semantics for high-level constructs of concurrent programming languages in terms of elementary PNs. Formulas of PBC are combined not just from single (visible or invisible) actions and variables, like in CCS, but from multisets of elementary actions and their conjugates, called multiactions. The empty multiset of actions is interpreted as the silent multiaction specifying an invisible activity. The operators of PBC have been selected in order to obtain an easy and natural translation into Petri nets, thus combining the advantages of both the algebraic and net formal models. PBC has a step operational semantics in terms of labeled transition systems, constructed by the rules of the classical structural operational semantics (SOS). The denotational semantics of PBC was defined via a subclass of PNs, equipped with an interface and considered up to isomorphism, called Petri boxes (shortly, boxes).

To specify systems with time constraints, such as real-time systems, deterministic (fixed) or nondeterministic (interval) time delays are used. In this

* Corresponding author e-mail: hermenegilda.macia@uclm.es

way, PBC was enriched by adding time constraints that permit to represent time-dependent and time-critical systems within the calculus.

A time extension of PBC with a nondeterministic time model, called time Petri box calculus (tPBC), was proposed in [13]. In tPBC, timing information was added by associating time intervals (the earliest and the latest firing time) with instantaneous *actions* (i.e. not with multiactions). Its denotational semantics was defined in terms of a subclass of labeled time PNs (tPNs) [19], called time Petri boxes (ct-boxes). tPBC has a step time operational semantics in terms of labeled transition systems.

Another time enrichment of PBC, called Timed Petri box calculus (TPBC), was defined in [18], and it accommodates a deterministic model of time. In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. Additionally, in TPBC there exist no "illegal" multiaction occurrences, unlike tPBC. The complexity of "illegal" occurrences mechanism was one of the main intentions to construct TPBC, though this calculus appeared to be more complex than tPBC. The denotational semantics of TPBC was defined in terms of a subclass of labeled Timed Petri nets (TPNs) [22], called Timed Petri boxes (T-boxes). TPBC has a step timed operational semantics in terms of labeled transition systems.

The state space of the systems with deterministic or nondeterministic delays often differs drastically from that of the timeless systems, hence, the analysis results for untimed systems may be non-applicable to the time ones. Therefore, stochastic delays are considered, which are the random variables with (discrete or continuous) probability distributions. In particular, PBC was extended by including stochastic time, with the goal to describe a wider class of systems, such as fault-tolerance ones.

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [14,15,16]. In sPBC, delays of stochastic multiactions follow (negative) exponential distribution. Each multiaction is equipped with a rate that is a parameter of the corresponding exponential distribution. The instantaneous execution of a stochastic multiaction is possible only after the stochastic time delay assigned. The calculus has an interleaving operational semantics in terms of transition systems, labeled with multiactions and their rates. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs (CTSPNs) [1], called stochastic Petri boxes (s-boxes). In sPBC, performance is evaluated by analyzing the underlying stochastic process, which is a continuous time Markov chain (CTMC).

In [17], sPBC was enriched with immediate multiactions, which have a deterministic zero time delay. Immediate multiactions improve the specification capabilities: for instance, they can model instantaneous probabilistic choices and activities whose duration is insignificant compared to those of others. This allows us to get a simpler and clearer representation of the systems. sPBC with immediate multiactions has an interleaving operational semantics via transition systems, labeled with stochastic or immediate multiactions, together with their rates or probabilities, respectively. The denotational semantics of sPBC with immediate multiactions was defined via a subclass of labeled generalized stochastic PNs (GSPNs) [2], called generalized stochastic Petri boxes (gs-boxes). The performance analysis in sPBC with immediate multiactions is accomplished via the underlying semi-Markov chains (SMCs).

In [24], a discrete time stochastic extension dtsPBC of PBC was presented. In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (DTSPNs) [21], called discrete time stochastic Petri boxes (dts-boxes). The underlying stochastic process, which is a discrete time Markov chain (DTMC), was constructed and investigated to analyze performance in dtsPBC.

In [25], discrete time stochastic and immediate PBC (dtsiPBC) was introduced, as an extension of dtsPBC by adding immediate multiactions. Thus, dtsiPBC possess concurrent discrete time semantics with geometrically distributed (like in dtsPBC) or zero sojourn time in the states of algebraic processes. dtsiPBC has a step operational semantics, based on labeled probabilistic transition systems. The denotational semantics of the calculus is defined via a subclass of labeled discrete time stochastic and immediate PNs (DTSIPNs), called dtsi-boxes. To evaluate performance in dtsiPBC, the underlying stochastic process is studied, which is an SMC. In addition, the alternative solution methods were developed, based on the underlying discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states (those with zero residence time).

In this paper, we focus on sPBC with immediate multiactions [17] (we shall simply call it sPBC from now on) that is a (semi-)Markovian extension of PBC. If we compare sPBC with the classical SPAs MTIPP, PEPA and EMPA, the first main difference between them comes from PBC, since sPBC is based on this calculus: all algebraic operations and a notion of multiaction are inherited from PBC. The second main difference is immediate multiactions, since there are no instantaneous activities in MTIPP and PEPA, although immediate actions in EMPA partly resemble immediate multiactions. Thus, unlike the classical SPAs, in sPBC, we have multiactions, whose constituent elementary actions model low-level concurrent activities in a natural way. More elaborated stochastic multiactions are suitable for modelling durational computations and delayed work while immediate multiactions are useful to specify logical conditions, probabilistic branching and urgent events. Each stochastic multiaction is specified by the pair $<\alpha, r>$, where α denotes a (classical) multiaction of PBC

and $r \in \mathbb{R}^+$ is the parameter of the associated exponential distribution. Each immediate multiaction is specified by the pair $<\alpha, \infty_{l,p}>$, where α is again a (classical) multiaction of PBC and $l \in \mathbb{N}$ is the priority of this immediate multiaction while $p \in \mathbb{R}^+$ is its weight, interpreted in the same way as in GSPNs.

In addition, sPBC has the sequence operator, in contrast to the prefix operator in the classical SPAs. Relabeling in sPBC is analogous to that in EMPA, but it is additionally extended to conjugated actions. Restriction in sPBC differs from hiding in PEPA and functional abstraction in EMPA, where hidden actions are labeled with a symbol of "silent" action τ. For a given expression of sPBC, the restriction by an action means that any process behaviour containing the action or its conjugate is not allowed. sPBC has no recursion operation or recursive definitions, but it includes the iteration operator, which allows us to specify in a clear and easy way infinite repetitive and looping behaviours.

Furthermore, unlike the other classical SPAs, the synchronization operator in sPBC is separated from the parallel one, thereby system design is considerably simplified. The synchronization over an elementary action in sPBC collects all the pairs consisting of this elementary action and its conjugate, which are contained in the multiactions from the synchronized activities. This operation produces some new activities whose first element is the union of the multiactions, excepting all the pairs of conjugate actions that have been synchronized. The second element of the activity resulting from a synchronization between stochastic multiactions is the *conflict rate*, taking the slowest one, and it is inspired by the *apparent rate* of PEPA. On the other hand, we only allow two immediate multiactions to synchronize if they have the same level of priority, and in this case, the second element of the resulting activity is the joint weight, obtained as the product of the weights of the immediate multiactions involved in the synchronization.

An important advantage of sPBC is its simple and natural Petri net semantics, as it occurs in plain PBC, but obviously extended with continuous stochastic and deterministically zero timing. The denotational semantics of sPBC is defined by taking as semantic domain a special class of GSPNs, called gs-boxes. Each immediate multiaction will correspond to an immediate transition in a gs-box, and each stochastic multiaction will correspond to a stochastic transition in a gs-box. As in GSPNs, in case of conflict, immediate multiactions are executed before stochastic multiactions. Furthermore, when two or more immediate multiactions are simultaneously activated, the one with the highest priority will be executed. If some of such immediate multiactions have the same (highest) priority level, then we apply the *branching policy*, according to their weights. If we have only stochastic multiactions (transitions) activated, then we will adopt the *race policy*. Thus, whenever two or more stochastic multiactions (transitions) are executable (while any immediate ones are not), the fastest stochastic multiaction will be executed.

We have preferred sPBC over dtsPBC and dtsiPBC, since for our design and analysis purposes, a formalism with a continuous stochastic time and a GSPNs-based denotational semantics, which is also supported by a computer tool, appeared to be more appropriate. Thus, the main distinctive features of sPBC that we consider useful for the intended system modelling are: immediate and stochastic multiactions, suitable and practical algebraic operators, continuous stochastic time and Petri net semantics.

The main goal of this paper is therefore to show how sPBC can be used as a tool to model and analyze some information science phenomena, where time, concurrency and synchronization play a crucial role. We have chosen the Video Conference System (VCS) case study, based on the model from [9], because it allows us to illustrate the power, flexibility and ease to use of this algebra thanks to its specific features. The users communicate to each other in VCS via terminals that establish connections using switches, each initiating a separate conference. First, we consider the case of 2 terminals (users) and 1 switch (conference) and we calculate the corresponding performance measures. We then extend VCS to n terminals and 1 or 3 switches. Finally, we model 3 terminals and 1 special new switch, with 1 caller user and 2 callee ports. In this case study, we have immediate and stochastic multiactions that are combined with the operators of sPBC: sequence, choice, parallel, synchronization, restriction, scoping and iteration.

The paper is structured as follows. Section 2 describes the syntax of sPBC and its semantics, by using both the operational and denotational approaches. The Video Conference System application example is presented in Section 3. The concluding Section 4 contains a summary of the results obtained and some hints for our future work.

2 Stochastic Petri Box Calculus

sPBC (stochastic Petri Box Calculus) is defined as a stochastic extension of PBC, taking both stochastic and immediate multiactions, but keeping the compositional nature of PBC. In the following subsections we present the syntax of the language, its operational semantics, and the denotational semantics defined by using the so-calles gs-boxes.

2.1 Syntax and overview of the language

From now onwards we will use the following notation:

– $\mathscr{A}$ will be a countable set of action names, as in CCS (Calculus of Communicating Systems) [20], for each $a \in \mathscr{A}$, there exists $\widehat{a} \in \mathscr{A}$, such that $a \neq \widehat{a}$ and $\widehat{\widehat{a}} = a$. Letters $a, b, \widehat{a}, \ldots$ will be used to denote the elements of $\mathscr{A}$.

- $\mathscr{L} = \mathscr{B}(\mathscr{A})$ will represent the set of all finite multisets of elements in $\mathscr{A}$, called *multiactions*.
- The alphabet of $\alpha \in \mathscr{L}$ is defined by: $A(\alpha) = \{a \in \mathscr{A} \mid \alpha(a) > 0\}$.
- We will consider relabelling functions $f : \mathscr{A} \to \mathscr{A}$, which are the functions that preserve conjugates: $\forall a \in \mathscr{A},\ f(\widehat{a}) = \widehat{f(a)}$ (we will only consider bijective relabelling functions).
- $\mathscr{SL} = \{<\alpha, r> \mid \alpha \in \mathscr{L}, r \in \mathbb{R}^+\}$ represents the set of all stochastic multiactions. We allow the same multiaction $\alpha \in \mathscr{L}$ to have different stochastic rates in the same specification.
- $\mathscr{IL} = \{<\alpha, \infty_{l,p}> \mid \alpha \in \mathscr{L}, l \in \mathbb{N}, p \in \mathbb{R}^+\}$ represents the set of all immediate multiactions. We also allow the same multiaction $\alpha \in \mathscr{L}$ to have some different immediate parameters in the same specification.
- Finally, we define the synchronization of multiactions $\alpha \oplus_a \beta =_{def} \gamma$, as follows:

$$\gamma(b) = \begin{cases} \alpha(b) + \beta(b) - 1 & \text{if } b = a \vee b = \widehat{a} \\ \alpha(b) + \beta(b) & \text{otherwise} \end{cases}$$

which is only applicable when either $a \in A(\alpha)$ and $\widehat{a} \in A(\beta)$, or $\widehat{a} \in A(\alpha)$ and $a \in A(\beta)$.

As in plain PBC, static s-expressions are used to describe the structure of a concurrent system, while dynamic s-expressions describe the current state of a system (they correspond to unmarked and marked Petri nets, respectively). As a system evolves by executing multiactions, the dynamic s-expression describing its current state changes. This is captured by means of both overbars and underbars that decorate the static s-expression. Static s-expressions of sPBC are those defined by the following BNF expression:

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

where $\widetilde{r} \in \mathbb{R}^+ \cup Inf$, and $Inf = \{\infty_{l,p} \mid\ l \in \mathbb{N},\ p \in \mathbb{R}^+\}$. Thus, if $\widetilde{r} = r \in \mathbb{R}^+$ then $<\alpha, r> \in \mathscr{SL}$ is a *stochastic 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. On the other hand, if $\widetilde{r} = \infty_{l,p} \in Inf$, then $<\alpha, \infty_{l,p}>$ is an *immediate multiaction*, where l is its level of priority, and p is its weight, and it corresponds to the simultaneous execution of all the actions in α, but no time elapses in this case.

For the remaining operators of sPBC we have the following intuitive meaning: $E_1; E_2$ stands for the sequential execution of E_1 and E_2, $E_1 \,\square\, E_2$ is the choice between its argument, $E[f]$ 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 or immediate multiactions $<\alpha, \widetilde{r}>$ with either $a \in A(\alpha)$ or $\hat{a} \in A(\alpha)$. The parallel operator, $\parallel$, represents the (independent) parallel execution of both components, where as in PBC there is no any synchronization embedded in the operator. Synchronization is introduced by the operator *sy*, thus the process $E\,sy\,a$ behaves in the same way as E, but it can also execute those new multiactions generated by the synchronization of a pair of actions $(a, \hat{a})$. $[a:E]$ is the derived operator *scoping* defined by $[a:E] = (E\,sy\,a)\,rs\,a$. Finally, the iteration operator $[E_1 * E_2 * E_3]$ represents the process that performs E_1, then executes several (possibly 0) times E_2, and finishes after performing E_3. 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 this operator with the restriction operator.

We will denote static s-expressions by the following letters: $E, F, E_i, \ldots$, and the set of static s-expressions by *StatExpr*. In sPBC without immediate multiactions 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 [14]. Terms fulfilling this restriction are called *regular terms*, and the operational semantics is 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 that illustrate the need of this restriction in [14,16]). Thus, the following static s-expression:

$$E = (<a, r_1> \parallel <a, r_2>) \,\square\, <b, r_3>$$

is not regular, and consequently, it is not allowed in sPBC without immediate multiactions.

Then, we define regularity as follows. Regular static s-expressions are those static s-expressions E of sPBC that are constructed as follows:

$$\begin{aligned} D ::= {} & <\alpha, \widetilde{r}> \mid D;E \mid D\,sy\,a \mid D\,rs\,a \mid D[f] \mid [a:D] \mid \\ & D \,\square\, D \mid [D * D * D] \\ E ::= {} & <\alpha, \widetilde{r}> \mid E;E \mid E\,sy\,a \mid E\,rs\,a \mid E[f] \mid [a:E] \mid \\ & E \parallel E \mid D \,\square\, D \mid [E * D * D] \end{aligned}$$

where $\widetilde{r} \in \mathbb{R}^+ \cup Inf$.

2.2 Operational Semantics of sPBC

The operational semantics of sPBC is defined on dynamic s-expressions G, which are obtained from the static s-expressions, by 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:

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

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 [4,5,7]. We firstly present the inaction rules in Table 1. They are introduced 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.

Table 1: Inaction rules

$$\overline{E;F} \xrightarrow{\emptyset} \overline{E};F \qquad E;F \xrightarrow{\emptyset} E;\overline{F}$$

$$E;\underline{E} \xrightarrow{\emptyset} E;\underline{E}$$

$$\overline{E \Box F} \xrightarrow{\emptyset} \overline{E} \Box F \qquad \overline{E \Box F} \xrightarrow{\emptyset} E \Box \overline{F}$$

$$\underline{E} \Box F \xrightarrow{\emptyset} \underline{E \Box F} \qquad E \Box \underline{E} \xrightarrow{\emptyset} \underline{E \Box F}$$

$$\frac{\forall op \in \{;, \Box\}, G \xrightarrow{\emptyset} G'}{G op E \xrightarrow{\emptyset} G' op E} \qquad \frac{\forall op \in \{;, \Box\}, G \xrightarrow{\emptyset} G'}{E op G \xrightarrow{\emptyset} E op G'}$$

$$\overline{E \| F} \xrightarrow{\emptyset} \overline{E} \| \overline{F} \qquad \underline{E} \| \underline{F} \xrightarrow{\emptyset} \underline{E \| F}$$

$$\frac{G_1 \xrightarrow{\emptyset} G_1'}{G1 \| G_2 \xrightarrow{\emptyset} G_1' \| G_2} \qquad \frac{G_2 \xrightarrow{\emptyset} G_2'}{G_1 \| G_2 \xrightarrow{\emptyset} G_1 \| G_2'}$$

$$\overline{E[f]} \xrightarrow{\emptyset} \overline{E}[f] \qquad \underline{E}[f] \xrightarrow{\emptyset} \underline{E[f]}$$

$$\frac{G \xrightarrow{\emptyset} G'}{G[f] \xrightarrow{\emptyset} G'[f]}$$

$$\overline{E sy a} \xrightarrow{\emptyset} \overline{E} sy a \qquad \overline{E rs a} \xrightarrow{\emptyset} \overline{E} rs a$$

$$\underline{E} rs a \xrightarrow{\emptyset} \underline{E rs a}$$

$$\frac{\forall op \in \{sy, rs\}, G \xrightarrow{\emptyset} G'}{G op a \xrightarrow{\emptyset} G' op a}$$

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

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

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

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

Definition 1. We say that a regular dynamic s-expression G is *operative* if it is not possible to apply any inaction rule to it. For instance, $\overline{<\alpha, r>} \Box <\beta, s>$ is operative, but not $<\alpha, r> \Box <\beta, s>$. We will denote the set of all the operative regular dynamic s-expressions by *OpReDynExpr*.

We say that an operative s-expression G is *immediate* if it has over-barred some immediate multiactions. For instance, $\overline{<\alpha, \infty_{l,p}>} \Box <\beta, s>$ is immediate. We will denote the set of all the immediate operative regular dynamic s-expressions by *IOpReDynExpr*. □

Definition 2. 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 $[G]_{\equiv}$. □

Before defining the rules of the operational semantics we need to introduce some definitions. First, we need to detect all the possible sets of bags of multiactions that can potentially be executed concurrently by the corresponding operative regular dynamic s-expression G, which will be called $BC(G)$. Next, we define the *level* of its class, $level([G]_{\equiv})$, and finally we define $now(G)$, which gives us the specific bags of stochastic or immediate multiactions that can be executed from G.

Definition 3. We define the sets of bags of multiactions that can potentially be executed concurrently from an operative dynamic regular s-expression as follows:

$$BC : OpReDynExpr \longrightarrow \mathscr{P}(\mathscr{B}(\mathscr{SL} \cup \mathscr{IL}))$$

– If $G \in OpReDynExpr$ is final, i.e. $G = \underline{E}$, we take $BC(G) = \emptyset$.

– If $G \in OpReDynExpr$ is not final, we distinguish the following cases:

  – $BC(\overline{<\alpha, \tilde{r}>}) = \{\{<\alpha, \tilde{r}>\}\}$

  – 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 $a, \widehat{a} \notin A(\gamma)$), $\gamma \in BC(G sy a)$, $f(\gamma) \in BC(G[f]), \gamma \in BC([G * E * F]), \gamma \in BC([E * G * F]), \gamma \in BC([E * F * G])$.

  • If $\gamma_1 \in BC(G)$, then $\gamma_1 \in BC(G \| H)$.

  • If $\gamma_2 \in BC(H)$, then $\gamma_2 \in BC(G \| H)$.

  • 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 $<\alpha, \tilde{r_1}>, <\beta, \tilde{r_2}> \in \gamma$, (with either $<\alpha, \tilde{r_1}> \neq <\beta, \tilde{r_2}>$ or they are two different instances of the same stochastic or immediate multiaction in γ), with $a \in A(\alpha)$, and $\widehat{a} \in A(\beta)$, then: $\gamma' \in BC(G sy a)$, where: $\gamma' = (\gamma + \{<\alpha \oplus_a \beta, \tilde{R}>\}) \setminus \{<\alpha, \tilde{r_1}>, <\beta, \tilde{r_2}>\}$ and $\tilde{R} \in \mathbb{R}^+ \cup Inf$ is the parameter of the new multiaction, to be later defined (see rules **Sy2** and **Sy2$_i$** in Table3).

□

Now, level of $[G]_{\equiv}$ is defined as the maximum level of priority of the immediate multiactions that can potentially be executed from an operative in $[G]_{\equiv}$, and -1 when there is no executable immediate multiaction from an operative in $[G]_{\equiv}$. Formally:

Definition 4. Let $G \in OpReDynExpr$, then we define $level([G]_{\equiv})$ as follows: $level([G]_{\equiv}) = \{\max\{l \mid \{<\alpha, \infty_{l,p}>\} \in BC(G_i), G_i \in [G]_{\equiv}\}$.
By convention, $level([G]_{\equiv}) = -1$ if the previous set is empty. □

Stochastic multiactions are only executable from an operative regular expression G when $level([G]_{\equiv}) = -1$. Furthermore, an immediate multiaction $<\alpha, \infty_{l,p}>$ is only executable from G if $level([G]_{\equiv}) = l$. This is captured by the following function *now*.

Definition 5. We define the set of stochastic or immediate multiactions than can be executed from an operative dynamic s-expressions as follows:
$now : OpReDynExpr \longrightarrow \mathscr{P}(\mathscr{B}(\mathscr{SL})) \cup \mathscr{P}(\mathscr{B}(\mathscr{IL}))$
where

$$now(G) = \{\gamma \in BC(G) \mid level([G]_{\equiv}) = -1 \quad \vee \\ (level([G]_{\equiv}) = l \in \mathbb{N} \wedge \forall < \alpha, \infty_{l',p} > \in \gamma,\ l = l')\}$$

□

The following example illustrates the above definitions.

Example 1. Let us consider $G =$

$$\overline{< \{a\}, \infty_{2,2} >} \| \overline{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >} \| \overline{< \{b\}, \infty_{1,2} >} \| \overline{< \{c\}, 0.5 >}$$

Then:

$$BC(G) = \\ \{\{< \{a\}, \infty_{2,2} >\}, \{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >\}, \{< \{b\}, \infty_{1,2} >\}, \\ \{< \{c\}, 0.5 >\}, \{< \{a\}, \infty_{2,2} >, < \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{b\}, \infty_{1,2} >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{c\}, 0.5 >\}, \\ \{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, < \{b\}, \infty_{1,2} >\}, \\ \{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, < \{c\}, 0.5 >\}, \\ \{< \{b\}, \infty_{1,2} >, < \{c\}, 0.5 >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, < \{b\}, \infty_{1,2} >\}, \\ \{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, < \{b\}, \infty_{1,2} >, < \{c\}, 0.5 >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{b\}, \infty_{1,2} >, < \{c\}, 0.5 >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, < \{c\}, 0.5 >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >, \\ < \{b\}, \infty_{1,2} >, < \{c\}, 0.5 >\}\},$$

$level([G]_{\equiv}) = 2$, and

$$now(G) = \{\{< \{a\}, \infty_{2,2} >\}, \{< \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >\}, \\ \{< \{a\}, \infty_{2,2} >, < \{\widehat{a}, \widehat{a}\}, \infty_{2,3} >\}\}.$$

□

We have two types of transitions:

– *Stochastic transitions*, which have the following form: $G \stackrel{<\alpha,r>}{\longrightarrow} G'$, where G is regular and operative, $r \in \mathbb{R}^+$, and $\{< \alpha, r >\} \in now(G)$.

– *Immediate transitions*, which have the following form: $G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'$, where G is regular, operative and immediate, and $\{< \alpha, \infty_{l,p} >\} \in now(G)$.

The rules defining the stochastic and immediate transitions are presented in Table 2, together with those corresponding to the synchronization operator, which will be described in detail later. We assume that all dynamic s-expressions that appear on the left-hand sides of each transition in the rules are regular and operative.

Let us now see the semantics of the synchronization. We have to distinguish two cases:

– *Synchronization between two immediate multiactions*: They must have the same priority, the new multiaction has this priority, and its weight is the product of the weights of the arguments. The purpose of this definition is to benefit the synchronization with respect to the single execution of the arguments. Notice that, in fact, this only occurs when the weights of the arguments are greater than 1, which is the usual case.

– *Synchronization between two stochastic multiactions*: In this case, in order to define the rates for the stochastic multiactions generated by a synchronization, we need to identify the situations of conflict (or competition, but we prefer the term *conflict* since we only need to consider those stochastic multiactions with exactly the same multiaction). Concretely, for each operative regular dynamic s-expression G we define the multiset of associated conflicts for every instance of a stochastic multiaction $< \alpha, r >_i$ ($r \in \mathbb{R}^+$) executable from G, which we will denote by $Conflict(G, < \alpha, r >_i)$. We only need to consider those stochastic multiactions in conflict executing the same multiaction α. We will denote this multiset of conflicts by $Conflict(G, < \alpha, r >_i)$, although we will omit the subindex i if it is clear which instance of $< \alpha, r >$ we are considering.

Definition 6. We define the following partial function:

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

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

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 any bijective function f, $Conflict(G[f], < f(\alpha), r >) = f(C)$,
   (e) For the choice operator we need to distinguish the following two cases:
   - If $G \not\equiv \overline{E}$: $Conflict(G \,\square\, F, < \alpha, r >) = Conflict(F \,\square\, G, < \alpha, r >) = C$
   - If $G \equiv \overline{E}$: $Conflict(G \,\square\, F, < \alpha, r >) = Conflict(F \,\square\, G, < \alpha, r >) = C + \{\!\{< \alpha, r_j > \mid \exists H_i \in OpReDynExpr, H_i \equiv \overline{F} \text{ and } H_i \stackrel{<\alpha,r_j>}{\longrightarrow} H_i'\}\!\}$
   (f) For the iteration operator we have:
   – $Conflict([G * E * F], < \alpha, r >) = C$
   – For the two last arguments of an iteration, we also need to consider two cases:

Table 2: Rules defining the stochastic and immediate transitions (I).

(B)	$\dfrac{}{<\alpha,\tilde{r}> \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} <\alpha,\tilde{r}>}$	$\tilde{r} \in \mathbb{R}^+ \cup Inf$
(S1)	$\dfrac{G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'}{G;F \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G';F}$	$\tilde{r} \in \mathbb{R}^+ \cup Inf$
(S2)	$\dfrac{G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'}{E;G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} E;G'}$	$\tilde{r} \in \mathbb{R}^+ \cup Inf$
(Rs)	$\dfrac{G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'}{Grsa \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'rsa}$	$a,\hat{a} \notin A(\alpha), \quad \tilde{r} \in \mathbb{R}^+ \cup Inf$
(Re)	$\dfrac{G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'}{G[f] \stackrel{<f(\alpha),\tilde{r}>}{\longrightarrow} G'[f]}$	$\tilde{r} \in \mathbb{R}^+ \cup Inf$
(C1)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G\|H \stackrel{<\alpha,r>}{\longrightarrow} G'\|H}$	if $level([H]_{\equiv}) = -1, \quad r \in \mathbb{R}^+$
(C1i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{G\|H \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'\|H}$	if $level([H]_{\equiv}) \leq l$
(C2)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{H\|G \stackrel{<\alpha,r>}{\longrightarrow} H\|G'}$	if $level([H]_{\equiv}) = -1, \quad r \in \mathbb{R}^+$
(C2i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{H\|G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} H\|G'}$	if $level([H]_{\equiv}) \leq l$
(E1)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{G\Box F \stackrel{<\alpha,r>}{\longrightarrow} G'\Box F}$	if $G \not\equiv \overline{E} \vee (G \equiv \overline{E} \wedge level([\overline{F}]_{\equiv}) = -1), \quad r \in \mathbb{R}^+$
(E1i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{G\Box F \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'\Box F}$	if $G \not\equiv \overline{E} \vee (G \equiv \overline{E} \wedge level([\overline{F}]_{\equiv}) \leq l)$
(E2)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{E\Box G \stackrel{<\alpha,r>}{\longrightarrow} E\Box G'}$	if $G \not\equiv \overline{F} \vee (G \equiv \overline{F} \wedge level([\overline{E}]_{\equiv}) = -1), \quad r \in \mathbb{R}^+$
(E2i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{E\Box G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} E\Box G'}$	if $G \not\equiv \overline{F} \vee (G \equiv \overline{F} \wedge level([\overline{E}]_{\equiv}) \leq l)$
(It1)	$\dfrac{G \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} G'}{[G*E*F] \stackrel{<\alpha,\tilde{r}>}{\longrightarrow} [G'*E*F]}$	$\tilde{r} \in \mathbb{R}^+ \cup Inf$
(It2)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{[E*G*F] \stackrel{<\alpha,r>}{\longrightarrow} [E*G'*F]}$	if $G \not\equiv \overline{E_1} \vee (G \equiv \overline{E_1} \wedge level([\overline{F}]_{\equiv}) = -1), \quad r \in \mathbb{R}^+$
(It2i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{[E*G*F] \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} [E*G'*F]}$	if $G \not\equiv \overline{E_1} \vee (G \equiv \overline{E_1} \wedge level([\overline{F}]_{\equiv}) \leq l)$
(It3)	$\dfrac{G \stackrel{<\alpha,r>}{\longrightarrow} G'}{[E*F*G] \stackrel{<\alpha,r>}{\longrightarrow} [E*F*G']}$	if $G \not\equiv \overline{E_1} \vee (G \equiv \overline{E_1} \wedge level([\overline{F}]_{\equiv}) = -1), \quad r \in \mathbb{R}^+$
(It3i)	$\dfrac{G \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} G'}{[E*F*G] \stackrel{<\alpha,\infty_{l,p}>}{\longrightarrow} [E*F*G']}$	if $G \not\equiv \overline{E_1} \vee (G \equiv \overline{E_1} \wedge level([\overline{F}]_{\equiv}) \leq l)$

- If $G \not\equiv \overline{E'}$:
$Conflict([E*G*F], <\alpha,r>) =$
$Conflict([E*F*G], <\alpha,r>) = C$
- If $G \equiv \overline{E'}$:
$Conflict([E*G*F], <\alpha,r>) =$
$Conflict([E*F*G], <\alpha,r>) =$
$C + \{\!|<\alpha,r_j> \mid \exists H_i \in OpReDynExpr,$
$H_i \equiv \overline{F}$ and $H_i \stackrel{<\alpha,r_j>}{\longrightarrow} H_i' \}\!\}$

(g) $Conflict(Gsya, <\alpha,r>) = C$,

3. Let
$\{<\alpha_1,r_1>, <\alpha_2,r_2>\} \in BC(Gsya), a \in A(\alpha_1)$,
$\hat{a} \in A(\alpha_2)$ and $Gsya \stackrel{<\alpha_1 \oplus_a \alpha_2, R_{12}>}{\longrightarrow} G'sya$ obtained by applying rule **Sy2**. Then:
$Conflict(Gsya, <\alpha_1 \oplus_a \alpha_2, R_{12}>) =$
$\{\!|<\alpha_1 \oplus_a \alpha_2, R_{ij}> \mid <\alpha_1,r_i> \in C_1,$
$<\alpha_2,r_j> \in C_2,$ where
$R_{ij} = \frac{r_i}{cr(Gsya,<\alpha_1,r_1>)} \frac{r_j}{cr(Gsya,<\alpha_2,r_2>)} \cdot$
$\min_{i=1,2}\{cr(Gsya, <\alpha_i,r_i>)\}\}\!|$, taking:
$C_i = Conflict(Gsya, <\alpha_i,r_i>), i = 1,2,$ and

$cr(G,<\alpha,r>_i)$ is the so-called *conflict rate* for G and $<\alpha,r>_i$, defined by:

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

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

□

The rules for the synchronization are shown in Table 3. The first rule captures that $Gsya$ preserves the behaviour of G. **Sy2$_i$** captures the aforementioned behaviour of the synchronization between two immediate multiactions. With respect to **Sy2**, the synchronization between two stochastic multiactions, we take as rate of the new stochastic multiaction the minimum of the conflict rates of each one, weighted by a factor, which implies that, for instance, the following s-expressions are equivalent:

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

According to the race policy that we apply to resolve the choice, we obtain for both dynamic s-expressions, $\overline{E}$ and $\overline{F}$, the same delay for executing the multiaction α : $r = \sum_{i=1}^{n} r_i$. As a consequence, the CTMCs obtained from the transition systems of $\overline{E}$ and $\overline{F}$ would be the same and, thus, we may consider that both s-expressions are stochastically equivalent. In order to capture that we have defined the rate of synchronization of two stochastic multiactions by using the so-called *conflict rates* [16], which are based on the *apparent rates* of PEPA [11], but with the advantage that, using the conflict rates, we obtain a static translation to Petri Nets, while in PEPA the rates of the transitions of the corresponding Stochastic Petri Net can be marking dependent [23].

The following examples illustrate the above definitions and rules.

Example 2. Let us consider:

$$G = (\overline{<\{a\},1>} \| \overline{<\{\widehat{a},\widehat{a}\},2>} \| (\overline{<\{a\},3>} \Box <\{a\},4>) sya); <b,\infty_{2,3}>,$$

$$H = (\overline{<\{a\},1>} \| \underline{<\{\widehat{a},\widehat{a}\},2>} \| (\underline{<\{a\},3>} \Box <\{a\},4>) sya); <b,\infty_{2,3}>,$$

G is a regular operative dynamic s-expression that has no executable immediate multiactions, and $\{<\{\widehat{a},\widehat{a}\},2>,<\{a\},3>\} \in now(G)$.
Furthermore,
$Conflict(G,<\{\widehat{a},\widehat{a}\},2>) = \{<\{\widehat{a},\widehat{a}\},2>\}$
and $cr(G,<\{\widehat{a},\widehat{a}\},2>) = 2$,
$Conflict(G,<\{a\},3>) = \{<\{a\},3>,<\{a\},4>\}$
and $cr(G,<\{a\},3>) = 7$.
Then, by applying rule **Sy2**, we obtain the following transition: $G \xrightarrow{<\{\widehat{a}\},R>} H$, where $R = \frac{3}{7}\min\{2,7\} = \frac{6}{7}$.

□

Example 3. Let us consider:

$$G = \overline{<\{a\},\infty_{2,2}>} \| \underline{<\{\widehat{a},\widehat{a}\},\infty_{2,3}>} \| \overline{<\{b\},\infty_{1,2}>} \| \overline{<\{c\},0.5>} sya$$

Then, $\{<\{a\},\infty_{2,2}>,<\{\widehat{a},\widehat{a}\},\infty_{2,3}>\} \in now(G)$ By applying rule **Sy2$_i$** we obtain the following transition (synchronizing $<\{a\},\infty_{2,2}>$ and $<\{\widehat{a},\widehat{a}\},\infty_{2,3}>$): $G \xrightarrow{<\{\widehat{a}\},\infty_{2,6}>} H$, where

$$H = \underline{<\{a\},\infty_{2,2}>} \| \underline{<\{\widehat{a},\widehat{a}\},\infty_{2,3}>} \| \overline{<\{b\},\infty_{1,2}>} \| \overline{<\{c\},0.5>} sya$$

□

Definition 7. For each $G \in ReDynExpr$, we define the set of all dynamic s-expressions that can be derived from $[G]_{\equiv}$, as follows:

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

□

We proved in [15,17] that, given $G \in OpReDynExpr$ and $\gamma \in now(G)$, every serialization of γ is executable from G. Moreover, by means of these serializations we always obtain equivalent (with respect to $\equiv$) dynamic s-expressions. On the other hand, for all the possible transition sequences obtained by serialization of γ, if we can apply rule **Sy2** a number of times in order to reach a single stochastic or immediate multiaction, then we conclude that it does not matter in which order rule **Sy2** or **Sy2$_i$** has been applied, neither the transition sequence used, i.e. we will always obtain the same stochastic or immediate multiaction. In fact, if we synchronize stochastic multiactions, the rate of this new stochastic multiaction is the minimum of the conflict rates of the stochastic multiactions that have been synchronized, weighted by a factor, which is the product of the ratios of each rate with respect of its corresponding conflict rate. The same occurs when the multiactions in γ are all immediate, since they all have the same priority, and the application of $Sy2_i$ generates a new multiaction that has the same priority, and its weight is the product of the weights of the arguments.

Definition 8. We define the labelled (multi) transition system of any regular dynamic s-expression 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 stochastic and immediate transitions, given by:

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

Table 3: Rules for the synchronization operator.

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

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

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

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

(**Sy2$_\mathbf{i}$**) Let $\{<\alpha_1,\infty_{l,p_1}>,<\alpha_2,\infty_{l,p_2}>\} \in now(G\,sy\,a)$, $a \in A(\alpha_1)$, $\hat{a} \in A(\alpha_2)$, then

$$\frac{G\,sy\,a \xrightarrow{<\alpha_1,\infty_{l,p_1}>} G_1\,sy\,a \; (\xrightarrow{\emptyset})^* \; G_1^*\,sy\,a \xrightarrow{<\alpha_2,\infty_{l,p_2}>} G_{12}\,sy\,a}{G\,sy\,a \xrightarrow{<\alpha_1 \oplus_a \alpha_2, \infty_{l,p_1 \cdot p_2}>} G_{12}\,sy\,a}$$

In order to compute the number of different instances of each transition $([H]_{\equiv},<\alpha,\tilde{r}>,[J]_{\equiv})$ in A, we consider equivalent all the different ways to derive the same transition by considering the different serializations of the same γ, as we said before. Then, when we apply the rules **Sy2** or **Sy2$_\mathbf{i}$**, the generated stochastic or immediate multiaction can be annotated with the concatenation of the numbering of the stochastic or immediate multiactions involved in the synchronization[1], then when we detect that a permutation of the numbering has been already obtained by a previous application of the corresponding rule, **Sy2** or **Sy2$_\mathbf{i}$**, then that new stochastic or immediate transition will not be considered.

□

Notice that in the labelled transition system, $ts(G)$, of any regular dynamic s-expression G, the distribution of the sojourn time in an arbitrary node can be expressed as a composition of negative exponential and deterministically (with time zero) distributions, depending on whether we have stochastic or immediate multiactions executable from that state. Furthermore, if it is possible to execute some immediate multiactions from a node $[H]_{\equiv}$, then we apply a branching policy according to the weights of the involved immediate multiactions. For instance, if we only have the following two transitions from the node $[H]_{\equiv}$: $([H]_{\equiv},<\alpha,\infty_{l,p_1}>,[J_1]_{\equiv})$ and $([H]_{\equiv},<\beta,\infty_{l,p_2}>,\ [J_2]_{\equiv})$, then we have that $level([H]_{\equiv}) = l$ and the probability to execute $<\alpha,\infty_{l,p_1}>$ is $\frac{p_1}{p_1+p_2}$. On the other hand, if $level([H]_{\equiv}) = -1$, we apply a race policy according to the rates of the involved stochastic multiactions. For instance, if we only have these two transitions from the node $[H]_{\equiv}$: $([H]_{\equiv},<\alpha,r_1>,[J_1]_{\equiv})$ and $([H]_{\equiv},<\beta,r_2>,[J_2]_{\equiv})$, then the probability to execute $<\alpha,r_1>$ is $\frac{r_1}{r_1+r_2}$. In this way, we can recognize the evolution of $ts(G)$ as a semi-Markov stochastic chain.

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

2.3 Denotational semantics

Now, we present a denotational semantics for s-expressions, which is obtained by taking Generalized Stochastic Petri Nets as plain boxes. With this semantics we have a graphical representation of the system, in terms of a GSPN. Therefore, the semantic objects that we use will be called generalized stochastic Petri boxes or just *gs-boxes*. Thus, these *gs-boxes* are essentially GSPNs, but they have the same structure as the Petri boxes of PBC. These boxes of PBC are labelled Petri nets fulfilling some restrictions. 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 $\mathscr{B}(\mathscr{L}) \times \mathscr{L}$; i.e. $\lambda(t)$ is a relation which associates elements of $\mathscr{L}$ to bags of multiactions. By convention, ${}^\circ\Sigma$ and Σ° will denote the set of *e-labelled* places and the set of *x-labelled* places, respectively. Given a place $s \in S$, we will denote by ${}^\bullet s$ ($s^\bullet$) the set of input (output) transitions of s (called preconditions and postconditions of s, respectively). A similar notation is used for preconditions and postconditions of transitions. Both can be easily extended to sets of places and sets of transitions. Then, our boxes are defined to be labelled simple nets such that the following conditions hold: ${}^\circ\Sigma \neq \emptyset \neq \Sigma^\circ$, ${}^\bullet({}^\circ\Sigma) = \emptyset = (\Sigma^\circ)^\bullet$ and $\forall t \in T : {}^\bullet t \neq \emptyset \neq t^\bullet$. A box is said to be *plain* when for every $t \in T$, $\lambda(t)$ is a constant relation, i.e. an element of $\mathscr{L}$.

Definition 9. A plain generalized stochastic Petri box (or just *plain gs-box*) is a tuple $\Sigma = (S,T,W,\ \lambda,\mu)$, where (S,T,W,λ) is a plain box, and

$$\mu : T \longrightarrow \mathbb{R}^+ \cup Inf,\ with\ Inf = \{\infty_{l,p} | l \in \mathbb{N}, p \in \mathbb{R}^+\}$$

If $\mu(t) \in \mathbb{R}^+$ then t is a stochastic transition, with rate $\mu(t)$; otherwise, if $\mu(t) = \infty_{l,p} \in Inf$, then t is an immediate transition, with priority l and weight p. In this way, (S,T,W,μ) is a GSPN. We will denote by T_{exp} the

set of stochastic transitions, and by T_{imm} the set of immediate transitions. □

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

2.3.1 Algebra of gs-boxes

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

Another decision that we must take is the selection of the operator box that we will use for the iteration, since we have two proposals in plain PBC for that purpose (see [5]). One of them provides us with a 1-safe version (with six transitions in the operator box), but there is also a simpler version, which has only three transitions in the operator box. In general, in PBC, with the latter version we may generate 2-safe nets, which only occurs when a parallel behaviour appears at the highest level of the body of the iteration. Nevertheless, in our case, and due to the syntactical restriction introduced, this particular case cannot occur, so that the net obtained will be always 1-safe (for more details see [14]).

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

$$\begin{aligned} \eta &: T \longrightarrow \mathbb{N}^*, \\ \kappa &: T_{exp} \longrightarrow \mathscr{P}(\mathbb{N}^*), \\ \mu &: T \longrightarrow \mathbb{R}^+ \cup Inf \end{aligned}$$

where $\eta(t)$ stands for the numeration of t according to our criterion (enumeration from left to right, and concatenation in case of synchronization), $\kappa(t)$ is only defined if t is a stochastic transition and it identifies the set of stochastic transitions in conflict with t, and $\mu(t)$ is the rate of the exponential distribution for $t \in T_{exp}$, or, if $t \in T_{imm}$, then $\mu(t) = \infty_{l,p}$, i.e. an immediate multiaction with level l and weight p.

These functions will be defined in a structural way, as we construct the corresponding plain gs-box. For each transition $t \in T_{exp}$, we also define its corresponding *conflict rate*, and we will denote it by $cr(t)$:

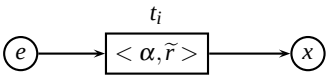
$$cr(t) = \sum_{\eta(t_j) \in \kappa(t)} \mu(t_j)$$

Then, the structure of the net is obtained as in PBC, combining both refinement and relabelling. Consequently, the gs-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:

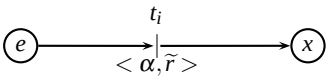
$$\begin{aligned} &Box_{gs}(<\alpha,\tilde{r}>_i) = N_{<\alpha,\tilde{r}>_i}, \\ &Box_{gs}(op(E_1,\ldots,E_n)) = \Omega_{op}(Box_{gs}(E_1),\ldots,Box_{gs}(E_n)), \end{aligned}$$

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

– $Box_{gs}(<\alpha,\tilde{r}>_i) = N_{<\alpha,\tilde{r}>_i} =$



or



depending on whether it is stochastic or not, taking $\eta(t_i) = i$ and $\mu(t) = \tilde{r}$. If $\tilde{r} \in \mathbb{R}^+$, then $\kappa(t_i) = \{i\}$, otherwise, if $\tilde{r} = \infty_{l,p} \in Inf$ then κ is not defined.

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

- $\rho_{id} = \{(\{<\alpha,\tilde{r}>\}, <\alpha,\tilde{r}>) \mid <\alpha,\tilde{r}> \in \mathscr{SL} \cup \mathscr{IL}\}$
- $\rho_{[f]} = \{(\{<\alpha,\tilde{r}>\}, <f(\alpha),\tilde{r}>) \mid <\alpha,\tilde{r}> \in \mathscr{SL} \cup \mathscr{IL}\}$
- $\rho_{rsa} = \{(\{<\alpha,\tilde{r}>\}, <\alpha,\tilde{r}>) \mid <\alpha,\tilde{r}> \in \mathscr{SL} \cup \mathscr{IL} \wedge a,\widehat{a} \notin A(\alpha)\}$

Thus, the corresponding semantic functions are now defined, taking $Box_{gs}(E_i) = (S_i, T_i, W_i, \lambda_i, \mu_i)$ as the plain gs-box corresponding to E_i, and η_i and κ_i are functions for the enumeration and conflict of E_i, $i = 1,2,3$.

[2] A marked plain gs-box is essentially a kind of marked labelled Generalized Stochastic Petri Net, whose behaviour follows the classical *firing rule* of GSPNs.

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

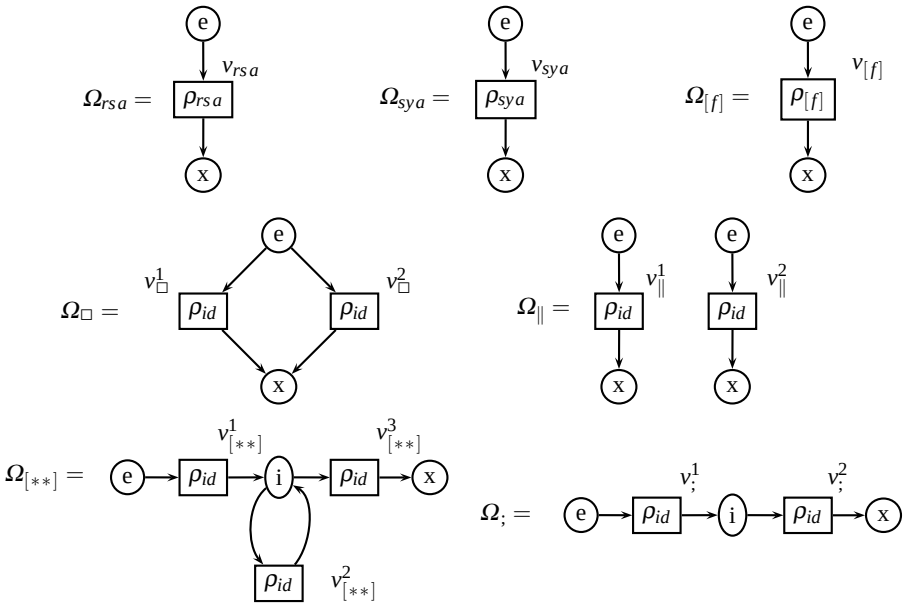


Fig. 1: Operator gs-boxes for sPBC

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

$$\eta(t) = \begin{cases} \eta_1(t) & \text{if } t \in T_1 \\ \eta_2(t) & \text{if } t \in T_2 \end{cases}$$
$$\mu(t) = \begin{cases} \mu_1(t) & \text{if } t \in T_1 \\ \mu_2(t) & \text{if } t \in T_2 \end{cases}$$
$$\kappa(t) = \begin{cases} \kappa_1(t) & \text{if } t \in T_{1_{exp}} \\ \kappa_2(t) & \text{if } t \in T_{2_{exp}} \end{cases}$$

– $Box_{gs}(E_1 \,\|\, E_2) = \Omega_{\|}(Box_{gs}(E_1), Box_{gs}(E_2))$.
η, μ and κ are defined in exactly the same way as in the previous case.

– $Box_{gs}(E_1[f]) = \Omega_{[f]}(Box_{gs}(E_1))$.

$$\eta(t) = \eta_1(t), \quad t \in T_1$$
$$\mu(t) = \mu_1(t), \quad t \in T_1$$
$$\kappa(t) = \kappa_1(t), \quad \text{if } t \in T_{1_{exp}}$$

– $Box_{gs}(E_1 \,\square\, E_2) = \Omega_{\square}(Box_{gs}(E_1), Box_{gs}(E_2))$.

$$\eta(t) = \begin{cases} \eta_1(t) & \text{if } t \in T_1 \\ \eta_2(t) & \text{if } t \in T_2 \end{cases}$$
$$\mu(t) = \begin{cases} \mu_1(t) & \text{if } t \in T_1 \\ \mu_2(t) & \text{if } t \in T_2 \end{cases}$$

$$\kappa(t) = \begin{cases} \kappa_1(t) \cup \kappa_2(t') & \text{if } t \in T_{1_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_1), \\ & \exists t' \in T_{2_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_2), \\ & \lambda(t) = \lambda(t') \\ \kappa_1(t) & \text{if } t \in T_{1_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_1), \\ & \nexists t' \in T_{2_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_2), \\ & \lambda(t) = \lambda(t') \\ \kappa_1(t) & \text{if } t \in T_{1_{exp}}, {}^\bullet t \notin {}^\circ Box_{gs}(E_1) \\ \kappa_2(t) \cup \kappa_1(t') & \text{if } t \in T_{2_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_2), \\ & \exists t' \in T_{1_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_1), \\ & \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_{2_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_2), \\ & \nexists t' \in T_{1_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_1), \\ & \lambda(t) = \lambda(t') \\ \kappa_2(t) & \text{if } t \in T_{2_{exp}}, {}^\bullet t \notin {}^\circ Box_{gs}(E_2) \end{cases}$$

– $Box_{gs}([E_1 * E_2 * E_3]) = \Omega_{[**]}(Box_{gs}(E_1), Box_{gs}(E_2), Box_{gs}(E_3))$.

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

$$\mu(t) = \begin{cases} \mu_1(t) & \text{if } t \in T_1 \\ \mu_2(t) & \text{if } t \in T_2 \\ \mu_3(t) & \text{if } t \in T_3 \end{cases}$$

$$\kappa(t) = \begin{cases} \kappa_1(t) & \text{if } t \in T_{1_{exp}} \\ \kappa_2(t) \cup \kappa_3(t') & \text{if } t \in T_{2_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_2), \\ & \exists t' \in T_{3_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_3), \\ & \lambda(t) = \hat{\lambda}(t') \\ \kappa_2(t) & \text{if } t \in T_{2_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_2), \\ & \nexists t' \in T_{3_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_3), \\ & \lambda(t) = \hat{\lambda}(t') \\ \kappa_2(t) & \text{if } t \in T_{2_{exp}}, {}^\bullet t \notin {}^\circ Box_{gs}(E_2) \\ \kappa_3(t) \cup \kappa_2(t') & \text{if } t \in T_{3_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_3), \\ & \exists t' \in T_{2_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_2), \\ & \lambda(t) = \hat{\lambda}(t') \\ \kappa_3(t) & \text{if } t \in T_{3_{exp}}, {}^\bullet t \in {}^\circ Box_{gs}(E_3), \\ & \nexists t' \in T_{2_{exp}}, {}^\bullet t' \in {}^\circ Box_{gs}(E_2), \\ & \lambda(t) = \hat{\lambda}(t') \\ \kappa_3(t) & \text{if } t \in T_{3_{exp}}, {}^\bullet t \notin {}^\circ Box_{gs}(E_3) \end{cases}$$

– $Box_{gs}(E_1 \, rs \, a) = \Omega_{rs\,a}(Box_{gs}(E_1))$.
$\eta(t) = \eta_1(t)$, $\mu(t) = \mu_1(t)$, and
$\kappa(t) = \kappa_1(t)$ if $t \in T_{1_{exp}}$, $a, \hat{a} \notin \lambda_1(t)$

– $Box_{gs}(E_1 \, sy \, a) = \Omega_{sy\,a}(Box_{gs}(E_1))$.
We take the following relation for the synchronization:
$\rho_{sya} \subseteq (\mathscr{B}(\mathscr{SL}) \times \mathscr{SL}) \cup (\mathscr{B}(\mathscr{IL}) \times \mathscr{IL})$,
as the least relabelling relation containing ρ_{id}, and fulfilling:
$(\Gamma, \alpha + \{a\}) \in \rho_{sya} \quad \wedge \quad (\Delta, \beta + \{\hat{a}\}) \in \rho_{sya}$
then $(\Gamma + \Delta, \alpha + \beta) \in \rho_{sya}$ Thus, ρ_{sya} allows us to obtain the net structure, as well as the multiactions labelling the transitions. Now, for every $t_1, t_2 \in T_{1_{exp}}$, $\lambda_1(t_1) = \alpha + \{a\}$, $\lambda_1(t_2) = \beta + \{\hat{a}\}$, a new stochastic transition t is generated by the synchronization, whose label is $\lambda(t) = \alpha + \beta$, and its rate is computed as follows:

$$\mu(t) = \frac{\mu_1(t_1)}{cr(t_1)} \cdot \frac{\mu(t_2)}{cr(t_2)} \cdot min(cr(t_1), cr(t_2))$$

Moreover,

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

On the other hand, for every $t_1, t_2 \in T_{1_{imm}}$, with $\mu_1(t_1) = \infty_{l_1, p_1}$ and $\mu_1(t_2) = \infty_{l_2, p_2}$, with $\lambda_1(t_1) = \alpha + \{a\}$, $\lambda_1(t_2) = \beta + \{\hat{a}\}$ and $l_1 = l_2 = l$, then a new immediate transition t is generated by the synchronization, whose label is $\lambda(t) = \alpha + \beta$, and

$$\eta(t) = \eta_1(t_1) \,.\, \eta_1(t_2)$$
$$\mu(t) = \infty_{l, p_1 \cdot p_2}$$

Notice that in order not to introduce redundant transitions, we only consider in the plain gs-box a single one of the possible transitions that we can obtain by synchronizing (in different order) the same set of transitions. Furthermore, those stochastic transitions that were in $T_{1_{exp}}$ have the same label, rate, numeration and conflict that they had in $Box_{gs}(E_1)$; and those immediate transitions that were in $T_{1_{imm}}$ have the same label, numeration and immediate information that they had in $Box_{gs}(E_1)$. On the other hand, with this construction we can obtain in principle infinite nets, as it occurs in PBC, but, taking into account that the obtained nets are safe, the arcs having non-unitary weight will not enable the corresponding transitions, and thus, these transitions and arcs can be removed from the net structure, without affecting its behaviour.

Another classical operator of PBC is the *scoping*, which is a derived operator defined by $[a : E] = (E \, sy \, a) \, rs \, a$. Thus we take:
– $Box_{gs}([a : E_1]) = \Omega_{rs\,a}(Box_{gs}(E_1 \, sy \, a))$

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

Theorem 1. For any regular static s-expression E, the transition system $ts(\overline{E})$ associated with $\overline{E}$, and the reachability graph of the marked GSPN $(Box_{gs}(E), M_e)$ are isomorphic.

Proof. See [17]. □

3 Case Study: Video Conference System

In this section we illustrate the applicability of our model with an example in the context of telecommunication systems. It is inspired by the description that appears in [9], which uses Intelligent Network (IN) architecture, but here we consider a simpler version. Our intention is not to make an exhaustive performance evaluation, instead, we intend to show with this example the flexibility and power of sPBC in this area.

We first consider a case in which two users communicate to each other in a Video Conference System (VCS) with a single switch. Let us see a brief description of the system, following the illustration depicted in Table 4, where the events are shown in the order they occur: the

Table 4: VCS Description

caller	calling terminal	switch	called terminal	callee
Pick Up / $Disconnect2 \longrightarrow$				
	$OffHook \longrightarrow$ / $NotOffHook \longrightarrow$			
		$DialTone \longleftarrow$		
Dial				
	$ConReq \longrightarrow$			
		$RoutSignal \longleftarrow$ $Ring \longrightarrow$		
				Pick Up Callee / $Disconnect2 \longleftarrow$
			$ResultRing \longleftarrow$	
		$ConResp \longleftarrow$		
Talk				*Talk*
HangUp				HangUp
		$Disconnect \longleftarrow$ $Disconnect \longrightarrow$		

caller is responsible to initiate the conversation by *Picking Up* the phone. Then, his terminal sends the *Off Hook* signal to the switch, and waits for the *Dial Tone*. Then, he *dials* the number and a *Connection Requirement* (*ConReq*) is sent from his terminal to the switch. After that, the switch sends a *Routing Signal* to the caller and a *Ring* signal to the callee at the same time. When the callee *Picks Up* his terminal in order to answer, it sends a *Result Ring* signal to the switch, and it sends a *Connection Response* (*ConRes*) signal to the caller, and the conversation can now be initiated. When both the callee and the caller *Hang Up*, the system is disconnected by a double *Disconnect* signal, sent from the switch to the callee and the caller.

On the other hand, when the caller *Picks Up*, but it is not possible for his terminal to send an *Off Hook* signal to the switch, possibly because it is busy, then the action *NotOffHook* is introduced to return to the starting point.

In order to model this system with sPBC, we first identify which actions are immediate, and which ones are stochastic. The following actions are considered to be immediate, because the time required to perform these actions is negligible: *ConReq*, *ResultRing*, *ConResp* and *HangUp*; all of them with the same priority (1) and weight (1). But we also consider as immediate actions *OffHook* (with priority 2) and *NotOffHook* (with priority 1). Notice that priorities are used to enforce the execution of *OffHook* as soon as it becomes permitted. Then, action *NotOffHook* can only be performed when *OffHook* cannot be executed.

On the other hand, *PickUpCallee* is the action corresponding to the answer of the callee user (with priority 1 and weight 8). However, we also consider the possibility for him not to answer, and thus, we have introduced the action *Disconnect2*, with priority 1 and weight 2, so as to associate to this action a probability of 0.2, since it is in conflict with the *PickUpCallee* which has weight 8

The other multiactions have a random delay associated which follows a negative exponential distribution. We have considered the following values: every 12 minutes the caller *PickUps* the phone in order to make a call, so the rate of *PickUp* is $r = 1/12$. The switch takes 3 seconds ($1/20$ min) to transmit the *DialTone* ($r = 20$), the caller takes 10 seconds ($1/6$ min) to *Dial* ($r = 6$), the switch takes 30 seconds ($1/2$ min) ($r = 2$) to send both a *RoutSignal* and a *Ring* signal at the same time. *Talk* takes 3 minutes ($r = 1/3$), and finally *Disconnect* takes 1 second ($1/60$ min, $r = 60$). Conjugate actions are used to represent the receiving of signals, whereas non-conjugate actions represent the sending of them. Notice that we use weights 1 for the conjugate of immediate multiactions, and the same priority for their partner actions, in order to obtain as final weight that of the non-conjugate action. In a similar way, we assign for conjugates of stochastic multiactions a value greater than

Table 5: VCS specification in sPBC

$Terminal_i =$
$[< b, \infty_{1,1} > * Caller \,\Box\, Callee * < f, \infty_{1,1} >] rs\ f$

$Caller =$
$< PickUp, 1/12 >; (< OffHook, \infty_{2,1} >; < \widehat{DialTone}, 200 >;$
$< Dial, 6 >; < ConReq, \infty_{1,1} >; < \widehat{RoutSignal}, 20 >;$
$(< \widehat{ConResp}, \infty_{1,1} >; < Talk, 1/3 > ; < \widehat{HangUp}, \infty_{1,1} >;$
$< \widehat{Disconnect}, 600 >) \,\Box\, < Disconnect2, \infty_{1,1} >) \,\Box$
$< NotOffHook, \infty_{1,1} >$

$Callee =$
$< \widehat{Ring}, 20 >; (< PickUpCallee, \infty_{1,8} >; < ResultRing, \infty_{1,1} >;$
$< Talk, 1/2 >; < HangUp, \infty_{1,1} >; < \widehat{Disconnect}, 600 >) \,\Box$
$< Disconnect2, \infty_{1,2} >$

$Switch =$
$[< b, \infty_{1,1} > * < \widehat{OffHook}, \infty_{2,1} >; < DialTone, 20 >;$
$< \widehat{ConReq}, \infty_{1,1} >; < \{RoutSignal, Ring\}, 2 >;$
$(< \widehat{ResultRing}, \infty_{1,1} >; < ConResp, \infty_{1,1} >;$
$< \{\widehat{Talk}, \widehat{Talk}\}, 10/3 >; < \{Disconnect, Disconnect\}, 60 >) \,\Box$
$< \{\widehat{Disconnect2}, \widehat{Disconnect2}\}, \infty_{1,1} > * < f, \infty_{1,1} >] rs\ f$

their non-conjugate associated actions, in order to get as synchronization rate that of the non-conjugate action. The corresponding sPBC specification is shown in Table 5. Notice that we have introduced an initial action b and a final action f within the iteration, the latter is also restricted in order to enforce a repetitive infinite behavior.

The whole system is therefore described by the process:

$$VCS = [\, A : Terminal_1 \| Terminal_2 \| Switch \,]$$

where $A = \{$*OffHook*, *DialTone*, *ConReq*, *RoutSignal*, *Ring*, *ResultRing*, *ConResp*, *Talk*, *HangUp*, *Disconnect*, *Disconnect2*$\}$

All of stochastic multiactions obtained by synchronization have as rate the minimum of the involved rates. For the action *Talk* we have a rate of $1/3$ for the caller, $1/2$ for the callee and $10/3$ for the switch. The rate for the resulting synchronization action is $r = min\{1/3, 1/2, 10/3\} = 1/3$, i.e. the average time of a conversation is 3 minutes.

For the synchronization of immediate multiactions (all of them having the same priority), the new weight is obtained by multiplying the involved weights. For example, if the callee does not answer, it follows that the weight of his *Disconnect2* action is 2, and considering that the action $\widehat{Disconnect2}$ of the caller and the multiaction $\{\widehat{Disconnect2}, \widehat{Disconnect2}\}$ of the switch have both weight 1, then, if we synchronize these three immediate multiactions, the obtained weight is 2 and, since the weight of *PickUpCallee* is 8, it follows that the probability for a call not to be answered is 0.2.

The corresponding plain gs-boxes are shown in Figures 2 to 4. Notice that exit places are isolated, as a consequence of the restriction over f, so we do not introduce them in the tool GreatSPN [8,12]. In the figures presented conjugates are shown by prefixing the actions with 'conj', due to the use of the GreatSPN tool. Furthermore, we just take the Petri net obtained by removing the initial places, i.e. we only consider the repetitive net obtained after initialization, thus we have an ergodic model, and we are able to make a performance analysis. The obtained throughputs of the relevant transitions of this model are shown in Table 6.

Table 6: Throughputs

PickUp	0.058035
OffHook	0.054764
DialTone	0.054764
Dial	0.054764
ConReq	0.054764
RoutSignal	0.054764
ConResp	0.043811
Talk	0.043811
HangUp	0.043811
Disconnect	0.043811
Disconnect2	0.010953
NotOffHook	0.003271

Thus, if this system is working for 10 hours (600 minutes) then we obtain the following approximated estimations: a user *Picks Up* his terminal about 35 times (600×0.058035), in two of them he does not receive the signal *Off Hook*, and in 33 he gets the signal and *Dials*. Additionally, from the table we conclude that for about 7 times the callee will not answer the call, and 26 times there has been a conservation.

Let us now analyse the productivity of the switch. In this case we have obtained that the switch is idle with probability 0.65717. Of course, the values in Table 6 change according to the parameters introduced. For instance, when the caller *Picks Up* every 8 minutes in average, the probability for the switch to be idle becomes 0.56101.

In this first version, multiactions are used in a limited way, only the *Switch* uses this capability of the model. However, we can scale this system by adding some new terminals in the following way:

$$VideoConference_n = [\, A : Terminal_1 \| Terminal_2 \| \ldots \| Terminal_n \| Switch \,]$$

where
$A =$
$\{$*OffHook*, *DialTone*, *ConReq*, *RoutSignal*, *Ring*, *ResultRing*, *ConResp*, *Talk*, *HangUp*, *Disconnect*, *Disconnect2*$\}$

As we have only one switch, a single videoconference is permitted at a time, and involving just two users.

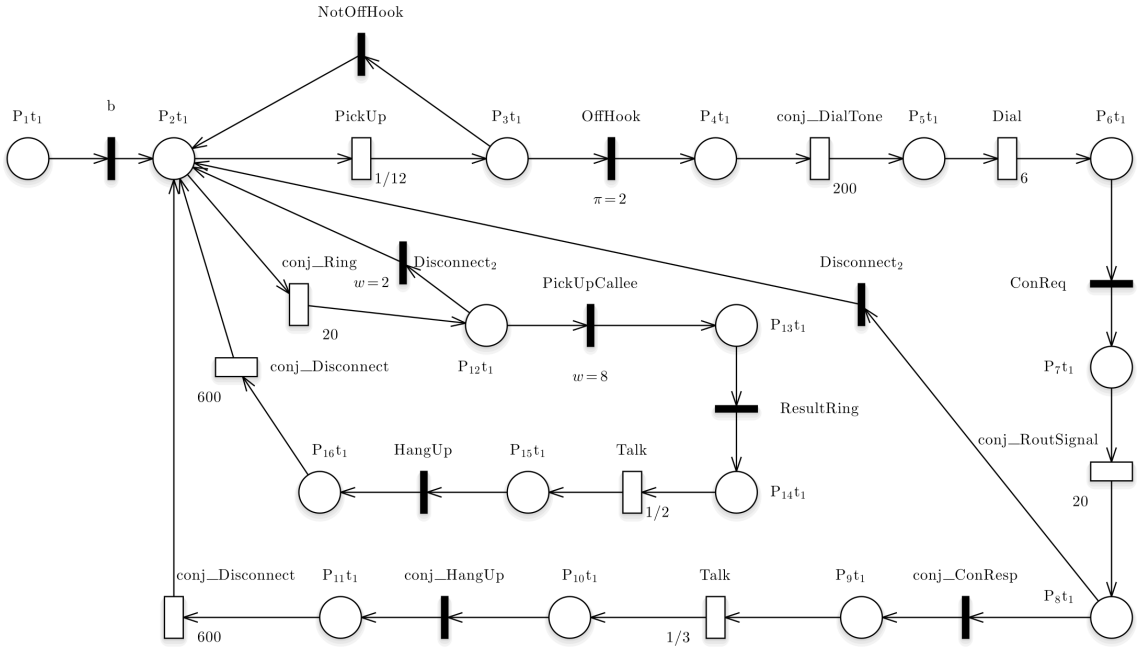


Fig. 2: gs-box for *Terminal*

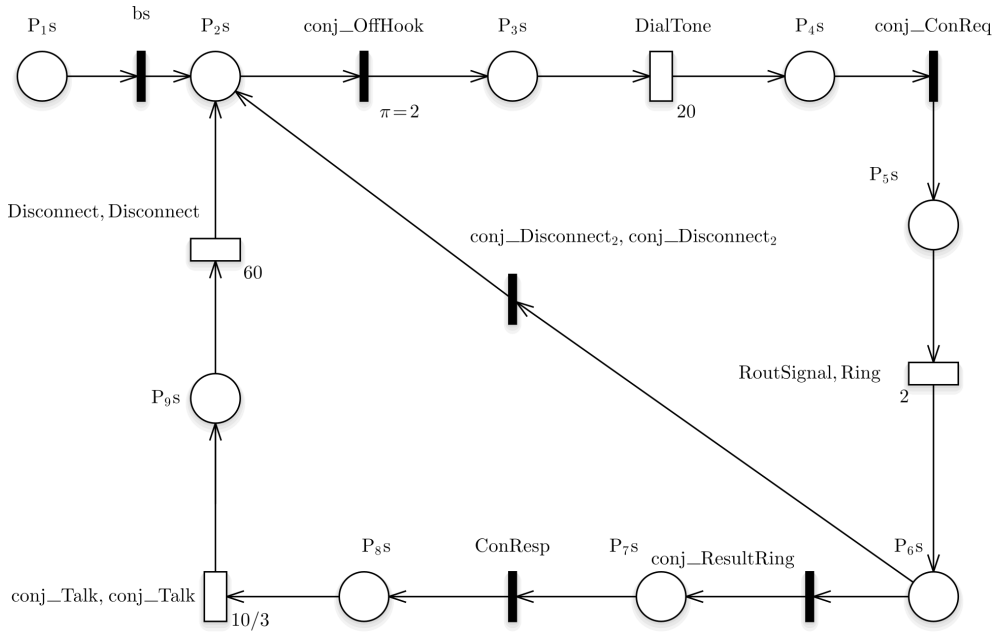


Fig. 3: gs-box for *Switch*

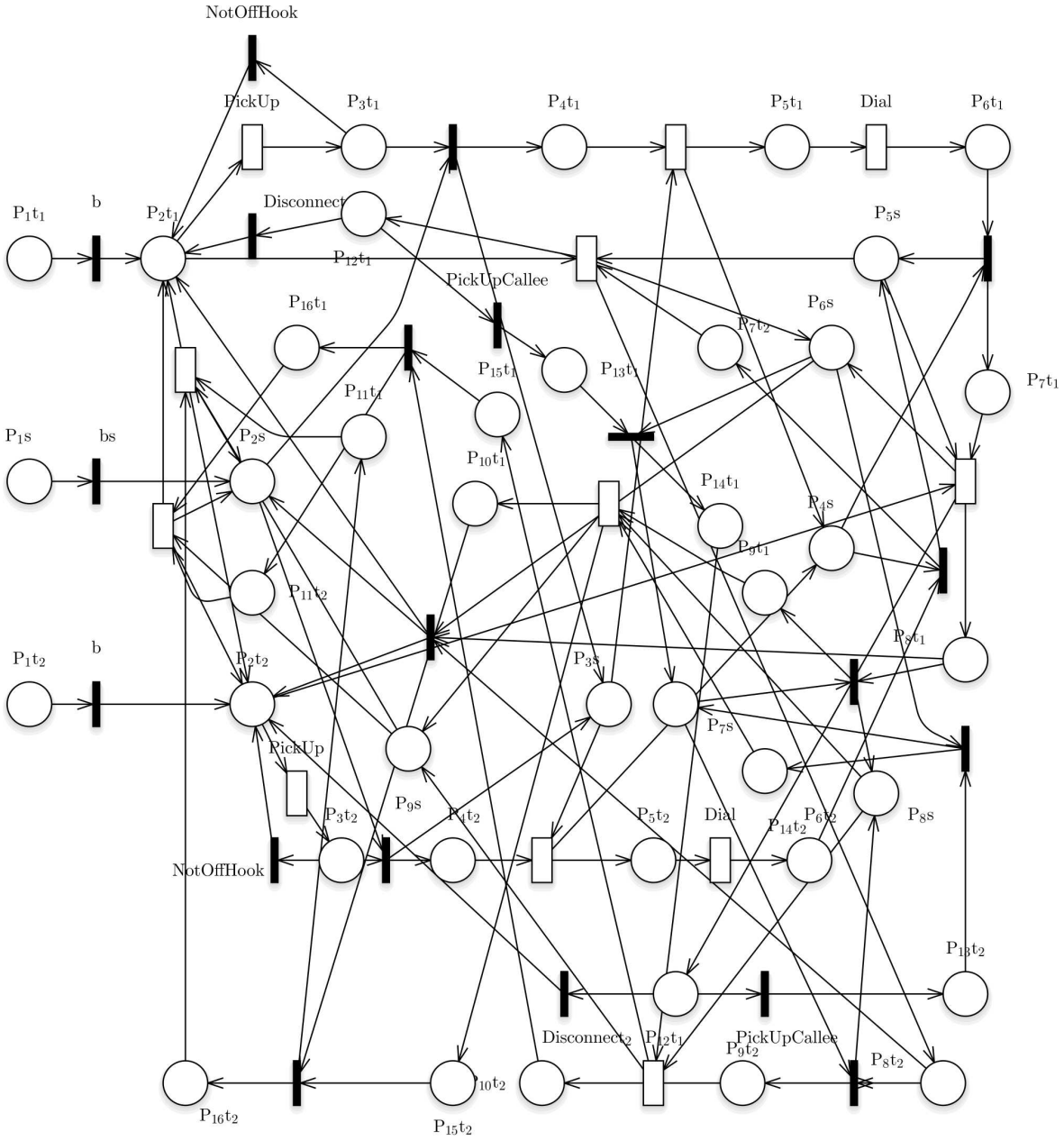


Fig. 4: gs-box for *Video ConferenceSystem*

Nevertheless, we can extend the model easily in order to allow several simultaneous videoconferences.

$$VideoConference_{n,3} = [\, A : Terminal_1 \| Terminal_2 \| \ldots \| Terminal_n \| Switch \| Switch \| Switch \,]$$

where
$A = \{OffHook, DialTone, ConReq, RoutSignal, Ring, ResultRing, ConResp, Talk, HangUp, Disconnect, Disconnect2\}$

Thus, three simultaneous videoconferences are enabled in this specific model. The most interesting case is, of course, a videoconference involving three or more people (a multiconference), for this case we need a new model for the *Switch*. In the following specification we allow a caller and 2 callees, i.e. three people are involved in the conversation.

$newSwitch =$
$[< b, \infty_{1,1} > * < \widehat{OffHook}, \infty_{1,1} >; < DialTone, 20 >;$
$< \widehat{ConReq}, \infty_{1,1} >; < \{RoutSignal, Ring, Ring\}, 2 >;$
$(< \{\widehat{ResultRing}, \widehat{ResultRing}\}, \infty_{1,1} >;$
$< ConResp, \infty_{1,1} >;$
$< \{\widehat{Talk}, \widehat{Talk}, \widehat{Talk}\}, 10/3 >;$
$< \{Disconnect, Disconnect, Disconnect\}, 60 >) \Box$
$< \{\widehat{Disconnect2}, \widehat{Disconnect2}, \widehat{Disconnect2}\}, \infty_{1,1} >$
$* < f, \infty_{1,1} >] rs\ f$

The specification of the whole system with three users and one switch is:

$newVideoConference\ =$
$[A : Terminal_1 || Terminal_2 || Terminal_3 || newSwitch]$

where $A =$
$\{OffHook, DialTone, ConReq, RoutSignal, Ring, ResultRing, ConResp, Talk, HangUp, Disconnect, Disconnect2\}$

4 Conclusions and Future Work

sPBC is a stochastic extension of PBC, which was presented in [16,14,17]. It is a semi-Markovian extension of PBC, which preserves the main features of that model. Thus, the syntax of sPBC is a natural stochastic extension of PBC, by annotating the multiactions with rates, which represent the parameters of exponential distribution. An important difference with respect to PBC is that in sPBC we define a semantics where no simultaneous execution of two multiactions is possible, although parallelism is maintained at the level of multiactions, as all the actions inside a multiaction are performed simultaneously.

In this paper we have considered an extended operational and denotational semantics of sPBC, by including immediate multiactions, in a similar way as they are considered in GSPNs. The denotational semantics of sPBC is defined using as semantic objects a special kind of labelled generalized stochastic Petri nets, called *gs-boxes*. An important characteristic of this translation is that it is static, in the sense that the rates or weights of the transitions will not be marking dependent.

Our main goal in this paper has been to show the flexibility and specification power of sPBC in the area of information science phenomena, modeling a *Video Conference System*. The main features of this language make an evidence the advantages of its use. Stochastic and immediate multiactions (with priorities and weights) are useful to describe control systems with quantitative information about times of actions and probabilities of execution, a special synchronization operator allowing multiway synchronization that considers the minimum conflict rate of the involved stochastic multiactions, which has the intuitive interpretation of taking the slowest one. Additionally, it has the iteration operator for repetitive behaviors, and, of course, an easy and natural translation to GSPNs, which allows us to apply some tools to obtain performance results. Consequently, we have both the advantages of using a simple stochastic process algebra language and Petri nets (GSPNs).

Our work in progress focuses on the definition of a stochastic bisimulation [15] that respects also immediate multiactions, which will capture precisely those processes that can be considered equivalent taking into account the stochastic information. Our plans for future work also include the treatment of the recursion operator and the develop of a particular tool based on sPBC and dtsiPBC.

Acknowledgement

This work received financial support from the Spanish Government (cofinanced by FEDER funds) through the TIN2012-36812-C02-02 Project. I.V. Tarasyuk was also supported in part by Deutsche Forschungsgemeinschaft (DFG), grant BE 1267/14-1, and Russian Foundation for Basic Research (RFBR), grant 14-01-91334. The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] M. Ajmone Marsan. Stochastic Petri Nets: An Elementary Introduction. *Lecture Notes in Computer Science; Advances in Petri Nets 1989*, 424:1–29, 1990.

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

[3] M. Bernardo and R. Gorrieri. A Tutorial on EMPA: A Theory of Concurrent Process with Nondeterminism, Priorities, Probabilities and Time. *Theoretical Computer Science, 202:1–54*, 1998.

[4] E. Best, R. Devillers, and M. Koutny. A Consistent Model for Nets and Process Algebra. In the book *The Handbook on Process Algebras*, J.A. Bergstra, A. Ponse and S.S. Smolka (Eds.), North Holland, Chapter 14, pp. 873–944, 2001.

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

[6] E. Best, R. Devilllers, and J. Hall. The Box Calculus: A New Causal Algebra with Multi-label Communication. In *Advances in Petri Nets*, G. Rozenberg (Eds.), LNCS 609, Springer, pp. 21–69, 1992.

[7] E. Best and M. Koutny. A Refined View of the Box Algebra. In *Application and Theory of Petri Nets 1995, 16th International Conference, Turin, Italy*, G. De Michelis and M. Diaz (Eds.), LNCS 935, Springer, pp. 1–20, 1995.

[8] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24:47–68, 1995.

[9] M.P. Gervais. Telecommunications Systems. In the book: *Petri Nets for Systems Engineering*, C. Girault and R. Valk (Eds.), Springer Verlag, Chapter 26, pp. 540–566, 2002.

[10] H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences and Axioms for MTIPP. In *Proc. of the 2nd Workshop on Process Algebra and Performance Modelling*, U. Herzog and M. Rettelbach, (Eds.) Regensberg/Erlangen, pp. 71–88, 1994.
[11] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, UK, 1996.
[12] GreatSPN homepage . http://www.di.unito.it/∼greatspn/index.html .
[13] M. Koutny. A Compositional Model of Time Petri Nets. In *Application and Theory of Petri Nets 2000, 21st International Conference, ICATPN 2000, Aarhus, Denmark*, M. Nielsen and D. Simpson (Eds.), LNCS 1825, pp. 303–322, Springer, 2000.
[14] H. Macià, V. Valero, D. Cazorla, and F. Cuartero. Introducing the iteration in sPBC. In *Proc. of Formal Techniques for Networked and Distributed Systems, FORTE 2004, 24th IFIP WG 6.1 International Conference, Madrid, Spain*, D. de Frutos and M. Núñez (Eds.), LNCS 3235, pp. 292–309, Springer, 2004.
[15] H. Macià, V. Valero, F. Cuartero, and D. de Frutos. A congruence relation for sPBC. *Formal Methods in System Design*, 32(2):85–128, 2008.
[16] H. Macià, V. Valero, F. Cuartero, and F. L. Pelayo. A new synchronization in finite stochastic Petri box calculus. In *Proc. 3rd International Conference on Application of Concurrent to System Design, ACSD 2003, Guimaraes, Portugal, pp. 216–225, IEEE Computer Society Press*, 2003.
[17] H. Macià, V. Valero, F. Cuartero, and M. C. Ruiz. sPBC: A Markovian Extension of Petri Box Calculus with Immediate Multiactions. *Fundamenta Informaticae, IOS Press*, 87(3-4):367–406, 2008.
[18] O. Marroquín and D. de Frutos. Extending the Petri Box Calculus with Time. In *Application and Theory of Petri Nets, 22nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK*, J.M. Colom and M. Koutny (Eds.), LNCS 2075, Springer, pp. 195–207, 2001.
[19] P. Merlin. *A Study of the Recoverability of Communication Protocols*. PhD thesis, Dep. of Computer Science, University of California, USA, 1974.
[20] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.
[21] M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models*. PhD thesis, UCLA, Los Angeles, USA, 1981.
[22] C. Ramchandani. *Perfomance evaluation of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 1973.
[23] M. Ribaudo. Stochastic Petri Net Semantics for Stochastic Process Algebra. In *Proc. 6th Int. Workshop on Petri Nets and Performance Models, (PNPM'95)*, Durham, USA, 1995.
[24] I. V. Tarasyuk. Stochastic Petri box calculus with discrete time. *Fundamenta Informaticae, IOS Press*, 76(1-2): 189–219, 2007.
[25] I. V. Tarasyuk, H. Macià, and V. Valero. Discrete Time Stochastic Petri Box Calculus with Immediate Multiactions dtsiPBC. *Electric Notes Theoretical Computer Science*, 296:229–252, 2013.

Hermenegilda Macià is an Associate Professor of the Department of Mathematics at the University of Castilla-La Mancha, in the Computer Science School of Albacete, Spain. She received her degree in Mathematics from University of Valencia, and her PhD in Computer Science from the University of Castilla-La Mancha in 2003. She has published research articles in reputed journals of mathematics and computer science. Her main research interests include the theoretical study and applications of Formal Methods such as Process Algebras and Petri Nets considering timed, probabilistic and stochastic extensions.

Valentín Valero is a full Professor of Distributed Systems and Operating Systems at the University of Castilla-La Mancha, in the Computer Science School of Albacete, Spain. He received his degree in Mathematics from the Complutense University of Madrid in 1987, and his PhD. in Mathematics in 1993 at the Department of Computer Science of the Complutense University of Madrid. Since October 1987 he is a member of the Computer Science Department at the University of Castilla-La Mancha. His current research areas are in the field of concurrency, specifically in formal models for analysis and design of concurrent systems and real-time systems.

Fernando Cuartero is a full Professor of Automata Theory and Formal Languages at the University of Castilla-La Mancha, in the Computer Science School of Albacete, Spain. He received his degree in Mathematics from the Complutense University of Madrid in 1985, and his PhD. in Mathematics in 1993 at the Department of Computer Science of the Complutense University of Madrid. Since October 1986 he is a member of the Computer Science Department at the University of Castilla-La Mancha. His current research areas are in the field of concurrency, specifically in Formal Models, and in High Performance Computing.

M. Carmen Ruiz received her M.Sc. degree in Computer Science from the University of Murcia, Spain in 1997. She got her Ph.D. degree in Computer Science in 2007 for the University of Castilla-La Mancha. She is currently Associate Professor at the Department of Computer Systems at the University of Castilla-La Mancha. She has published research papers in reputed international journals of mathematical and engineering sciences. Her research interests include systems evaluation by means of formal methods and performance evaluation.

Igor V. Tarasyuk is a senior researcher at A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences (IIS SB RAS), Novosibirsk, Russian Federation. He received the B.Sc. degree in Mathematics and Applied Mathematics from Novosibirsk State University in 1992, the M.Sc. degree in Mathematics and Computer Science from the same university in 1994, and the Ph.D. degree in Mathematics and Computer Science from IIS SB RAS in 1997. His scientific interests include timed, probabilistic and stochastic extensions of Petri nets, process algebras and behavioral equivalences. He has published research articles in reputed national and international journals in the areas of mathematics and computer science. He is the author of the monograph ”Equivalences for behavioural analysis of concurrent and distributed computing systems”, Geo Academic Publisher, Novosibirsk, 2007 (ISBN 978-5-9747-0098-9).